भाकृअनुप
ICAR

राअस्ट्रैप्रसं
NIASM

Twenty-One Day Online Training Manual
on

Advanced Statistical and Machine Learning Techniques for Data Analysis Using Open-Source Software for Abiotic Stress Management in Agriculture

16 July – 5 August 2025

Training Manual

Volume 01

Foundations of Open-Source Tools & Statistical Methods for Data Analysis

Edited by

Dr. Santosha Rathod
Dr. Nobin Chandra Paul
Ms. Ponnaganti Navyasree
Mr. K Ravi Kumar
Dr. Prabhat Kumar

Organised by:

School of Social Science and Policy Support
ICAR-National Institute of Abiotic Stress Management, Baramati, Maharashtra - 413115

Foundations of Open-Source Tools & Statistical Methods for Data Analysis

Editors

Santosha Rathod

Nobin Chandra Paul

Ponnaganti Navyasree

K Ravi Kumar

Prabhat Kumar

2025

School of Social Science and Policy Support
ICAR-National Institute of Abiotic Stress Management, Baramati – 413115
Maharashtra, India

Title: Foundations of Open-Source Tools & Statistical Methods for Data Analysis

Editors: Santosha Rathod, Nobin Chandra Paul, Ponnaganti Navyasree, K Ravi Kumar, Prabhat Kumar

Published by: ICAR-National Institute of Abiotic Stress Management, Malegaon Khurd, Baramati – 413115, Maharashtra, India.

Edition: I

Volume: 1

ISBN: 978-81-985897-6-7

Citation:

Rathod, S., Paul, N. C., Ponnaganti, N., Kumar, K. R., & Kumar, P. (Eds.). (2025). *Foundations of open-source tools & statistical methods for data analysis: Training manual of the twenty-one-day online training programme on "Advanced statistical and machine learning techniques for data analysis using open-source software for abiotic stress management in agriculture"* (Vol. 1). ICAR-National Institute of Abiotic Stress Management. ISBN 978-81-985897-6-7.

CONTENTS

Importance of Data Science in Abiotic Stress Management in Agriculture

K Sammi Reddy

Director, ICAR-National Institute of Abiotic Stress Management, Baramati

Introduction

Abiotic stresses refer to the negative impact of non-living factors on plants and agricultural productivity. These stresses include drought, extreme temperatures (both heat and cold), salinity, flooding, nutrient deficiency, and radiation. Unlike biotic stresses caused by pests and diseases, abiotic stresses originate from environmental and edaphic (soil-related) conditions. These stressors can severely hamper plant growth, reduce yields, and threaten food security, especially under changing climate scenarios.

Common types of abiotic stress include:

- **Drought Stress**: Caused by prolonged periods without rainfall or inadequate irrigation.
- **Heat Stress**: Exposure to excessively high temperatures.
- **Cold Stress**: Low temperature damage during germination or flowering.
- **Salinity Stress**: High salt concentration in soil or irrigation water.
- **Waterlogging**: Saturated soil conditions leading to oxygen deficiency.
- **Nutrient Deficiency**: Lack of essential macro and micronutrients in the soil.

Problems Associated with Abiotic Stress

Abiotic stress is a major constraint in achieving optimum crop productivity. It affects plant physiology, reduces nutrient uptake, disturbs hormonal balance, and leads to oxidative damage in plant cells. These stresses are often interlinked and occur simultaneously, making management more challenging. Some of the major problems are:

- Yield losses and poor crop quality
- Reduced germination and delayed flowering

- Disruption of reproductive development
- Soil degradation and reduced microbial activity
- Increased vulnerability to pests and diseases
- Economic losses to farmers and food supply chain disruptions

Mitigation Strategies for Abiotic Stress

Managing abiotic stress involves both preventive and adaptive measures:

- **Breeding for Stress Tolerance**: Developing stress-resilient varieties through conventional and molecular breeding.
- **Soil and Water Management**: Enhancing soil health, using mulching, drip irrigation, and rainwater harvesting.
- **Climate-Smart Agriculture**: Adopting practices that increase resilience to climate variability.
- **Integrated Nutrient Management**: Ensuring balanced fertilization and use of biofertilizers.
- **Crop Diversification and Rotation**: Minimizing risk by growing a variety of crops.
- **Early Warning Systems**: Using weather forecasting tools for timely interventions.

Types of Data in Abiotic Stress Research

Effective management of abiotic stresses relies on the timely collection, integration, and analysis of diverse datasets. These data are gathered using a combination of traditional agronomic methods and modern technological tools such as remote sensing, IoT sensors, and cloud-based platforms. The accuracy and richness of these datasets are crucial for understanding the complex interactions between crops and environmental stressors, enabling better prediction, early warning, and targeted interventions. To manage abiotic stresses effectively, various types of data are collected and analyzed:

- **Climatic Data**: Temperature, rainfall, humidity, wind speed (from IMD).
- **Soil Data**: Soil moisture, pH, nutrient levels, EC, and structure.

- **Crop Data**: Growth stages, yield, phenology, stress symptoms.
- **Remote Sensing Data**: Satellite imagery for land use, crop health, and drought monitoring.
- **Sensor Data**: IoT-based sensors for real-time field monitoring (temperature, soil moisture).
- **Genomic and Phenotypic Data**: Information on stress-resilient genes and plant traits.
- **Socioeconomic Data**: Farmer practices, inputs, market access, and financial status.

Role of Data Science in Abiotic Stress Management

Data science plays a transformative role in managing abiotic stress by enabling data-driven decisions. Key applications include:

- **Predictive Modeling**: Machine learning models can predict drought events, crop yield losses, or heat waves using historical and real-time data.
- **Decision Support Systems (DSS)**: Integrating multiple data sources to provide farmers with timely recommendations.
- **Remote Sensing and GIS**: Analyzing spatial and temporal patterns of stress using satellite and drone imagery.
- **Big Data Analytics**: Handling large datasets to identify trends, correlations, and anomalies.
- **Precision Agriculture**: Site-specific recommendations for irrigation, fertilization, and crop management.
- **Sensor Integration**: Real-time monitoring of field conditions with smart devices for proactive responses.
- **Genomic Data Mining**: Identifying genes responsible for stress tolerance using AI/ML approaches.

Statistical Tools and Data Science Techniques in Abiotic Stress Research

Statistical methods and data science techniques are essential for analyzing the complex interactions in abiotic stress environments. These methods enable the quantification, visualization, and interpretation of stress responses in crops. Key techniques include:

- **Regression Analysis**: Used to understand relationships between stress factors (like temperature, rainfall) and crop yield.
- **Multivariate Analysis**: Helps in identifying key abiotic factors from multiple variables using PCA, cluster analysis, or discriminant analysis.
- **Time Series Analysis**: Evaluates seasonal and trend patterns in stress data over time.
- **Geo-Statistics**: Spatial interpolation methods like kriging to map soil moisture, salinity, or nutrient levels.
- **Machine Learning Algorithms**: Decision trees, random forests, SVMs, and neural networks to classify stress severity and predict outcomes.
- **Bayesian Models**: Useful in integrating prior knowledge and managing uncertainty in stress predictions.
- **Risk Modeling**: Quantitative risk assessment for forecasting production losses due to climate extremes.

These tools not only support research but also provide actionable insights for farmers and policymakers to take early interventions.

Role of ICAR-NIASM in Abiotic Stress Management in Indian Agriculture

The ICAR–National Institute of Abiotic Stress Management (ICAR-NIASM), located at Malegaon Khurd, Baramati, Maharashtra, is a premier institute under the Indian Council of Agricultural Research (ICAR). Established in 2009, ICAR-NIASM is dedicated to exploring innovative avenues for managing abiotic stresses that threaten the sustainability of India's food production systems. The institute focuses on mitigating stresses arising from atmospheric, water-related, and edaphic (soil) factors, which together are estimated to cause major losses in crop productivity.

With climate change and land degradation expected to amplify the frequency and intensity of these stresses, ICAR-NIASM's primary mission is to develop practical alleviation strategies through cutting-edge research in frontier sciences. Equipped with state-of-the-art laboratories, research farms, and multidisciplinary expertise, the institute develops climate-resilient technologies, decision support systems, and sustainable farming practices to help farmers adapt to and manage abiotic stresses effectively.

Conclusion

Abiotic stress is a major challenge facing Indian agriculture, especially in the context of climate change. ICAR-NIASM plays a crucial role in understanding and mitigating these stresses through innovative research. The integration of data science into abiotic stress management has revolutionized how we detect, predict, and respond to these challenges. By leveraging advanced tools such as AI, remote sensing, sensor networks, and statistical modeling, data science enables real-time monitoring and decision-making for farmers and policymakers. It empowers the development of stress-resilient crop varieties, site-specific agro-advisories, and sustainable farming practices.

Looking ahead, data-driven approaches will play an increasingly vital role in tackling the challenges of abiotic stress. Building capacity in data analytics, development interdisciplinary collaboration, and investing in digital agriculture infrastructure will be essential strategies. The integration of agricultural science with data science offers a transformative path toward achieving climate resilience and ensuring food security in India.

R Installation Guide

Santosha Rathod, Nobin Chandra Paul, Ponnaganti Navyasree, K. Ravi Kumar and Prabhat Kumar

ICAR-National Institute of Abiotic Stress Management, Baramati, Pune-413115
santosha.rathod@icar.org.in

Installation of R Version 4.1.2.

Step 1: Download the R Version 4.5.1 (32/64 bit) from The Comprehensive R Archive Network (CRAN) website using following links

Link to download R 4.5.1 for Windows

https://cran.r-project.org/bin/windows/base/

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed man

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.html.

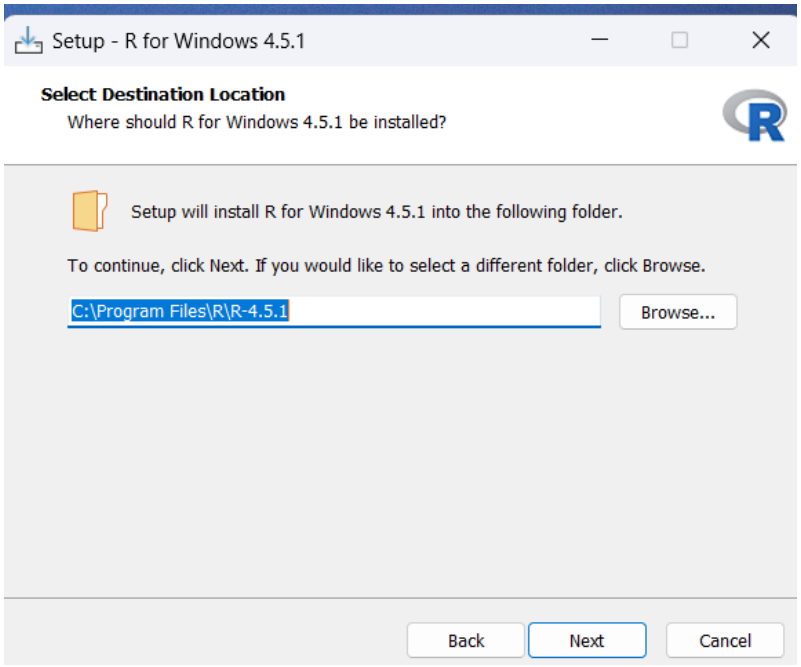
Last change: 2025-06-13

Link to download R for macOS

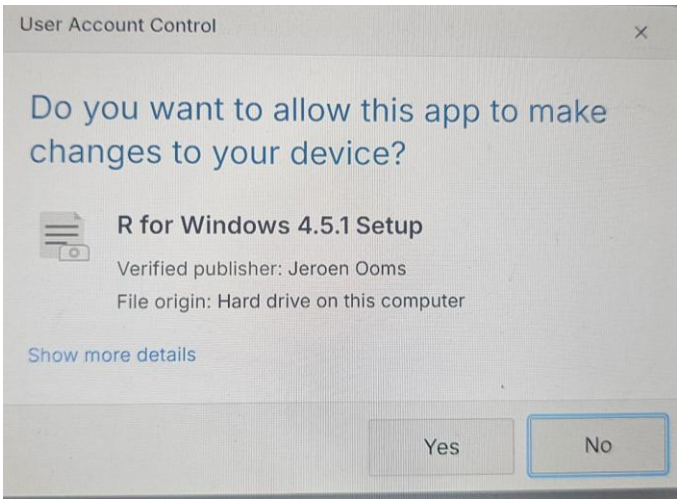
https://cran.r-project.org/bin/macosx/

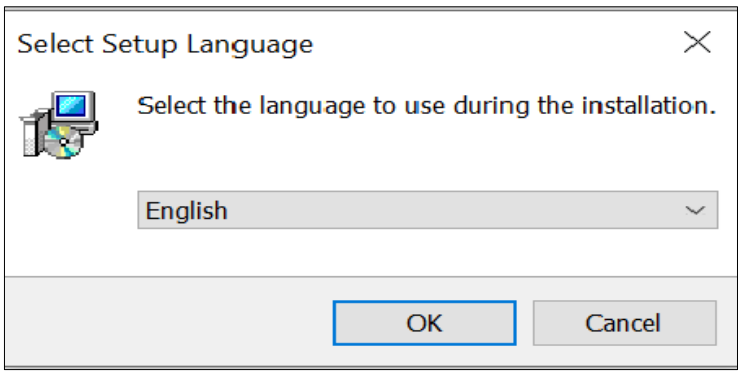
Step 2: Installation of R Version 4.5.1. (32/64 bit) for windows

Click on R-4.5.1-win.exe file and select English language in set up step

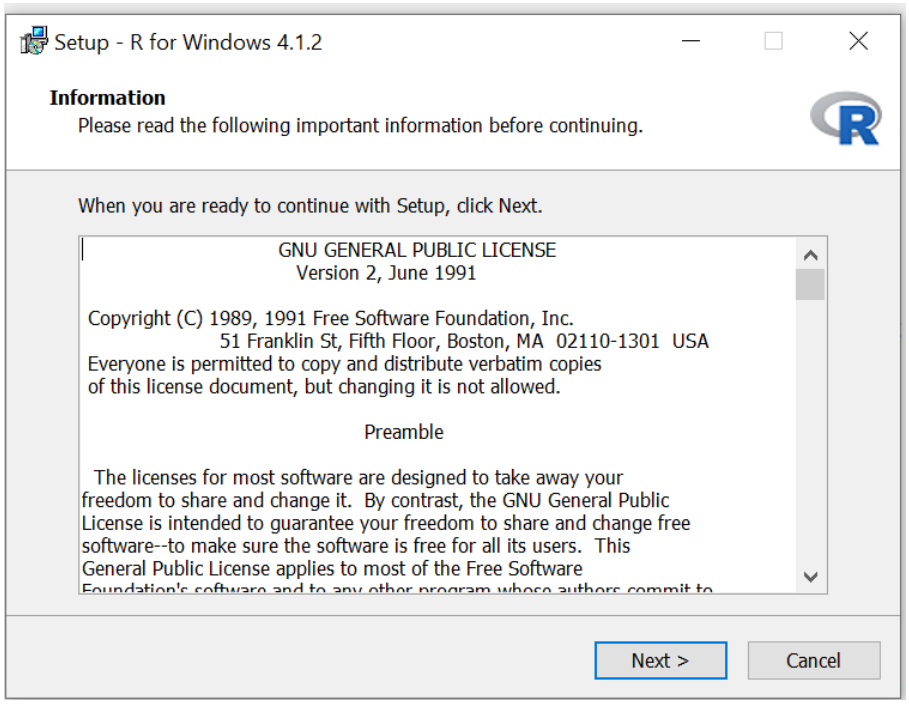


Simillary do it for 4.5.1



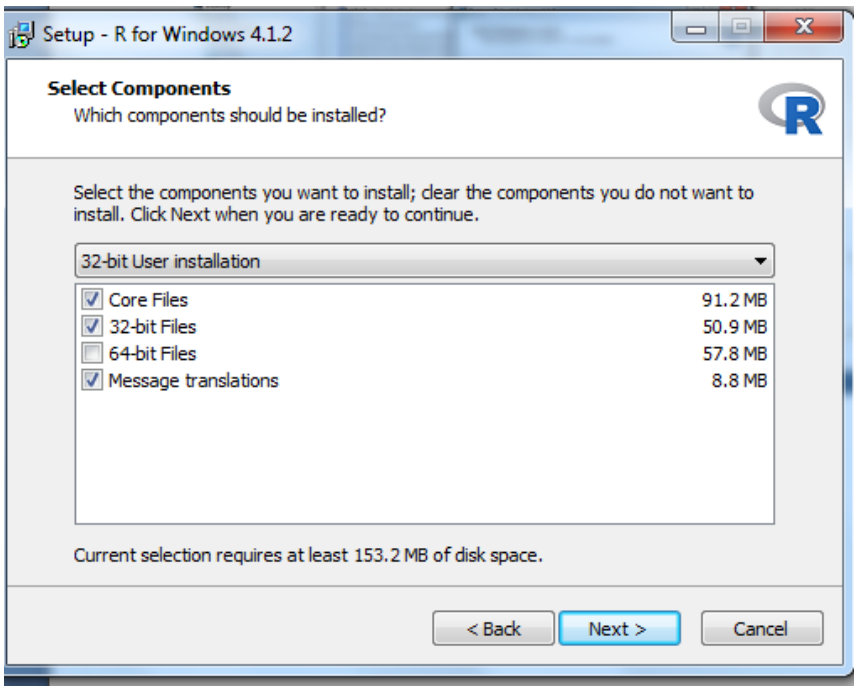


Step 3: Click next to install 4.5.1

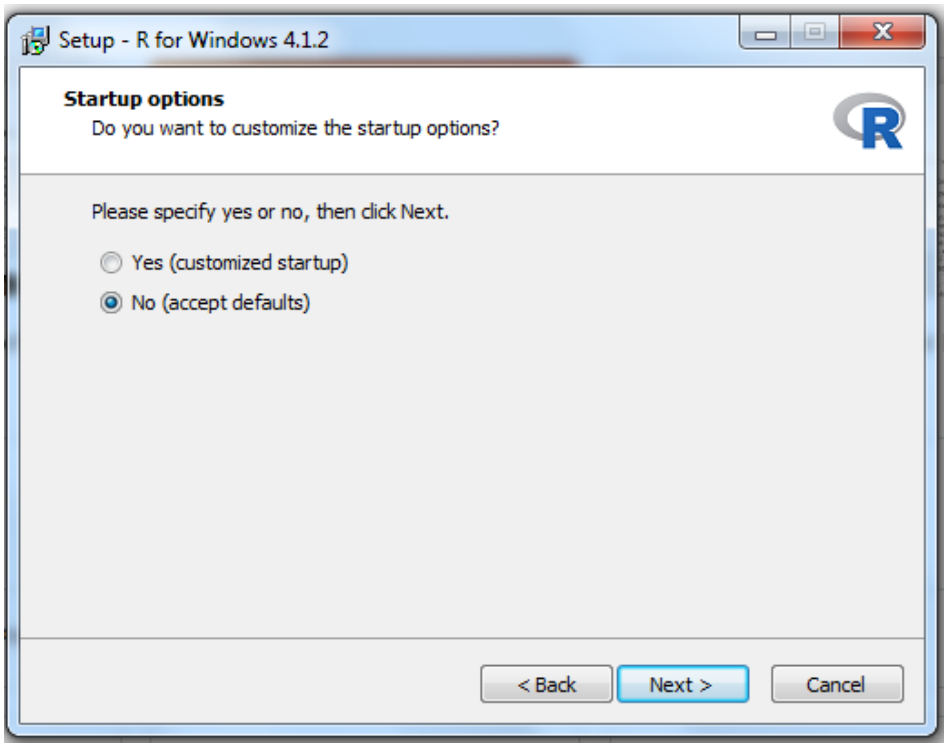


Step 4: select destination folder/location and click on next

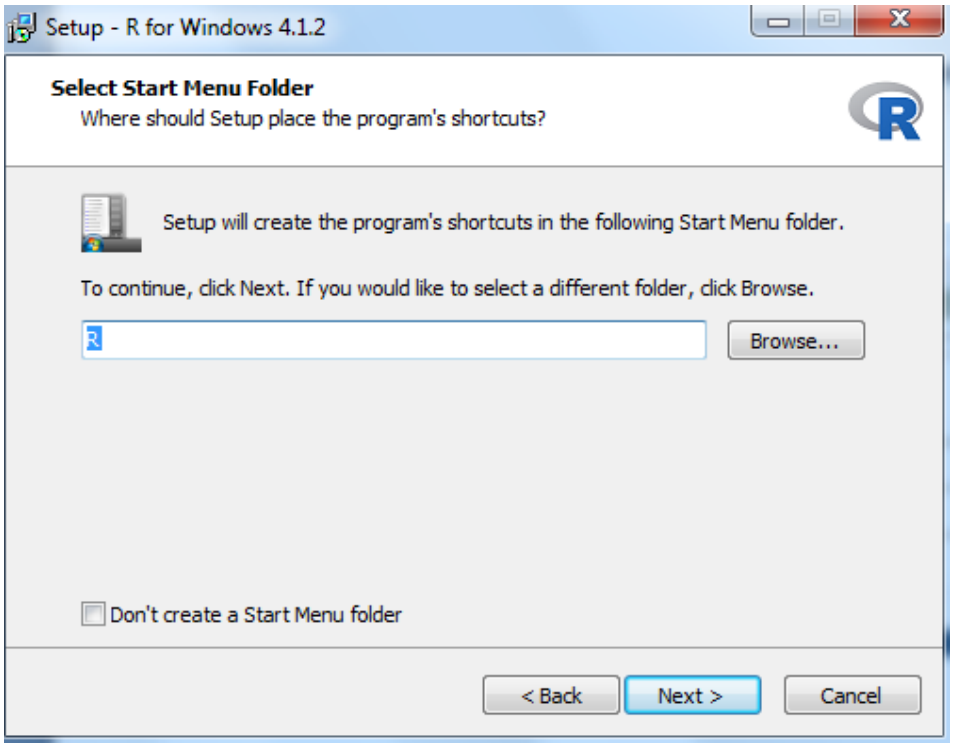
Step 5: Select 32-bit or 64-bit files, based on OS bit type and select Next



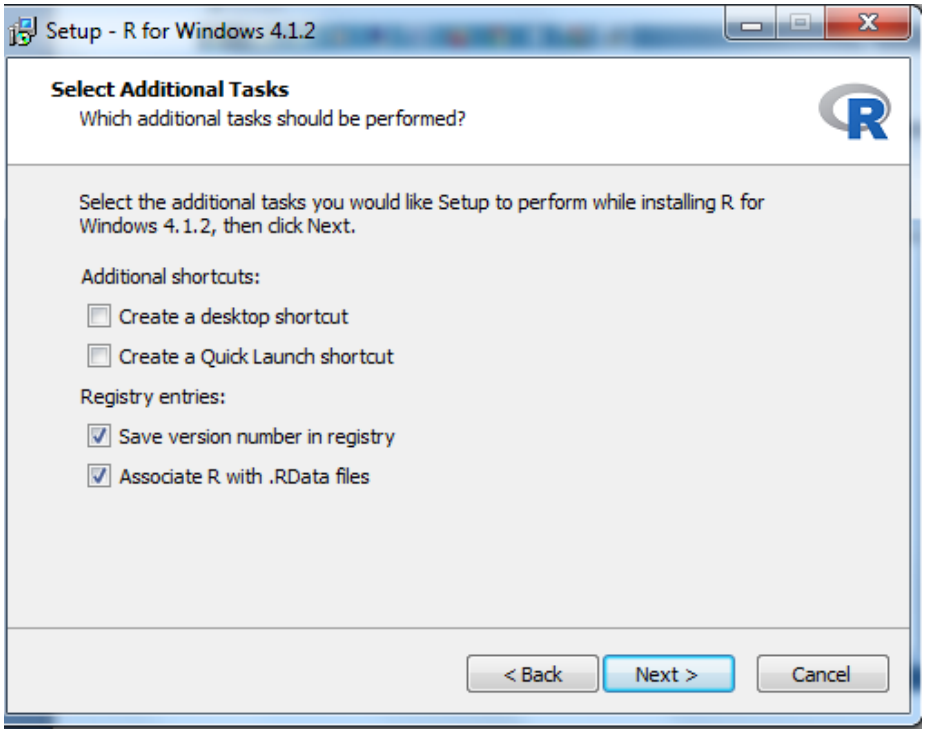
Step 6: Click on No (accept defaults) and click Next.



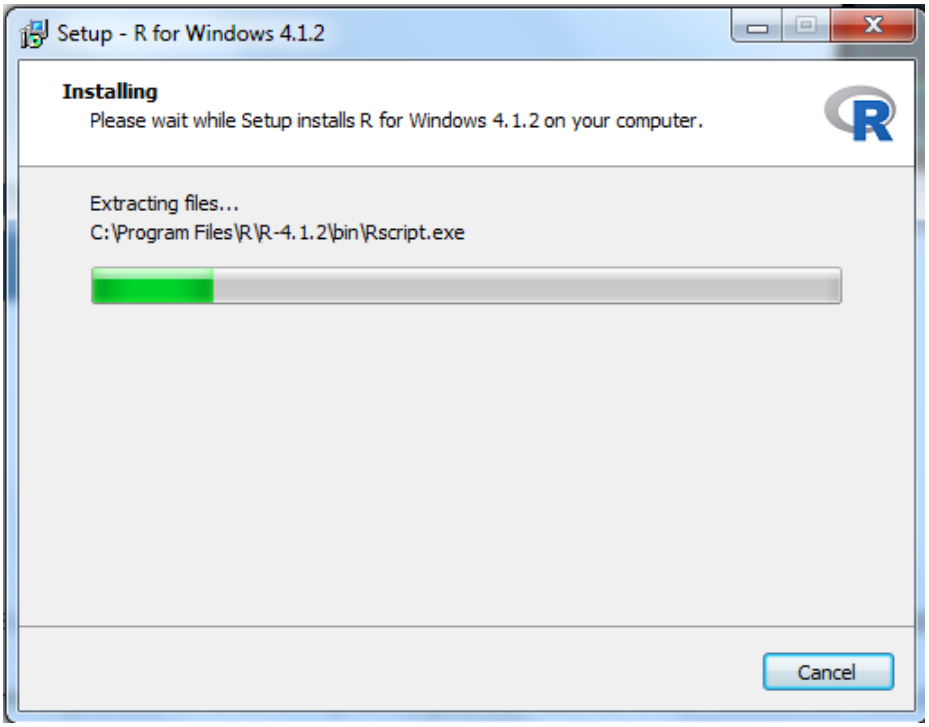
Step 7: Select start menu folder and select Next (Keep default)



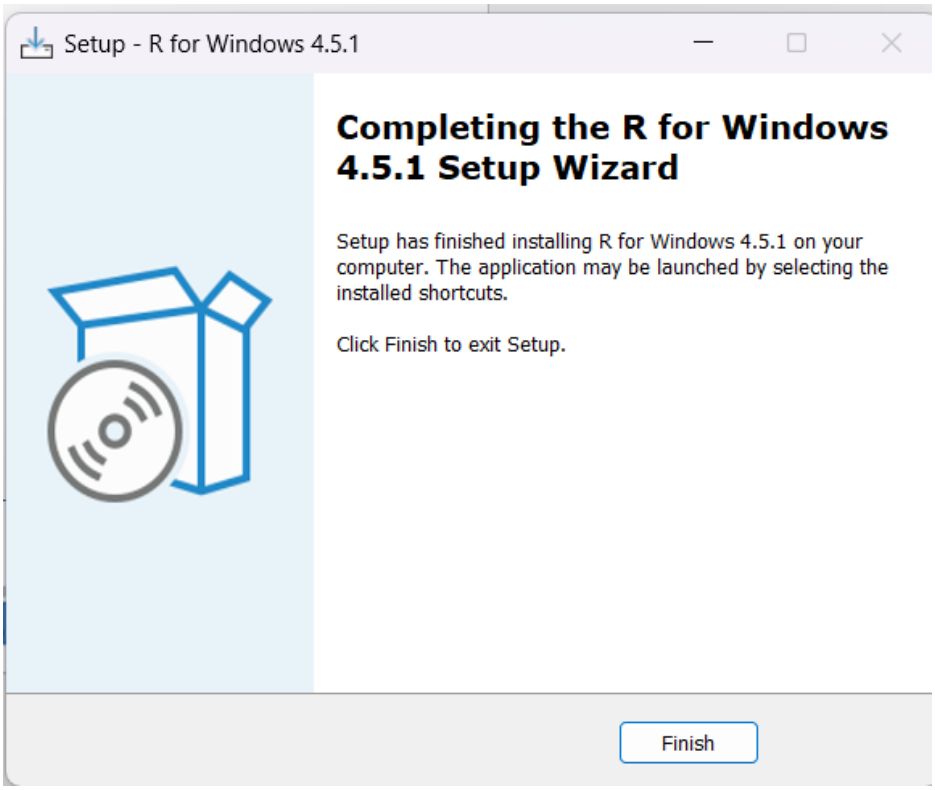
Step 8: Select Registry entries and don't select additional shortcuts



Step 9: Installing 4.5.1, click next



Step 10: Installation complete, click on Finish

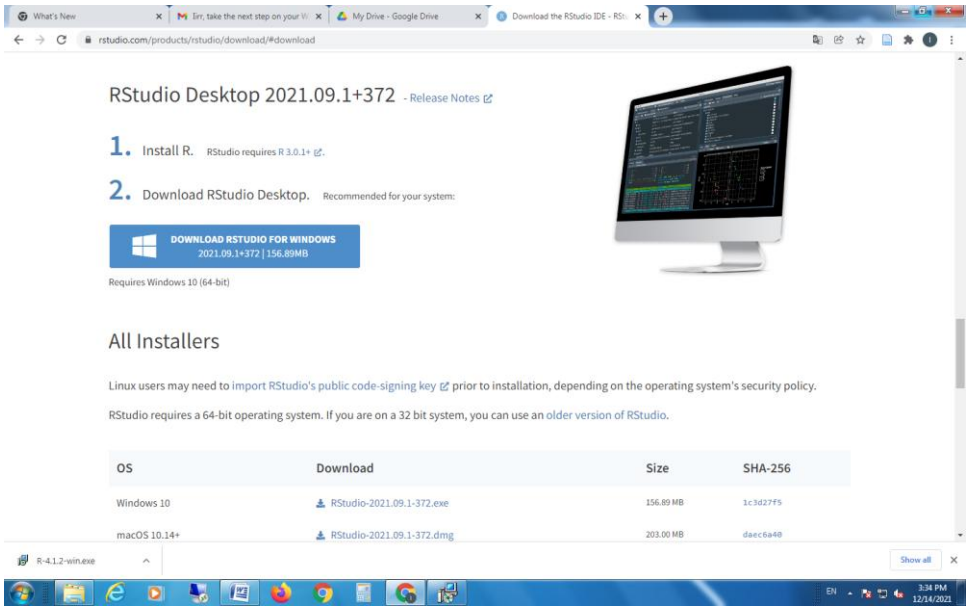


Installation of R Studio

Download R studio: Click on

https://www.rstudio.com/products/rstudio/download/#download

Latest Version is : Version: 2025.05.1+513 | Released: 2025-06-05



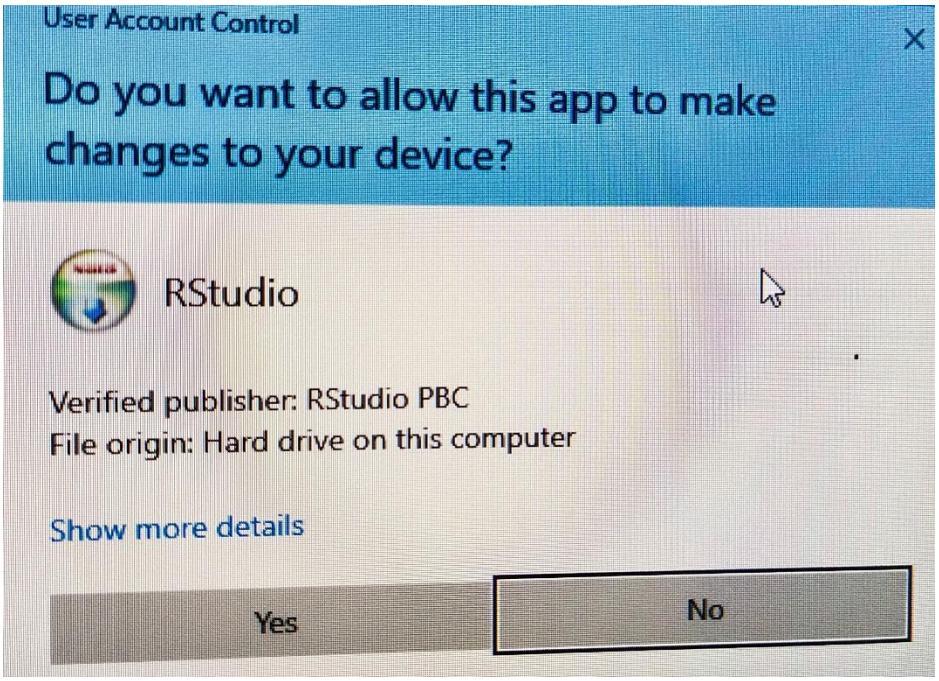
2: Install RStudio

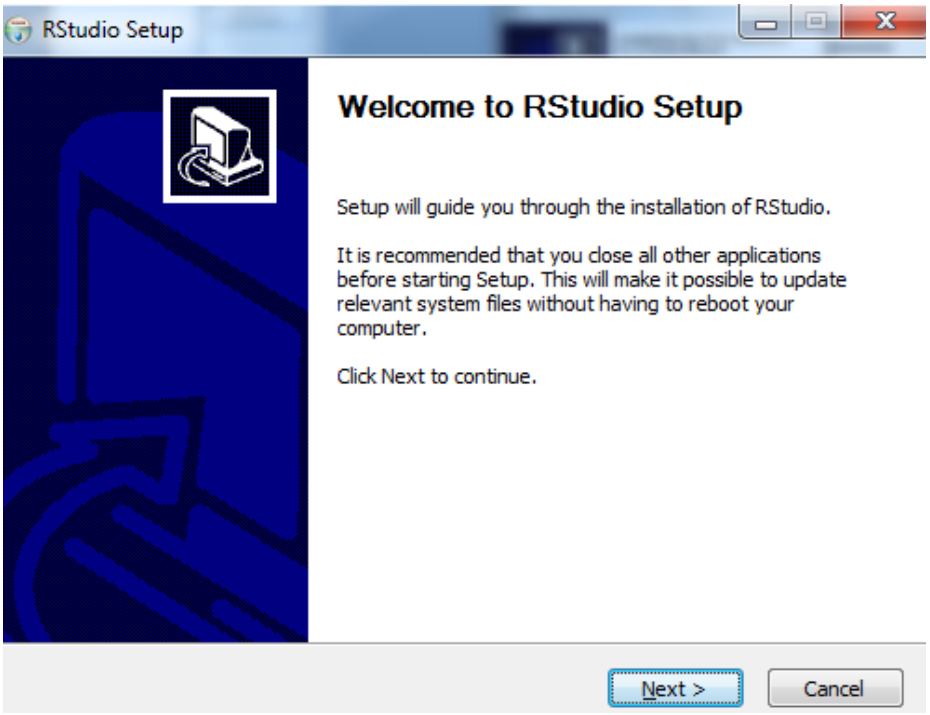
DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS

Size: 281.24 MB | SHA-256: 3A553330 | Version: 2025.05.1+513 | Released: 2025-06-05

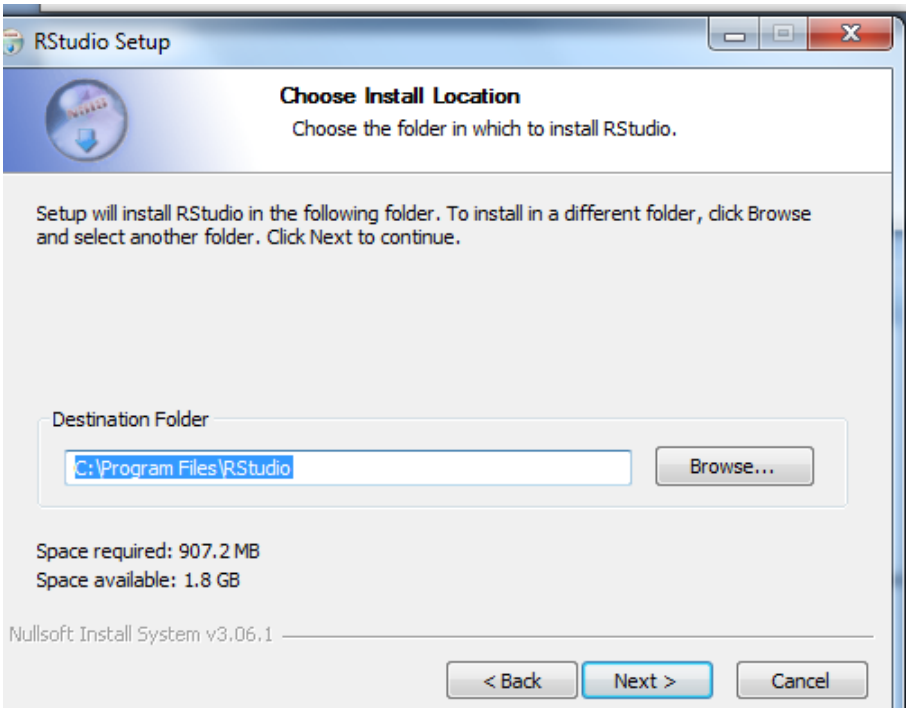
Installation steps

Step 1: Welcome to RStudio Setup - Click next

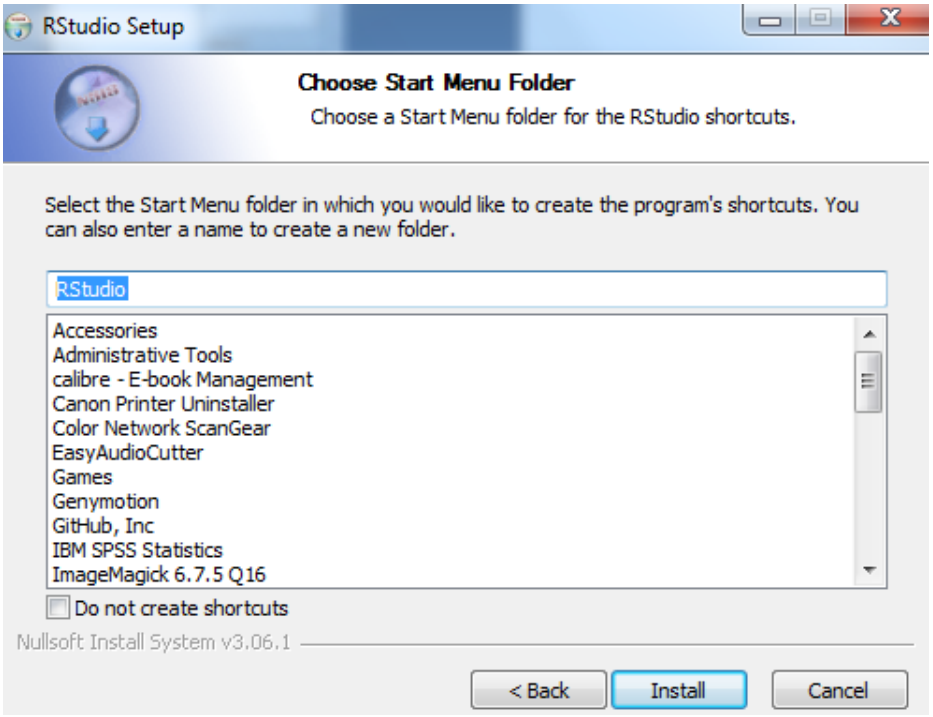




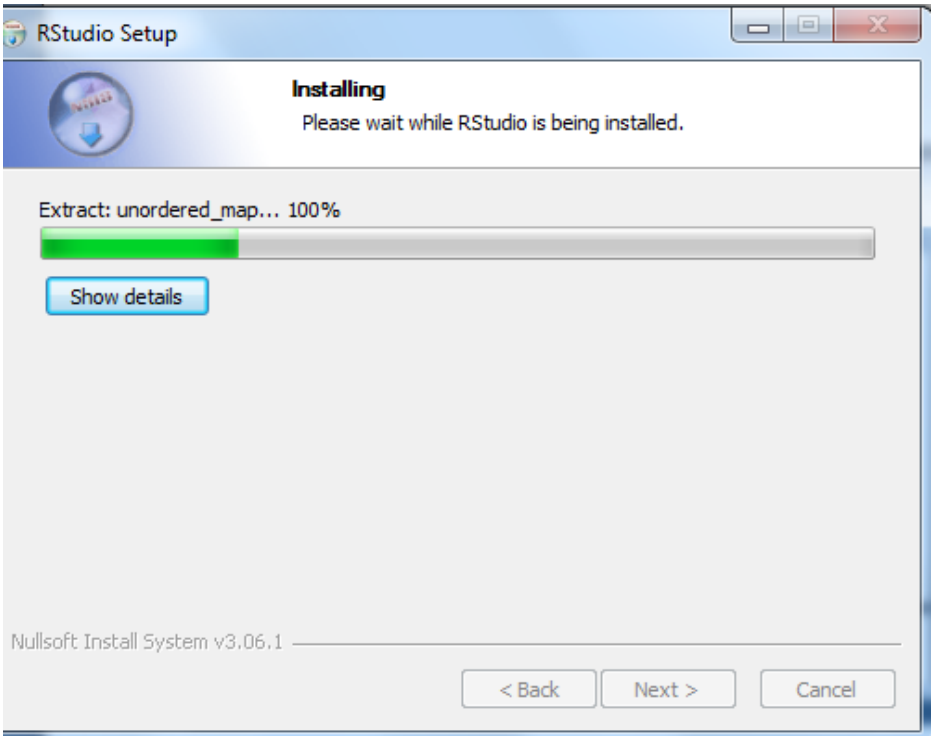
Step 2: Choose installation location – choose default



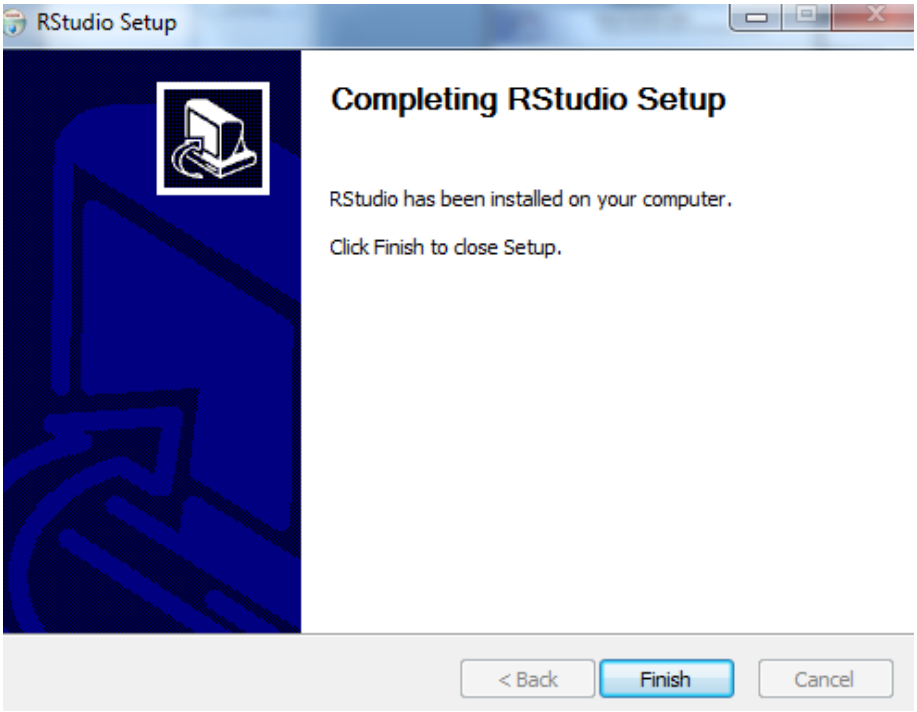
Step 3: Choose start menu folder – choose default RStudio



Step 4: Installing – Click on next



Step 5: Completing RStudio Setup – Click on Finish



Installation of R tools

Download Rtools45 from

https://cran.r-project.org/bin/windows/Rtools/rtools45/rtools.html

https://cran.r-project.org/bin/macosx/tools/

Rtools45 for Windows

Rtools is a toolchain bundle used for building R packages from source (those that need compilation of C/C++ or Fortran code) and for building R itse become R 4.6.0.

Rtools45 consists of Msys2 build tools, GCC 14/MinGW-w64 compiler toolchain, libraries built using the toolchain, and QPDF. Rtools45 supports 6

Compared to Rtools44, Rtools45 for 64-bit Intel machines has newer versions of two core components: GCC and binutils. It is recommended to re-cc

Rtools45 is also available for 64-bit ARM machines (aarch64): it includes Msys2 build tools (64-bit Intel builds running via emulation) and aarch64 t toolchain, and again QPDF. The 64-bit ARM version of Rtools45 is experimental: a number of CRAN packages don't work with it and the Fortran cc number of CRAN packages don't work because they require not-yet-available 64-bit ARM versions of external software.

Installing Rtools45

Rtools is only needed for installation of R packages from source (those that need compilation of C/C++ of Fortran code) or building R from source. F versions of CRAN packages, which does not require Rtools45.

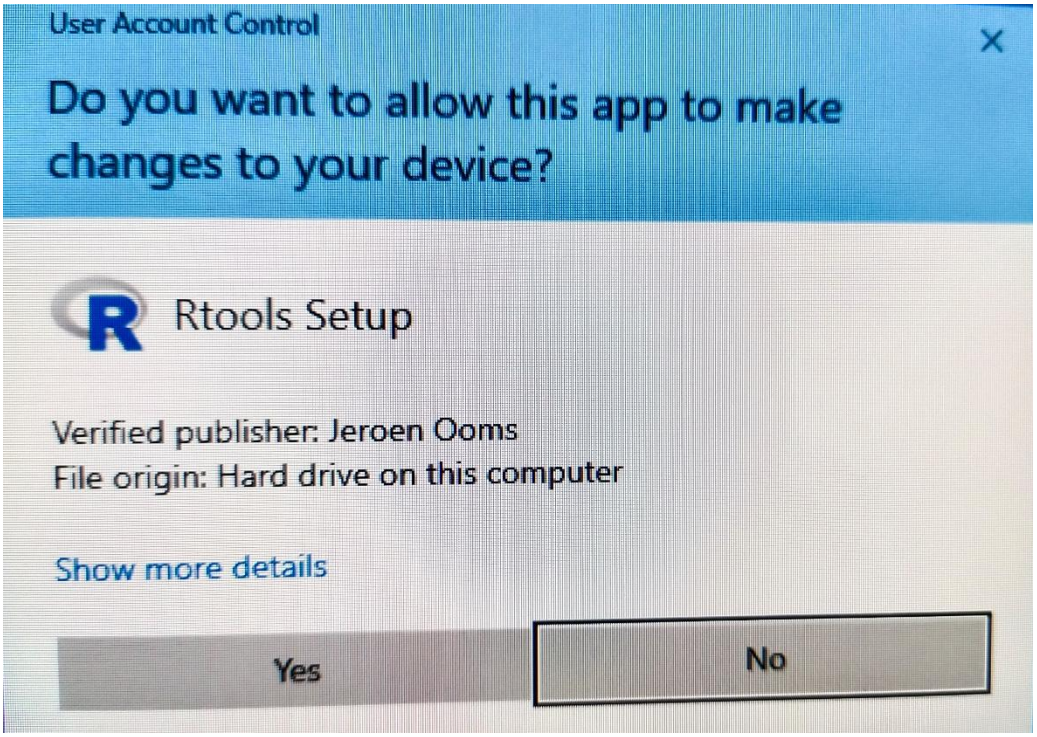
Moreover, online build services are available to check and build R packages for Windows, for which again one does not need to install Rtools45 loca packages checks and has already all CRAN and Bioconductor packages pre-installed.

Rtools45 may be installed from the Rtools45 installer or 64-bit ARM Rtools45 installer. It is recommended to use the defaults, including the default i

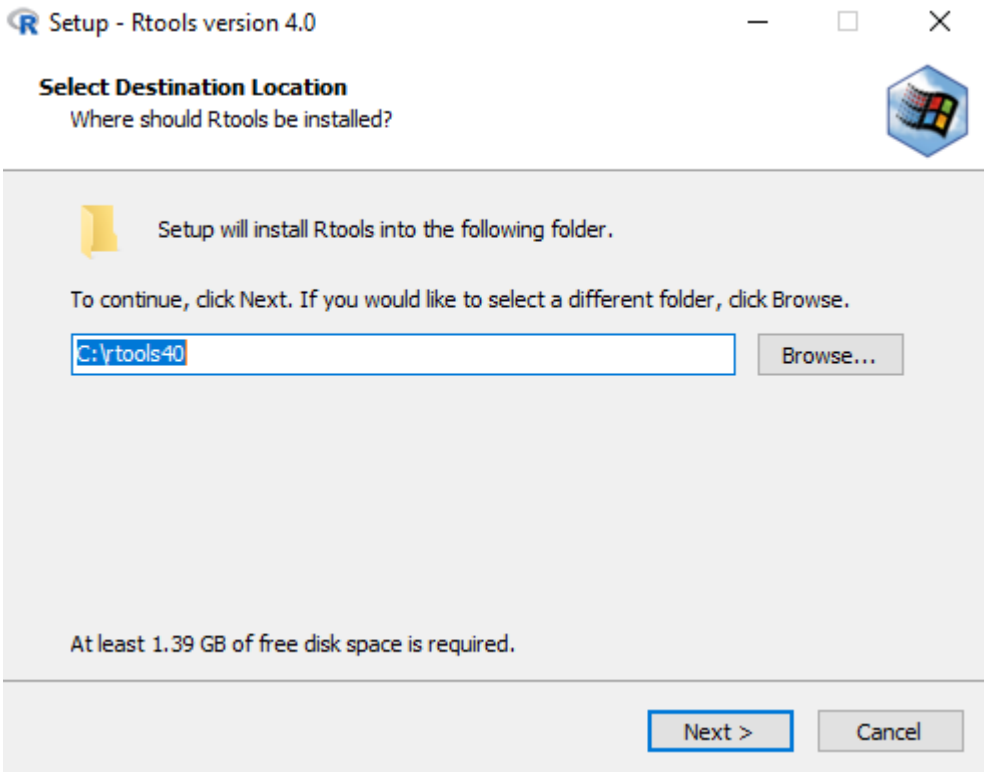
When using R installed by the installer, no further setup is necessary after installing Rtools45 to build R packages from source. When using the defau may be installed when R is already running.

On ARM, binary versions of packages are currently not available from CRAN, so Rtools45 is required to install any package that needs compilation.

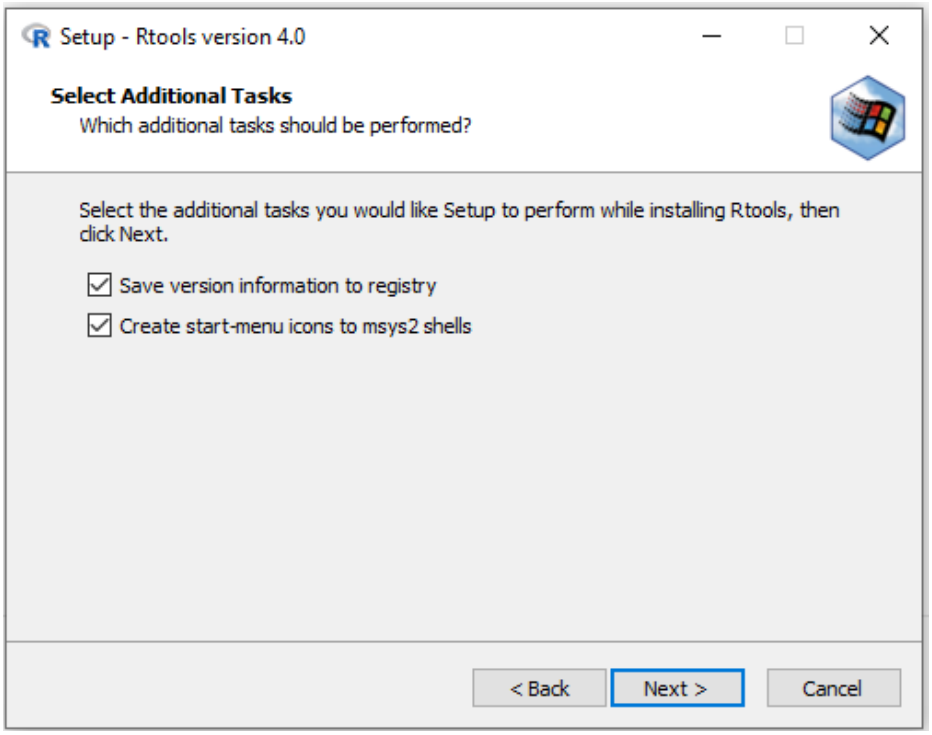
Step 1: Install Rtools45



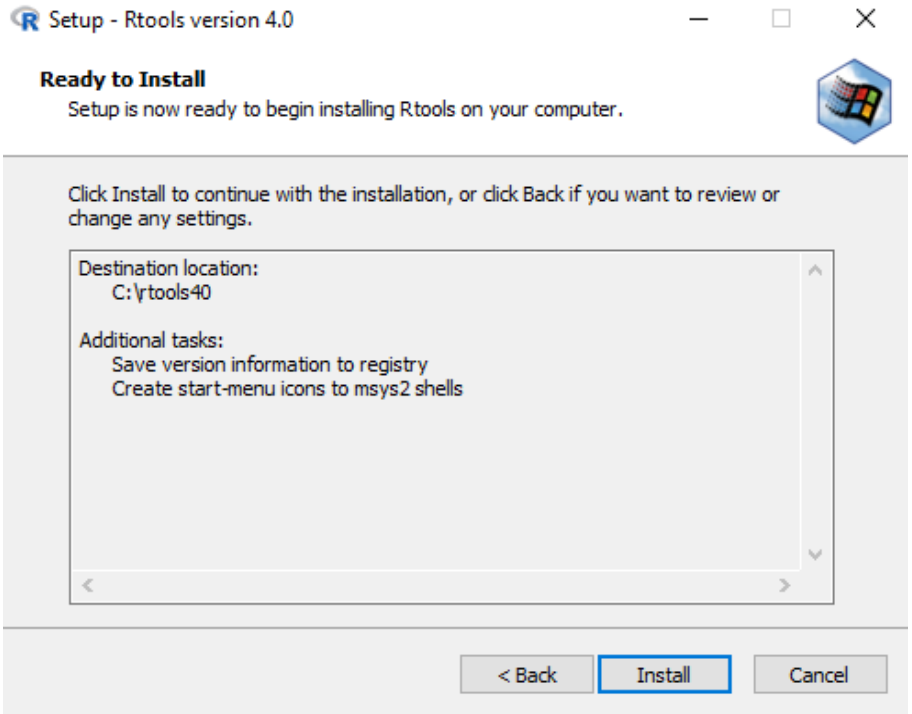
Step 2: Select Destination Location for Rtools45



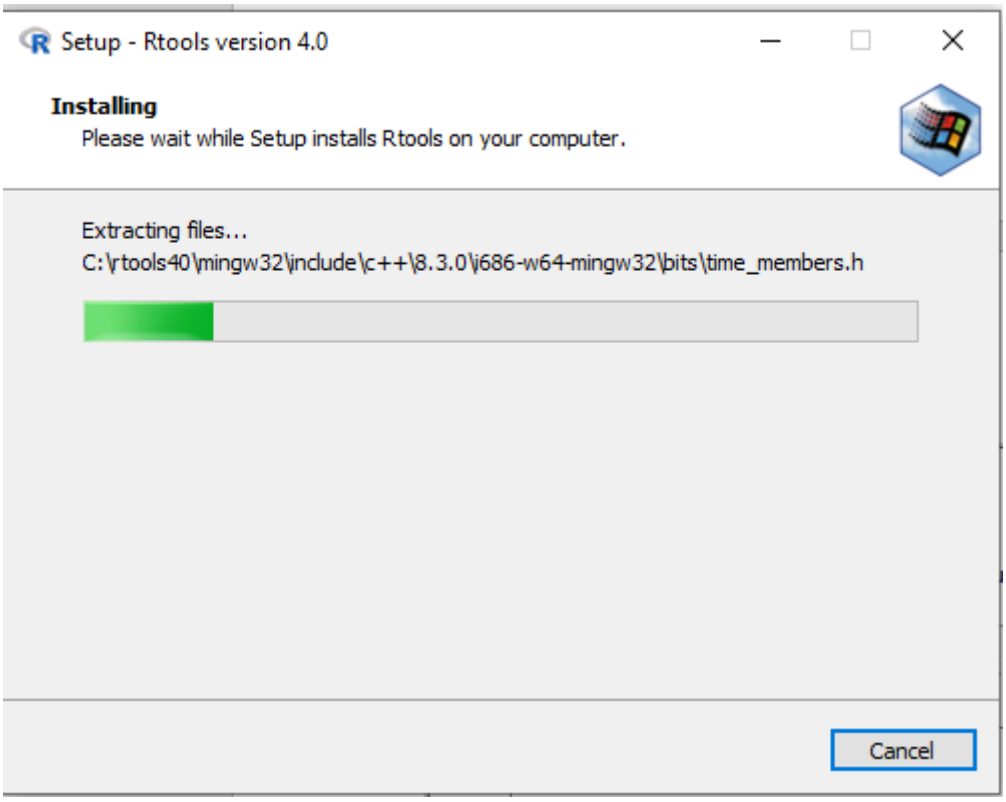
Step 3: Select Additional tasks – select both



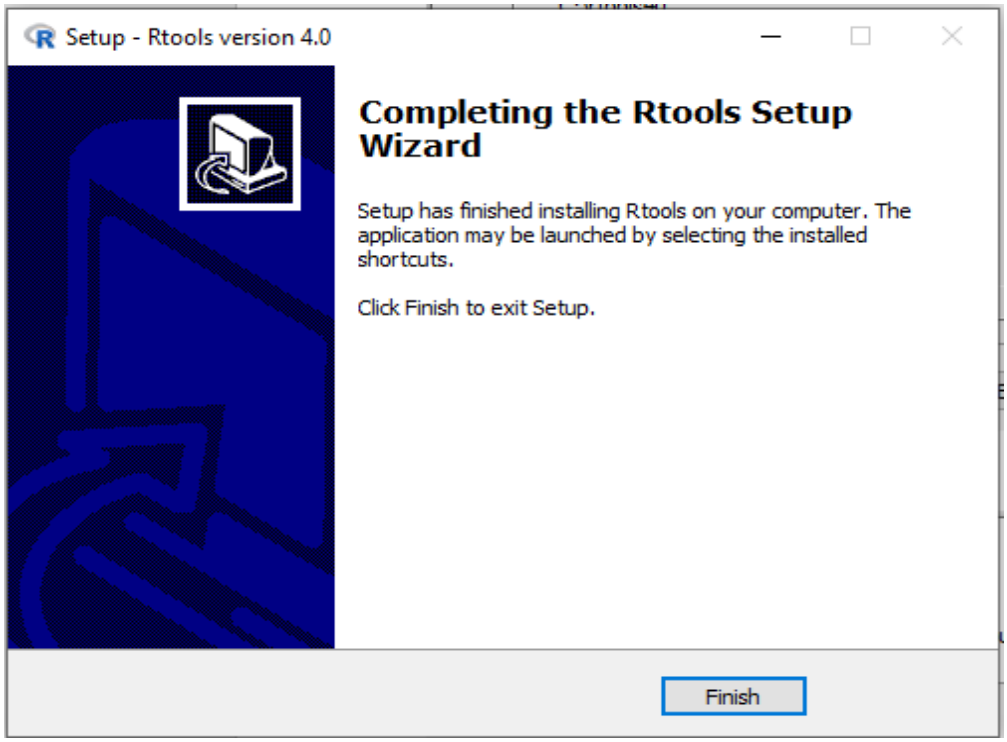
Step 3: Select Additional tasks – select both- click on install



Step 4: Installing



Step 5: Installation complete

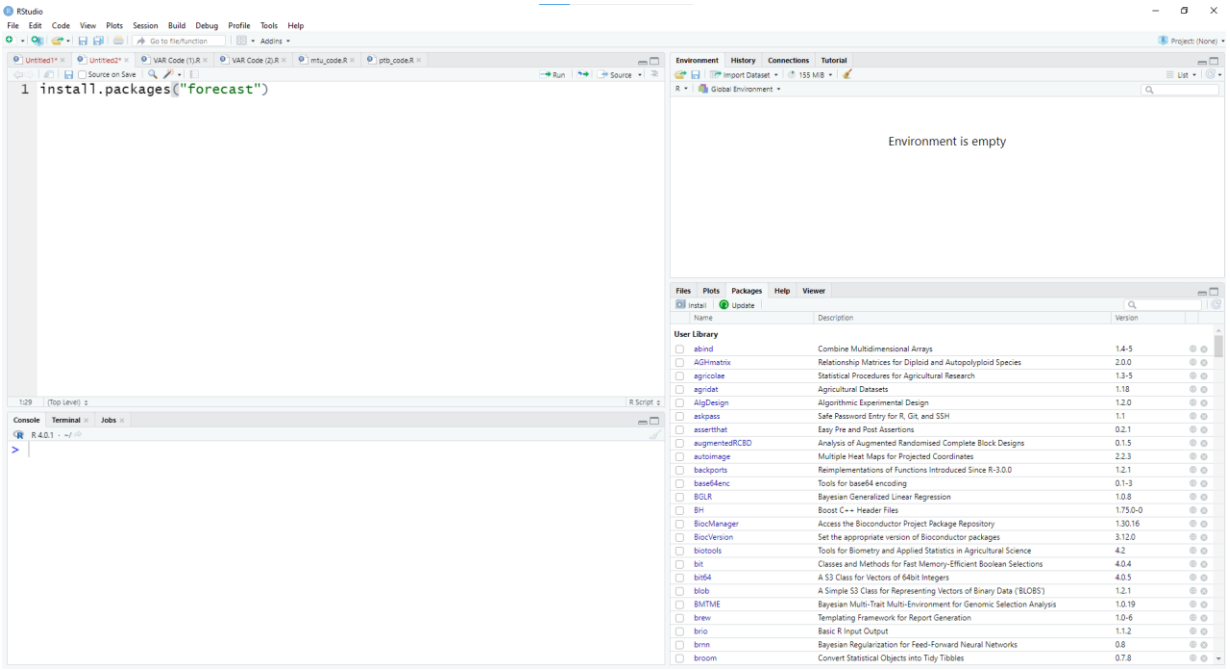


Step 3: Now restart R, and verify that make can be found, which should show the path to your Rtools installation.

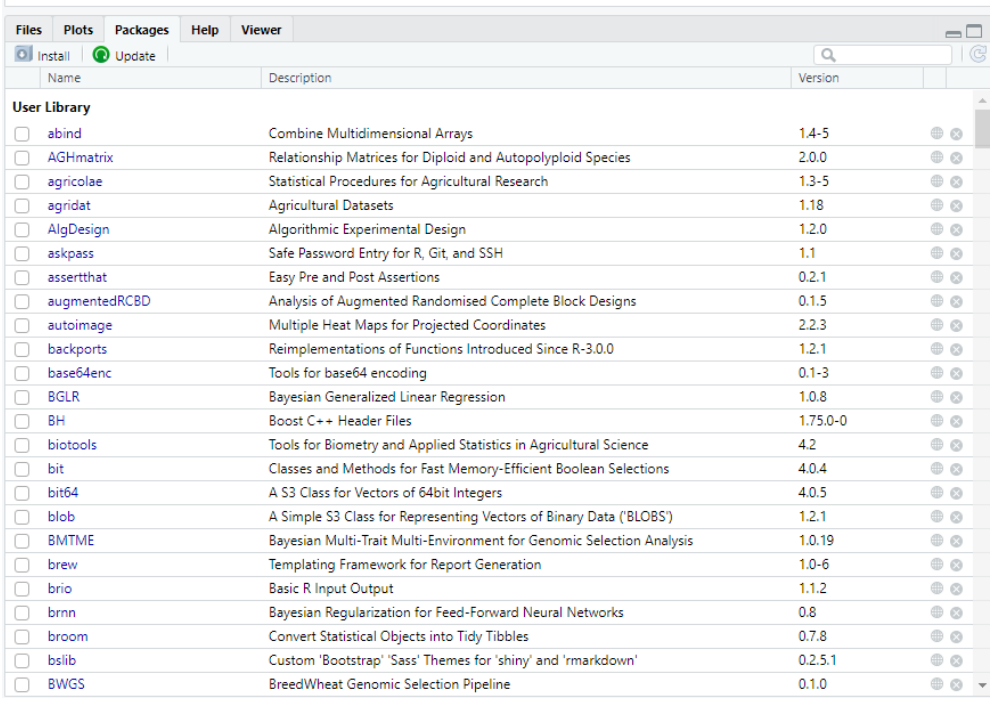
Sys.which("make")

Installation of R Packages

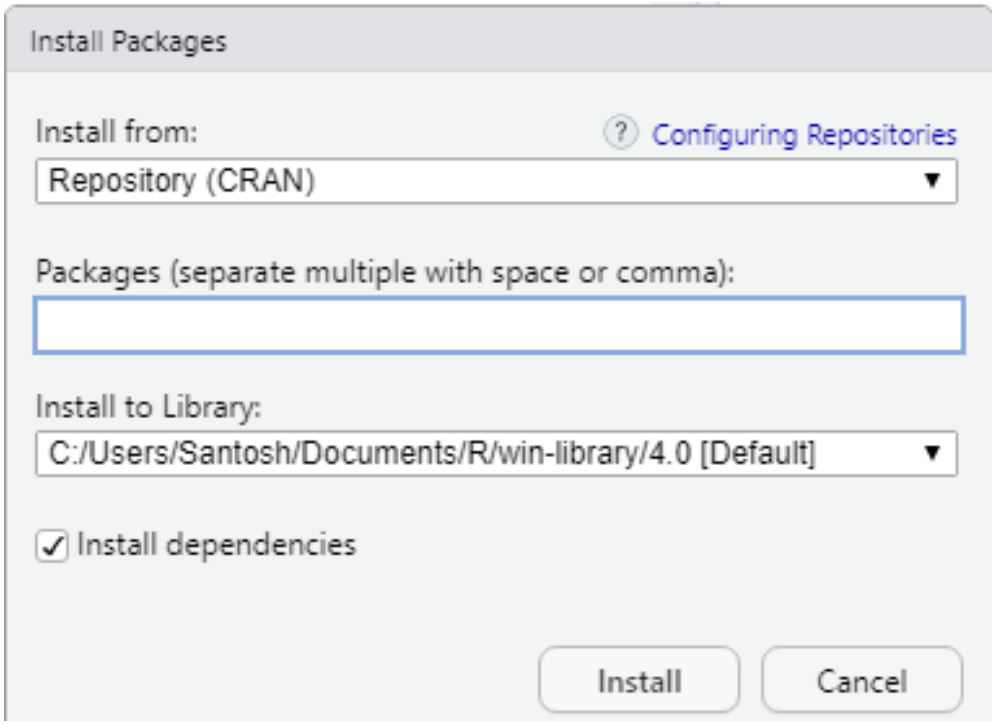
install.packages("forecast", type = "source")



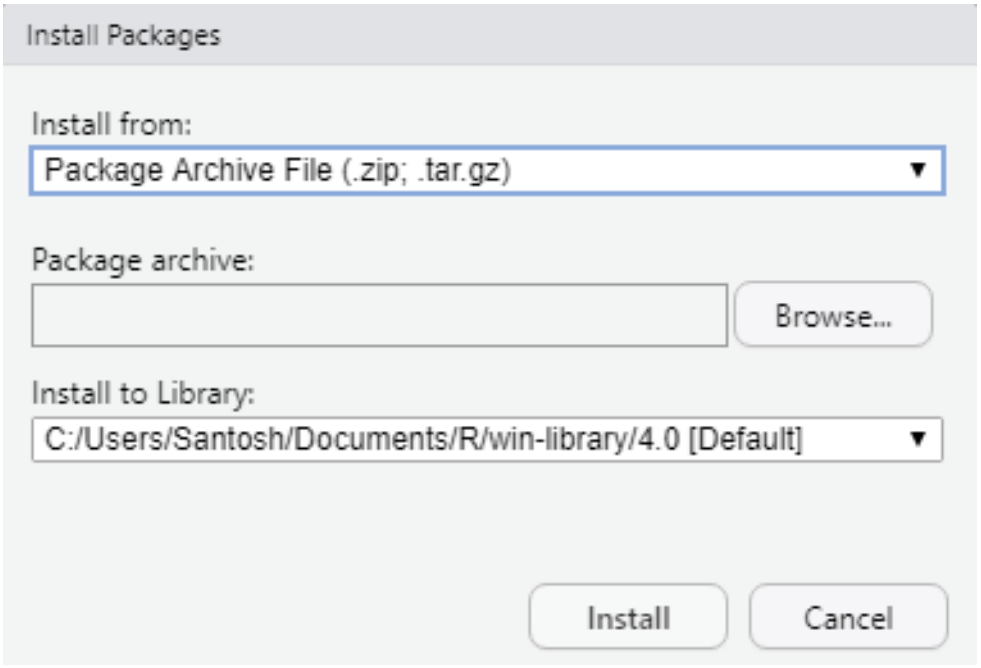
Installation of R packages from R window



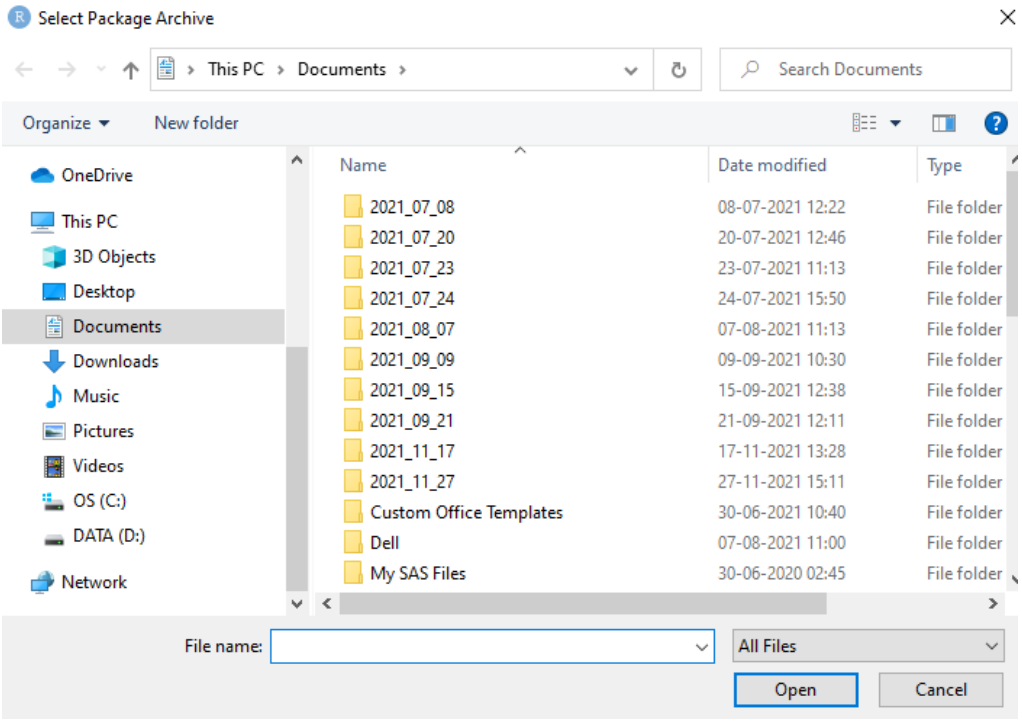
Installation or update from CRAN



Installation from local files



Select the folder containing R package



Install packages from website

install.packages("plantbreeding", repos="http://R-Forge.R-project.org")

Install packages from Bioconductor

To install core packages, type the following in an R command window:

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install()
```

Install specific packages, e.g., “GenomicFeatures” and “AnnotationDbi”, with

```
BiocManager::install(c("GenomicFeatures", "AnnotationDbi"))
```

To check the available Bioconductor packages

```
BiocManager::available()
```

Suggested Readings

1. https://cran.r-project.org/bin/windows/base/
2. https://www.rstudio.com/products/rstudio/download/
3. https://cran.r-project.org/bin/windows/Rtools/rtools40.html
4. https://www.bioconductor.org/install/

Introduction to R

Santosha Rathod, Nobin Chandra Paul, Ponnaganti Navyasree, K. Ravi Kumar and Prabhat Kumar

ICAR-National Institute of Abiotic Stress Management, Baramati, Pune-413115
Santosha.Rathod@icar.org.in

Preliminaries:

R is a comprehensive software tool utilised for the purpose of data manipulation, calculation, and graphical representation of information. It possesses several key capabilities: i) efficient data management and storage facilities, ii) seamless handling of computations involving arrays and matrices, iii) integration with various tools and databases for thorough data analysis, iv) provision of graphical outputs in multiple file formats, and v) a well-established, straightforward, and effective programming language ('S') that encompasses conditionals, loops, user-defined recursive functions, and input/output functionalities R has grown quickly and has been enhanced by many researchers and developers who offer a diverse collection of statistical computation methods through various packages. However, most programs written in R tend to be ephemeral, tailored for specific data analyses, and may not be suitable for datasets available in diverse formats.

Developmental history:

R is a robust programming language designed for statistical data analysis, developed as an extension of the S language at Bell Laboratories by Rick Becker, John Chambers, and Allan Wilks, and it acts as the foundation for S-Plus systems. The S language was presented in a set of four books authored by John Chambers and his collaborators. The methodologies of R are primarily based on "The New S Language: A Programming Environment for Data Analysis and Graphics" by Becker, Chambers, and Wilks. Currently, there are numerous books available to learn R, catering to both novice and advanced users. The R software was created in 1993 by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and it is presently maintained by the R Core Team. The name `R` is derived from the initial letters of its creators and signifies its roots in `S`. R has emerged as a popular tool for statistical analysis, is free and open-source, and can perform various statistical analyses, including modern methodologies. Additionally, it can be effortlessly expanded through contributions from statisticians and researchers worldwide in the form of packages.

R and statistics:

R is extensively utilized by statistician and data analysts for creating statistical software and conducting data analysis. According to survey, data mining studies, and analysis of academic literature, R has gained significant popularity, ranking 14th on the TIOBE index as of August 2021, which assesses the popularity of programming languages. While many documents about R do not specifically refer to statistics, it is commonly used as a software tool for statistical data analysis. The base R package includes numerous fundamental statistical methods, encompassing both descriptive and inferential statistics. In total, R comes with about 25 standard and recommended packages, and a wealth of additional packages can be found on CRAN (https://CRAN.R-project.org).

R and Operating System:

The majority of users are utilizing R on the Windows operating system, along with Linux and macOS. Distinct versions and packages are offered for every operating system individually. The latest version of R for Windows OS is 4.5.1.

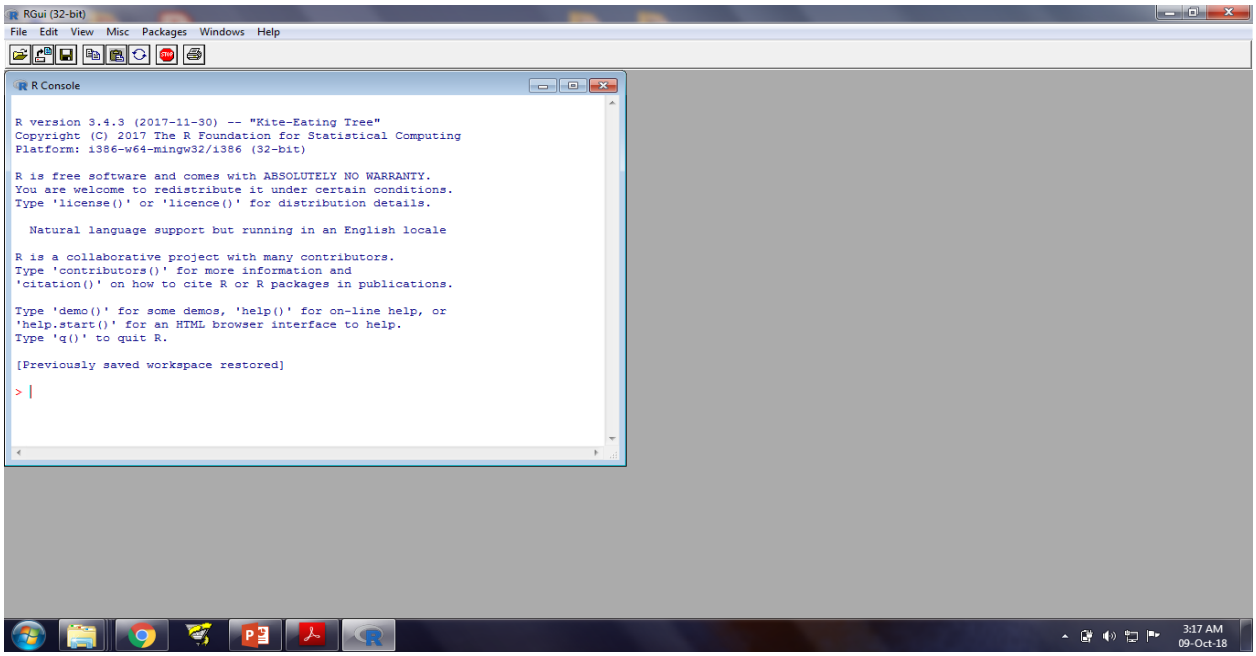
Getting and Installing R:

R can be downloaded for free from any of the mirror sites of the Comprehensive R Archive Network (CRAN) http://cran.r-project.org/. It is suggested to choose a mirror that is geographically closer to you.

Steps for installing R:

- R can be installed on Windows, Macintosh, and Unix platforms.
- To set up R on your machine, double-click the downloaded file R.exe and choose `English` as the language.
- A setup wizard for R will open.
- Click on `Next` and accept most of the default options throughout the installation process.
- As of July 11, 2025, the latest version of R available for Windows is 4.5.1.

R Console Window



Starting R

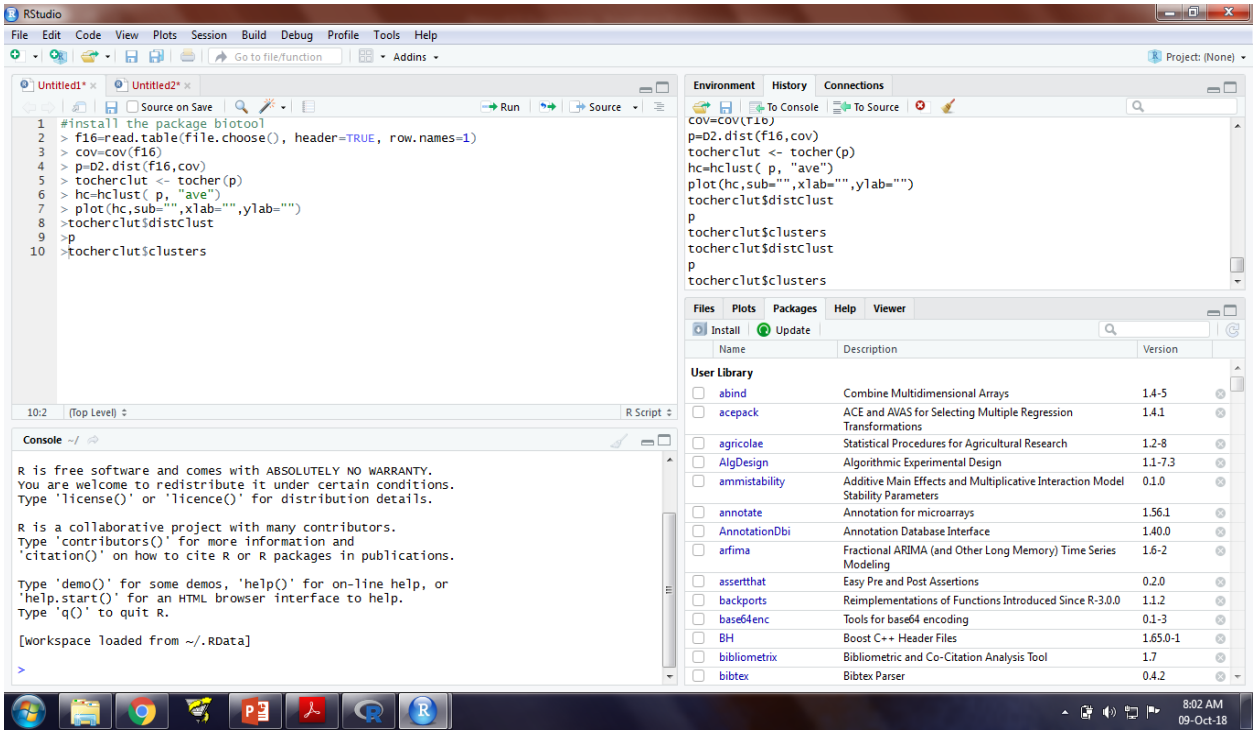
- To launch R, navigate to the start menu, select all programs, then R, and finally R 4.5.1, which will bring up a screen similar to the one shown above.
- The blank white screen is referred to as the R Console, where all R code is entered and results are displayed.
- At the top of the Console, you will find a toolbar along with several menus in the R window.
- To understand the functions of each button on the toolbar, hover your mouse over a button for a moment, and a description will pop up.
- There is an R editor available that can be used to compose and modify R code.
- The R editor window operates like a text editor, providing options for selecting, cutting, copying, pasting, typing, and deleting text, among other capabilities.
- You can open the R editor window by clicking on File, then selecting New Script from the menu bar.

- To execute the code written in the R editor window, it must be transferred to the R console by clicking the Run line or selection button found on the toolbar in the R editor window.

RStudio:

RStudio is an integrated development environment (IDE) designed for R. It features a console, an editor with syntax highlighting that allows for direct execution of code, along with utilities for plotting, history tracking, debugging, and managing the workspace. The latest version of RStudio available for Windows operating systems has a size of 281.24 MB | SHA-256: 3a553330 | Version: 2025.05.1+513 | Released: 2025-06-05. RStudio is offered in two versions:

- RStudio Desktop, which operates locally like a standard desktop application and is available for Microsoft Windows, Mac OS X, and Linux.
- RStudio Server, enabling users to access RStudio through a web browser while it runs on a remote Linux server.
- RStudio comes in both open source and commercial editions and can be used on desktops (Windows, Mac, and Linux) or via a browser connected to RStudio Server or RStudio Server Pro (Debian/Ubuntu, RedHat/CentOS, and SUSE Linux).
- The free version of RStudio Desktop can be downloaded from https://www.rstudio.com/.
- RStudio offers a more user-friendly interface for utilizing R.



The top left window in the figure displays the editor window for writing R scripts.

• The bottom left window resembles the R console where R scripts are executed.

• The top right window of RStudio contains two tabs: Environments and History.

• In the Environment tab, we can observe which objects are currently loaded in R.

• The History tab provides a record of commands that have been executed.

• The bottom right window of RStudio features several tabs, including Files, Plots, Packages, Help, and Viewer.

• The Files tab is utilized to navigate through different files.

• The Plots tab is for visualizing plots generated by R scripts.

• The Packages tab lists the packages that are already installed and facilitates the installation and loading of packages within a session.

• The Help tab can be accessed for assistance on R functions.

• The Viewer tab is used for displaying local web content.

Using R

To begin using R: Access R by going to the start menu → all programs → R → R 4.5.1, which opens the R console screen.

R commands:

• In the Console, you will see a > symbol.

• Type commands following this symbol, and then press the Enter key.

• After entering a command in the Console and hitting Enter, R processes the commands and provides output or an error message in the Console.

For instance, if you enter

>1+9

[1] 10

• In this situation, R calculated 1 and 9, yielding the result 10.

• Next, try typing

> 2+*6

Error: unexpected `*' in "2+*"

• This time, R has produced an error message because `+*' is not a recognized operator in R.

• It's important to know the correct syntax to use in the Console, or else it will consistently display an error message.

• R operates interactively, returning results to commands one at a time

> x=2

> x*15

[1] 30

• R primarily functions as an expression language and follows a specific syntax.

Using R: Common Rules of R Commands

R commands are sensitive to case, meaning that AB, Ab, aB, and ab are treated as distinct items.

• Symbols can include any alphanumeric characters, `.' and ` ' , and in certain countries, accented letters may also be used.

• An R name must begin with a letter or `.' and if it starts with `.', the second character cannot be a number. There is no limit to the length of names.

• Commands can be separated by a semi-colon (`;') or by a newline.

Curly braces `f' and `g' are utilized to define a block of code.

• Any line that begins with a # is treated as a comment and will not be executed, and comments can be inserted anywhere.

• If a command is not fully written out by the end of a line, R will present a different prompt, typically `+' on the following lines, and will continue to accept input until the command is complete in syntax.

• The length of a command entered at the Console is limited to approximately 4095 bytes.

Working Directory:

The working directory refers to the folder in which R is currently functioning.

• Typically, this directory defaults to My Documents or Documents.

• You can determine the present working directory by executing the command > getwd() [1] "C:/Users/User/Documents".

• R has the ability to directly read and access files from the working directory without the need to specify any path. In the same way, it can also save and write files directly within the working directory.

• To alter the working directory to another folder, you can use > setwd("C:/Users/Rathod/Documents/Rtraining/").

• It is recommended to set the working directory at the beginning of an R session to a folder that holds most of your data files and scripts.

> getwd()

[1] "C:/Users/Rathod/Documents/Rtraining".

Data Types in R:

R is a language that follows the principles of object-oriented programming. Objects can include variables, vectors, arrays, character strings, functions, or more complex structures made from these elements. Objects are created and assigned names for storage purposes.

```
> s <- list(name = "Paul", age = 22, GPA = 3.9)
> objects(s)
[1] "age"  "GPA"  "name“
 > s
$name
[1] "Paul"
$age
[1] 22
```

```
$GPA
[1] 3.9
```

You can eliminate the objects by using the command

```
>rm(s)
> s
Error: object 's' is not present
>rm(list=ls())
```

To clear the console, use

Ctrl+l

Vector:

A vector is a collection of elements. The command

>rath=c(2,4,52,99,26) creates a vector containing five numbers. In this case, the object named rath includes these numbers, and the function c() is utilized to assign those numbers to rath.

Vectors can be classified into three categories: i) numeric, ii) character, and iii) logical. A numeric vector consists of numbers, a character vector consists of characters, and a logical vector contains values such as TRUE, FALSE, or NA. For example,

a <- (2,4,7,-3,-1,-5,9) represents a numeric vector,

b<-c("Sam", "Paul", "John") represents a character vector, and

c <- c(TRUE, TRUE, FALSE, TRUE, TRUE, FALSE) represents a logical vector.

Matrices:

A matrix is a collection of elements organized in at least two dimensions. These elements can be of numeric, character, or logical types.

```
> x=matrix(c("3","4","5","6"),nrow=2)
> x
     [,1] [,2]
[1,] "1" "3"
 [2,] "2" "4"
```

Arrays:

Arrays are generalizations of vectors and matrices across multiple dimensions. A two-dimensional array is called a matrix, and arrays can have additional dimensions.

```
> vector1 <- c(3,4,6)
> vector2 <- c(16,17,12,13,90,15)
> a <- array(c(vector1,vector2),dim = c(3,3))
> a
     [,1] [,2] [,3]
[1,]    3   16   13
[2,]    4   17   90
```

```
[3,]    6   12   15
```

Factors:

Factor objects serve to define categorical, classificatory, or grouping variables. For instance, males and females represent two levels of the gender variable. Thus, gender can be regarded as a factor object.

```
 > gender=c("F", "F", "M","F")
 > gender=as.factor(gender)
 > levels(gender)
 [1] "F" "M"
                    >department = c("HR", "Finance", "IT", "HR", "IT", "Finance")
                    >department = as.factor(department)
                    >levels(department)
                    [1] "Finance" "HR" "IT"
```

Lists:

A list is a collection of items of different types, such as a vector, a matrix, or a data frame.

```
 > mylist=list(x=c(80,90,100),y=matrix(2:8,nrow=3))
 > mylist
$x
[1] 80 90 100
$y
      [,1] [,2]
[1,]    2    6
[2,]    3    7
[3,]    4    8
```

Data Frames:

A data frame is a two-dimensional structure with columns that can have different types, such as numeric, character, or factor. In this case, a column represents a variable.

```
>height = c(160, 165, 170, 175, 168)
>score  = c(85, 88, 90, 92, 87)
>student_data = data.frame(height, score)
>student_data
  height score
1    160    85
2    165    88
```

```
3   170   90
4   175   92
5   168   87
```

The data.frame() function is utilized to construct a data frame.

Functions:

Functions in R, are a kind of objects which takes one or more inputs and produces some results as output. R has a number of in-built functions. R also provides facility to create new functions by users.

Calculate the mean and variance of y

```
# Create a numeric vector
>y <- c(7, 12, 15, 22, 28, 19, 31, 14, 10, 18, 25, 27)
>mean(y) # Calculate the mean
  [1] 18.66667
>var(y)
[1] 61.24242
```

A comprehensive list of built-in functions is available in the R reference manual. Additionally, many external functions can be found in contributed packages. To explore a function's arguments and usage, use help(function name), which will open a webpage containing relevant details. For example, invoking help(aov) will provide information on the usage of the aov() function.

Create your own function:

```
myfunction <- function(argument1, argument2, ...) {
  # statements or code go here
  result <- argument1 + argument2  # example operation
  return(result)
}
```

Example 1:

```
sub<- function(a, b) {
   a-b
  }
```

Example 2:

```
twosam <-function(y1, y2){
n1 <- length(y1)
n2 <- length(y2)
yb1 <- mean(y1)
yb2 <- mean(y2)
s1 <- var(y1)
s2 <- var(y2)
```

```
df<-n1+n2-2
s <- ((n1-1)*s1+(n2-1)*s2)/(df)
 tst <- (yb1-yb2)/sqrt(s*((1/n1)+(1/n2)))
 return(tst)
}
```

R Packages

While most standard statistical analyses can be performed using base R, there are times when specific analyses require additional R packages. As of October 11, 2018, there are 13043 packages accessible on CRAN for various types of analyses, and this number continues to grow daily.

```
> nrow(available.packages()) # as of July 11, 2025

 [1] 22347
```

Downloading and installing a package

To utilize an R package, first download it from CRAN, then install and load it during an R session. You can download the package either within R or externally. To do this within R, go to the Packages menu, choose Install packages(s), select a mirror from the list provided, and then pick the packages you want.

```
>library(packagename)
 >library(agricolae) # package for ANOVA / DOE
>library(plantbreeding) ) # package for ANOVA / DOE
>library(biotool) ) # package for D-square analysis
```

Reading data in R

Loading Data in R directly:

```
>day <- c('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun')  # Define the vectors

>visitors <- c(120, 135, 150, 160, 145, 170, 200) # Combine into a data frame

>library_data <- data.frame(day, visitors)

>library_data

[1] day   visitors

1  Mon      120

2  Tue       135

3  Wed      150

4  Thu      160

5  Fri       145

6  Sat       170
```

7 Sun 200

names(library_data)

[1] "day" "visitors"

Loading data in R from an external file:

Data is often extensive, with many variables and observations. It may be stored in a spreadsheet, an external file, statistical software, or a webpage. R provides tools to import data from these sources.

Reading data from text file:

Data in text format should have individual observations separated by delimiters like `,`, `:`, `t`, space, `~`, `@`, `&`, blank space, or `*`. Any observations or variables containing these characters can cause errors during data loading.

```
>mydata2=read.table("rainfall.txt",header=TRUE, sep=",")

>rainfall=read.table(file.choose(), header=TRUE, row.names=1)
```

The first parameter of the read.table() function indicates the external file being referenced. The second parameter, header = TRUE, informs R that the rainfall.txt file contains a header, which will be used as variable names for the data. If the file lacks a header, header = FALSE should be specified. The third parameter, sep = ",", indicates that the data entries are separated by commas. While there are additional parameters available for the read.table() function, these three are essential.

Reading data from .csv files:

The read.csv() function imports comma-separated value (CSV) files, which are files that contain observations separated by commas.

The read.csv2() function imports data from files that are delimited by semicolons (`;`).

The functions read.delim() and read.delim2() are used to load data from files that are tab delimited.

Reading data from a webpage:

To retrieve a dataset from a webpage, you can utilize the read.table() function along with the full URL of the page.

>country_data <- read.table("https://people.sc.fsu.edu/~jburkardt/data/csv/hw_200.csv", header = TRUE, sep = ",")

>country_data

[1] Index Height(Inches) Weight(Pounds)

1 1 65 112

2 2 67 120

3 3 68 130

4 4 69 135

Loading data from a spreadsheet:

To read data from Excel directly in R, you need to use a package named xlsx.

>library(xlsx)

>rain_data <- read.xlsx("ranfallsheet.xlsx", sheetName = "rain")

>rain_summary <- summary(rain_data)

>write.csv(as.data.frame(rain_summary), file = "rain_summary.csv")

>write.csv(as.data.frame(rain_data$rainfall), file = "rainfall_column.csv")

Suggested Readings

- An Introduction to R Notes on R: A Programming Environment for Data Analysis and Graphics. Version 4.1.2 (2021-11-01)
- https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf
- Niel J Le Roux and Sugnet Lubbe. A step by step R tutorial: An introduction into R applications and programming, 2015, 1st edition, Bookboon.com
- Dalgaard, Peter. Introductory Statistics with R, 2008, 2nd edition, springer.
- W. N. Venables, D. M. Smith and the R Core Team: An Introduction to R Notes on R: A Programming Environment for Data Analysis and Graphics. Version 3.2.3 (2015-12-10).
- R Documentation.
- R help manual

R Tidyverse for Data Manipulation

Santosha Rathod, Nobin Chandra Paul, Ponnaganti Navyasree, K. Ravi Kumar and Prabhat Kumar

ICAR-National Institute of Abiotic Stress Management, Baramati, Pune-413115
Santosha.Rathod@icar.org.in

According to the tidyverse website, https://www.tidyverse.org/, “The tidyverse is a curated set of R packages intended for data science. All the packages utilize a shared fundamental design philosophy, grammar, and data structures.” Tidyverse was created by Hadley Wickham, who serves as the chief scientist at RStudio. The principles and functions of the tidyverse are also evident in other packages, ensuring compatibility within the collection. Understanding the tidyverse can be broken down into four key aspects: i) it consists of a suite of R packages - when we install the tidyverse package, we are also installing several other packages included in the tidyverse. Notable packages that are part of the tidyverse include “dplyr, tidyr, ggplot2, tibble, readr, stringr, purrr, and forcats,” with dplyr, tidyr, and ggplot2 being the most widely used; ii) they all share a common philosophy and application programming interfaces. This harmonious feature of the tidyverse eliminates the need to adjust your data for different packages, as all tidyverse packages work with the same datasets, making it a highly popular and universally accepted tool for data manipulation; iii) it is geared toward data science – while tidyverse does not handle extensive statistical modeling, it aids in completing much of the preparatory work that other tools may not tackle, thus expediting the statistical modeling process. In this respect, tidyverse is particularly suited for data science; iv) they are opinionated - the developers are receptive to any potential inclusions or modifications to the tidyverse.

Similar to other packages, tidyverse can be installed with the command install.packages("tidyverse"). Once the installation is complete, we need to load the library using library(tidyverse). The essential tidyverse comprises all the packages that you will probably utilize in any data analysis. The core tidyverse contains several packages that are categorized into four groups.

1. Visualization and exploration of data: ggplot2

2. Data manipulation and restructuring: dplyr, tidyr, stringr, forcats

3. Data importing and organization: tibble, reader

4. Functional programming: purrr

ggplot2 is a widely used toolkit for data visualization. In contrast to other visualization tools, ggplot2 offers extensive customization options for your plots, making it ideal for tailoring visualizations to meet specific goals. dplyr assists with data manipulation by enabling operations such as adding or removing columns, filtering and subsetting, as well as summarizing data (like calculating averages for groups within your dataset). tidyr aids in transforming your data from a "wide" format to a "tall" format, similar to the reshape and reshape2 packages. The tibble package helps maintain the data frame's structure, unlike the traditional data.frame function. readxl enables you to import Excel files directly into R without needing to convert them to .csv format first. forcats provides several functions to manage categorical data effectively. stringr simplifies the handling of text within your dataset. Purrr facilitates functional programming by utilizing the map function..

1. Data visualization and exploration:

Data visualization involves converting extensive data sets and metrics into charts, graphs, and other visual formats. Data exploration serves as the preliminary phase in data analysis during which analysts examine the data.

ggplot2: gplot2 is an open-source data visualization package created by Hadley Wickham in 2005, inspired by Leland Wilkinson's Grammar of Graphics, which serves as a comprehensive framework for visualizing data by breaking down graphs into key components like scales and layers. This plotting package enables users to generate intricate plots from unorganized data contained within a data frame. It offers a more programmatic way to define which variables to graph, their visual representation, and overall design features. You can install ggplot2 by using install.packages("tidyverse") or by directly running install.packages("ggplot2").

Components of ggplot2

- **Data:** The dataset itself is referred to as the element and should be kept in an R data frame.

- **Layer:** Layers consist of geometric objects that represent the plot through the use of points for short data.

- **Scales:** Scales are utilized to translate values from the dataset accordingly.

- **Coordinate system:** This is the space that connects data to display using Cartesian coordinates, fixed settings, polar coordinates, and limits.

- **Faceting:** The faceting technique divides a plot into a grid of panels, each displaying a different subset of the data.

- **Theme:** It manages the more detailed aspects of the display, such as font size, background, and color attributes**.**

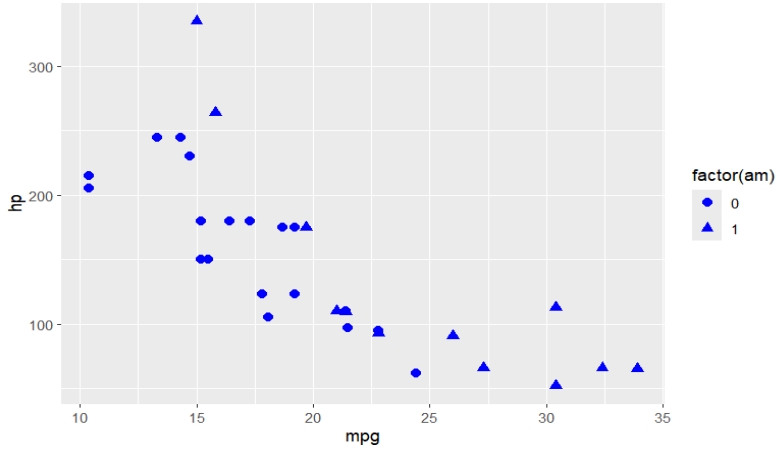
- **Geometrics:** This aspect illustrates how the data is presented using points, lines, charts, histograms, bars, etc.

- **Aesthetics:** Aesthetics pertains not to the visual appearance, but to which variable is assigned to it.

Example:

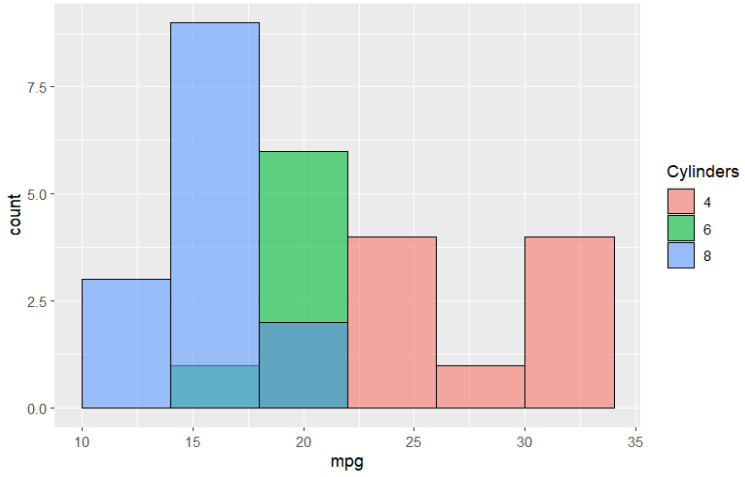
install.packages("tidyverse"), library("tidyverse")

ggplot(mtcars, aes(x = mpg, y = hp, shape = factor(am))) + geom_point(color = "blue",size=2.5)

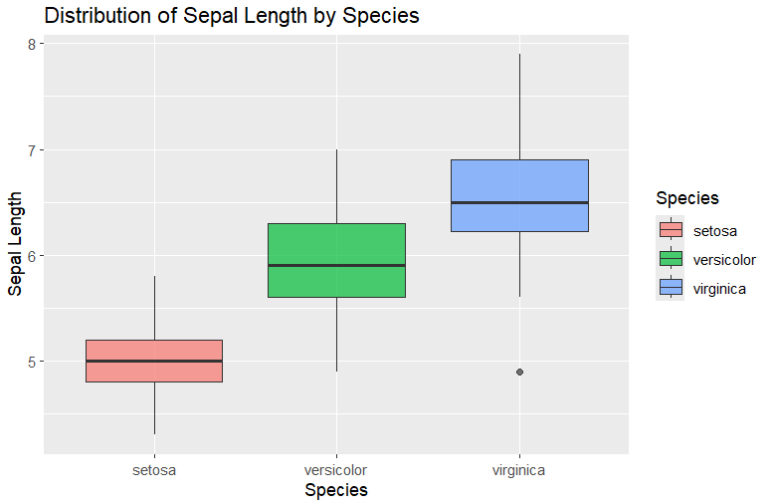


ggplot(mtcars, aes(x = mpg, fill = factor(cyl))) +

geom_histogram(binwidth = 4, position = "identity", alpha = 0.6, color = "black") + labs(fill = "Cylinders")




```
ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot(alpha = 0.7) +
  ylab("Sepal Length") +
  ggtitle("Distribution of Sepal Length by Species")
```



2. Data wrangling and transformation: dplyr, tidyr, stringr, forcats

Data wrangling involves organizing and standardizing complex data sets for easier analysis, while transformation converts raw data into a clean format.

dplyr package: This package facilitates data manipulation, making it more efficient for statistical data analysis. It reduces the time needed for data organization compared to traditional approaches. You can install the package either via tidyverse or directly using

install.packages("tidyverse"). The dplyr package offers several essential functions that can be utilized on a data frame.They are

Filter() function: The filter function allows you to select specific columns of a data frame for focus.

Example:

```
>library(dplyr)
>data(iris)
>ir = tbl_df(iris)
>ir
filter(ir, Sepal.Length > 5)
# A tibble: 118 × 5
[1]   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
        <dbl>       <dbl>        <dbl>       <dbl> <fct>
1         5.1         3.5          1.4           0.2 setosa
2         5.4         3.9          1.7           0.4 setosa
3         5.4         3.7          1.5           0.2 setosa
4         5.8         4           1.2            0.2 setosa
5         5.7         4.4          1.5           0.4 setosa
6         5.1         3.8          1.5           0.3 setosa
7         5.4         3.4          1.7           0.2 setosa
8         5.1         3.7          1.5           0.4 setosa
9         5.7         3.8          1.7           0.3 setosa
10          6.4         2.9          4.3           1.3 versicolor
# ... with 108 more rows
```

Select () function: The select() function allows you to choose specific columns from a data frame that you want to pay attention to.

```
> # Using previous ir = tbl_df(iris)
>ir %>% select(Sepal.Width, Species)
# A tibble: 150 × 2
 [1] Sepal.Width Species
        <dbl> <fct>
 1         3.5 setosa
 2         3.0 setosa
 3         3.2 setosa
 4         3.1 setosa
 5         3.6 setosa
 6         3.9 setosa
 7         3.4 setosa
 8         3.4 setosa
 9         2.9 setosa
10          3.1 setosa
# ... with 140 more rows
```

```
> select(ir, -Sepal.Width) # Select all columns except Sepal.Width
# A tibble: 150 × 4
[1]   Sepal.Length Petal.Length Petal.Width Species
        <dbl>        <dbl>       <dbl> <fct>
 1        5.1          1.4         0.2 setosa
 2        4.9          1.4         0.2 setosa
 3        4.7          1.3         0.2 setosa
 4        4.6          1.5         0.2 setosa
 5        5.0          1.4         0.2 setosa
 6        5.4          1.7         0.4 setosa
 7        4.6          1.4         0.3 setosa
 8        5.0          1.5         0.2 setosa
 9        4.4          1.4         0.2 setosa
10        4.9          1.5         0.1 setosa
# ... with 140 more rows
```

To bring the Sepal.Width column to the front, simply enter the name of that column followed by everything():

```
>ir %>% select(Sepal.Width, everything())
# A tibble: 150 × 5
 [1] Sepal.Width Sepal.Length Petal.Length Petal.Width Species
        <dbl>        <dbl>              <dbl>       <dbl> <fct>
 1        3.5          5.1                1.4         0.2 setosa
 2        3           4.9                 1.4         0.2 setosa
 3        3.2          4.7                1.3         0.2 setosa
 4        3.1          4.6                1.5         0.2 setosa
 5        3.6          5                  1.4         0.2 setosa
...
```

Summarise () function: Is used to get the summary statistics in tables like mean, median,.

```
>library(dplyr)
>data(iris)
>summarise(iris,
          mean_sepal = mean(Sepal.Length, na.rm = TRUE),
          sd_sepal = sd(Sepal.Length, na.rm = TRUE))
[1]  mean_sepal  sd_sepal
1   5.843333 0.8280661
```

Arrange () function: The arrange function is utilized to reorder the rows of a data frame based on the values of one of its variable columns.

#Sort the dataframe in R with the arrange function.

>arrange(ir, Sepal.Width)

A tibble: 150 × 5

```
 [1] Sepal.Length Sepal.Width Petal.Length Petal.Width Species
        <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1        5.5         2.0          4.0         1.3 versicolor
 2        5.6         2.2          4.5         1.5 versicolor
 3        5.5         2.3          4.0         1.3 versicolor
 4        6.1         2.4          5.6         1.4 virginica
 5        5.6         2.4          3.9         1.1 versicolor
 ...      ...         ...          ...         ... ...
```

 # Arrange by Petal.Length in ascending order
arrange(ir, Petal.Length)
# A tibble: 150 × 5

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
        <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1        5.1         3.5          1.0         0.2 setosa
 2        4.6         3.6          1.0         0.2 setosa
 3        4.3         3.0          1.1         0.1 setosa
 4        5.0         3.5          1.2         0.2 setosa
 5        4.7         3.2          1.3         0.2 setosa
...
```

Mutate () function: mutate offers a straightforward interface for generating new variables that are based on existing ones.

 # Create a new column Sepal.Ratio using mutate
> ir2 <- mutate(ir, Sepal.Ratio = Sepal.Length / Sepal.Width) *# Display the first few rows*
>head(ir2)
A tibble: 6 × 6

```
[1]Sepal.Length Sepal.Width Petal.Length Petal.Width Species Sepal.Ratio
        <dbl>       <dbl>        <dbl>       <dbl> <fct>         <dbl>
1         5.1         3.5          1.4         0.2 setosa         1.46
2         4.9         3            1.4         0.2 setosa         1.63
3         4.7         3.2          1.3         0.2 setosa         1.47
4         4.6         3.1          1.5         0.2 setosa         1.48
5         5.0         3.6          1.4         0.2 setosa         1.39
6         5.4         3.9          1.7         0.4 setosa         1.38
```

 # Add Large.Sepal column based on Sepal.Length
>ir2 <- mutate(ir, Large.Sepal = Sepal.Length > 6)
>head(ir2)

```
# A tibble: 6 × 6
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Large.Sepal
         <dbl>       <dbl>        <dbl>       <dbl> <fct>   <lgl>
1          5.1         3.5          1.4           0.2 setosa  FALSE
2          4.9         3.0          1.4           0.2 setosa  FALSE
3          4.7         3.2          1.3           0.2 setosa  FALSE
4          4.6         3.1          1.5           0.2 setosa  FALSE
5          5.0         3.6          1.            0.2 setosa  FALSE
6          5.4         3.9          1.7           0.4 setosa  FALSE
```

rename() function: To rename a column in tidyverse, utilize the rename function. Within this function, you specify the new name (without quotation marks), followed by an equals sign, and then the old name enclosed in quotes

```
library(dplyr)
iris %>%rename(
    X1 = "Sepal.Length",
    X2 = "Sepal.Width",
    X3 = "Petal.Length",
    X4 = "Petal.Width"
  )
# A tibble: 150 × 5
     X1    X2    X3    X4 Species
  <dbl> <dbl> <dbl> <dbl> <fct>
1   5.1   3.5   1.4   0.2 setosa
2   4.9   3.0   1.4   0.2 setosa
3   4.7   3.2   1.3   0.2 setosa
4   4.6   3.1   1.5   0.2 setosa
5   5.0   3.6   1.4   0.2 setosa
6   5.4   3.9   1.7   0.4 setosa
7   4.6   3.4   1.4   0.3 setosa
8   5.0   3.4   1.5   0.2 setosa
9   4.4   2.9   1.4   0.2 setosa
10  4.9   3.1   1.5   0.1 setosa
# … with 140 more rows
```

Group _by () function: The group_by() function is utilized to create summary statistics from the data frame, with strata determined by a specific variable.

%%() piping function: the pipeline operator %% is very handy for stringing together multiple dplyr function in a sequence of operations.

```
iris_grouped <- iris %>%
    mutate(Type = ifelse(Sepal.Length > 5.5, "Long", "Short")) %>%
```

```
    group_by(Type)
iris_grouped %>%
     as_tibble() %>%
    slice(1:10)
# A tibble: 10 × 6
# Groups:   Type [1]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species Type
          <dbl>       <dbl>        <dbl>       <dbl> <fct>   <chr>
 1          5.1         3.5          1.4         0.2 setosa  Short
 2          4.9         3            1.4         0.2 setosa  Short
 3          4.7         3.2          1.3         0.2 setosa  Short
 4          4.6         3.1          1.5         0.2 setosa  Short
 5          5           3.6          1.4         0.2 setosa  Short
 6          5.4         3.9          1.7         0.4 setosa  Short
 7          4.6         3.4          1.4         0.3 setosa  Short
 8          5           3.4          1.5         0.2 setosa  Short
 9          4.4         2.9          1.4         0.2 setosa  Short
10          4.9         3.1          1.5         0.1 setosa  Short
```

Merging

To combine two data frames in R, you can use either the merge() function from base R or dplyr's join functions, which are faster for larger datasets. The dplyr package offers advantages such as improved speed, clarity about merging keys, and flexibility with database tables.

dplyr joins:

left_join(): Combine two datasets while retaining all entries from the source table.
right_join(): Combine two datasets while retaining all entries from the target table.
inner_join(): Combine two datasets while omitting any unmatched rows.
full_join(): Combine two datasets while retaining all entries.

```
> ## function for merging
```

```
> ## establishing a database # tribble for generating small data tables
>df_iris1 <- tribble( ~ID, ~Sepal.Length,
                       "A",       4.6,
                       "B",       5.1,
                       "C",       6.7,
                       "D",       5.8,
                       "E",        7.0 )
>df_iris2 <- tribble( ~ID,  ~Species,
                        "A", "setosa",
                        "B", "setosa",
                        "C", "versicolor",
                        "F", "virginica",
                        "G", "virginica" )
>left_join(df_iris1, df_iris2, by = "ID")
# A tibble: 5 × 3
[1] ID    Sepal.Length Species
  <chr>       <dbl> <chr>
1 A             4.6 setosa
2 B             5.1 setosa
3 C             6.7 versicolor
4 D             5.8 NA
5 E             7   NA
>right_join(df_iris1, df_iris2, by = "ID")
# A tibble: 5 × 3
  ID    Sepal.Length Species
  <chr>       <dbl> <chr>
1 A             4.6   setosa
2 B             5.1   setosa
3 C             6.7   versicolor
4 F            NA     virginica
5 G            NA     virginica
>inner_join(df_iris1, df_iris2, by = "ID")
```

```
# A tibble: 3 × 3
[1] ID    Sepal.Length Species
  <chr>       <dbl> <chr>
1 A             4.6 setosa
2 B             5.1 setosa
3 C             6.7 versicolor
>full_join(df_iris1, df_iris2, by = "ID")
 # A tibble: 7 × 3
 [1]ID    Sepal.Length Species
  <chr>       <dbl> <chr>
1 A             4.6    setosa
2 B             5.1    setosa
3 C             6.7    versicolor
4 D             5.8     NA
5 E             7       NA
6 F             NA     virginica
7 G             NA     virginica
```

Tidyr: One of the key packages in R is the tidyr package. Tidyr is utilized for constructing tidy data. Tidy data refers to a dataset that contains various essential functions for data cleaning. It can be installed using the function install.packages("tidyverse"). Tidyr assists in transforming data from "wide" format to "tall" format. The following are significant functions found in the tidyr package:

Gather () and spread() function: The gather() function compiles a group of column names and consolidates them into a single key column. It also gathers the contents of those columns and combines them into a single value column. To use gather, you need to provide three arguments. The first is the key, which is an arbitrary label assigned to the new column we will create. The second is the value argument, which represents the name of the column that will hold the data. Lastly, you simply specify the columns you wish to merge into one.

```
>library(tibble)
```

```
>scores_wide <- tribble(
  ~student, ~e_1, ~e_2, ~e_3, ~e_4, ~e_5,
  "Alice", 85, 88, 90, 87, 92,
  "Bob",   78, 82, 85, 80, 83,
  "Cara",  92, 94, 91, 95, 96
)
>scores_wide
# A tibble: 3 x 6
[1]student   e_1  e_2  e_3  e_4  e_5
  <chr>   <dbl> <dbl> <dbl> <dbl> <dbl>
1 Alice     85    88    90    87    92
2 Bob       78    82    85    80    83
3 Cara      92    94    91    95    96
# Using gather (older tidyverse), or use pivot_longer as a modern alternative
>scores_tall <- scores_wide %>%
  gather(key = "exam", value = "score", -student)
>scores_tall
# A tibble: 15 × 3
 [1]student exam  score
   <chr>   <chr> <dbl>
 1 Alice   e_1      85
 2 Bob     e_1      78
 3 Cara    e_1      92
 4 Alice   e_2      88
 5 Bob     e_2      82
 6 Cara    e_2      94
 7 Alice   e_3      90
 8 Bob     e_3      85
 9 Cara    e_3      91
10 Alice   e_4      87
11 Bob     e_4      80
12 Cara    e_4      95
```

13 Alice e_5 92

14 Bob e_5 83

15 Cara e_5 96

Separate () function: Using the separate() function, we will extract a single column containing extensive information. We will then divide that information into several additional columns.

```
> ## for tidyver separate and unite function
> # Create a messy dataset
>messy_scores <- data.frame(
  student = c("Jon", "Amy", "Leo"),
  Term1_2020 = c(91, 84, 77),
  Term2_2020 = c(89, 86, 80),
  Term1_2021 = c(93, 88, 82),
  Term2_2021 = c(95, 87, 85)
)
>messy_scores
[1]  country q1_2017 q2_2017 q3_2017 q4_2017
1       A         0.03       0.05         0.04    0.03
2       B         0.05       0.07         0.05    0.02
3       C         0.01        0.02        0.01    0.04
#Reshape the messy data
tidier_scores <- messy_scores %>%
            gather(term_year, score, Term1_2020:Term2_2021)
>tidier_scores
[1]student  term_year      score
1     Jon Term1_2020        91
2     Amy Term1_2020       84
3     Leo Term1_2020        77
4     Jon Term2_2020        89
5     Amy Term2_2020       86
6     Leo Term2_2020        80
```

```
7    Jon Term1_2021        93
8    Amy Term1_2021        88
9    Leo Term1_2021        82
10   Jon Term2_2021        95
11   Amy Term2_2021        87
12   Leo Term2_2021        85
```

```
> # separate function - splinting the quarter from the year in the tidier dataset by applying the separate() function.
separate_tidier_scores <- tidier_scores %>% separate(term_year, c("term", "year"), sep = "_")
>head(separate_tidier_scores)
 [1] student    term     year score
1    Jon         Term1  2020    91
2    Amy        Term1   2020    84
3    Leo         Term1   2020    77
4    Jon        Term2   2020    89
5    Amy       Term2   2020    86
6    Leo       Term2    2020    80
```

Unite () function: To merge any two columns into one column it is convinient to use the unite function to combine several variable values into a single entity.

```
> # unite() function unites two columns into one
unit_tidier_scores <- separate_tidier_scores %>%
        unite(Term_Year, term, year, sep = "_")
 head(unit_tidier_scores)
[1]   student  Term_Year  score
1    Jon        Term1_2020     91
2    Amy      Term1_2020      84
3    Leo       Term1_2020     77
4    Jon       Term2_2020      89
5    Amy     Term2_2020      86
6    Leo       Term2_2020      80
```

Tidyverse offers a variety of other helpful functions, among which are the Nest() function, Unnest() function, Fill() function, Full-seq() function, Drop_na() function, and Replace_na() function.

stringer:

String manipulation plays a significant role in various data cleaning and preparation tasks. They developed the International Components for Unicode (ICU) C library to deliver quick and accurate implementations of standard string operations. The stringr package offers a unified collection of functions aimed at simplifying string handling as much as possible. It centers on the key and frequently utilized string manipulation functions.

```
>library(stringr)
>cities <- c("Delhi", "Tokyo", "Beijing", "Berlin", "Oslo", "Trivandrum", "Xyz")
>str_length(cities)  # Get string lengths
 [1] 5 5 7 6 4 10 3
>str_subset(cities, "[aeiouAEIOU]") # Return cities containing at least one vowel
[1] "Delhi"      "Tokyo"      "Beijing"    "Berlin"     "Oslo"       "Trivandrum"
str_detect(cities, "[aeiouAEIOU]") # Find out if each city contains at least one vowel
# [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
str_replace(cities, "[aeiouAEIOU]", "?") # Replace first vowel with "?"
# [1] "D?lhi"      "T?kyo"      "B?ijing"    "B?rlin"     "?slo"       "Tr?vandrum" "Xyz"
str_replace(cities, "[aeiouAEIOU]", "2") # Replace first vowel with "2"
# [1] "D2lhi"      "T2kyo"      "B2ijing"    "B2rlin"     "2slo"       "Tr2vandrum" "Xyz"
str_locate(cities, "[aeiouAEIOU]")  # Locate position of the first vowel
[1]      start end
 [1,]     2   2
 [2,]     2   2
 [3,]     2   2
 [4,]     2   2
 [5,]     1   1
 [6,]     3   3
 [7,]    NA  NA
```

forcats

A categorical variable plays a significant role in primary survey datasets, and in such cases, managing large sets of categorical variables demands a specialized R package. The forcats package will assist us in manipulating categorical variables. Factors can also aid in reordering character vectors for better display. The aim of the forcats package is to offer tools that address common issues with factors, such as altering the sequence of levels or the values.

Some examples include:

- fct_reorder(): Rearranging a factor based on another variable.
- fct_infreq(): Rearranging a factor according to the frequency of its values.
- fct_relevel(): Manually adjusting the sequence of a factor.
- fct_lump(): Merging the least or most common values of a factor into an "other" category.

```
>library(forcats)
>library(tibble)

# Use iris dataset's Species column as factor
>iris$Species <- factor(iris$Species)
>iris$Species
#Levels: setosa versicolor virginica
>class(iris$Species)
[1] "factor"
>levels(iris$Species)
[1] "setosa"     "versicolor" "virginica"
# Count number of each species
>fct_count(iris$Species, sort = TRUE)
# A tibble: 3 × 2
  f            n
  <fct>      <int>
1 setosa        50
2 versicolor    50
3 virginica     50
# Manually reorder the species levels
>fct_relevel(iris$Species, c("versicolor", "setosa", "virginica"))
# versicolor setosa virginica
# Reorder by frequency
>fct_infreq(iris$Species)
#Levels: setosa versicolor virginica
```

```
 # Reverse the factor levels
 >fct_rev(iris$Species)
#Levels: virginica versicolor setosa

>df <- tibble::tribble(
  ~color,     ~a, ~b,
  "cyan",      4,  3,
  "magenta",   7,  1,
  "orange",    2,  5,
  "black",     6,  2,
  "white",     3,  4
)
>df$color <- factor(df$color)
>df$color
[1] cyan    magenta orange  black   white
Levels: cyan magenta orange black white
>fct_reorder(df$color, df$a, min)
[1] cyan    magenta orange  black   white
Levels: orange white cyan black magenta

> # Creating two factors with different levels
> fac1 <- factor("cc")
> fac2 <- factor("dd")
> fct_c(fac1, fac2)
[1] cc dd
Levels: cc dd
>
> fct_unify(list(fac1, fac2)) # standardizing unique factor
[[1]]
[1] cc
Levels: cc dd

[[2]]
[1] dd
Levels: cc dd
```

3. Data import and management:

Getting data into R

The initial step in R involves bringing the data into R using text files, CSV, Excel files, or other formats. When you import a CSV or Excel file, printing the standard data.frame will display the entire content. You'll see all the rows (even if there are many) and every column, which can be difficult to read if your screen isn't wide enough, as the data may overflow into several rows.

```
library(tibble)
glass_sim <- tibble(
  RI    = c(1.52101, 1.51761, 1.51618, 1.51766, 1.51742, 1.51596, 1.51743, 1.51756,
            1.51918, 1.51755, 1.51571, 1.51763, 1.51589, 1.51748, 1.51763),
  Na    = c(13.64, 13.89, 13.53, 13.21, 13.27, 12.79, 13.30, 13.15, 14.04, 13.00,
            12.72, 12.80, 12.88, 12.86, 12.61),
  Mg    = c(4.49, 3.60, 3.55, 3.69, 3.62, 3.61, 3.60, 3.61, 3.58, 3.60, 3.46, 3.66, 3.43, 3.56, 3.59),
  Al    = c(1.10, 1.36, 1.54, 1.29, 1.24, 1.62, 1.14, 1.05, 1.37, 1.36, 1.56, 1.27, 1.40, 1.27, 1.31),
  Si    = c(71.78, 72.73, 72.99, 72.61, 73.08, 72.97, 73.09, 73.24, 72.08, 72.99, 73.20, 73.01,
73.28, 73.21, 73.29),
  K     = c(0.06, 0.48, 0.39, 0.57, 0.55, 0.64, 0.58, 0.57, 0.56, 0.57, 0.67, 0.60, 0.69, 0.54, 0.58),
  Ca    = c(8.75, 7.83, 7.78, 8.22, 8.07, 8.07, 8.17, 8.24, 8.30, 8.40, 8.09, 8.56, 8.05, 8.38, 8.50),
  Ba    = c(0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00),
  Fe    = c(0.00, 0.00, 0.00, 0.00, 0.00, 0.26, 0.00, 0.00, 0.00, 0.11, 0.24, 0.00, 0.24, 0.17, 0.00),
  Type  = as.factor(c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1))
)
glass_sim> tibble(Glass)
# A tibble: 15 × 10
     RI    Na    Mg    Al    Si     K    Ca    Ba    Fe Type
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>
 1  1.52  13.6  4.49  1.1   71.8  0.06  8.75     0  0    1
 2  1.52  13.9  3.6   1.36  72.7  0.48  7.83     0  0    1
 3  1.52  13.5  3.55  1.54  73.0  0.39  7.78     0  0    1
 4  1.52  13.2  3.69  1.29  72.6  0.57  8.22     0  0    1
 5  1.52  13.3  3.62  1.24  73.1  0.55  8.07     0  0    1
 6  1.52  12.8  3.61  1.62  73.0  0.64  8.07     0  0.26 1
 7  1.52  13.3  3.6   1.14  73.1  0.58  8.17     0  0    1
 8  1.52  13.2  3.61  1.05  73.2  0.57  8.24     0  0    1
 9  1.52  14.0  3.58  1.37  72.1  0.56  8.3      0  0    1
```

```
10  1.52  13    3.6   1.36  73.0  0.57  8.4      0  0.11 1
11  1.52  12.7  3.46  1.56  73.2  0.67  8.09     0  0.24 1
12  1.52  12.8  3.66  1.27  73.0  0.6   8.56     0  0    1
13  1.52  12.9  3.43  1.4   73.3  0.69  8.05     0  0.24 1
14  1.52  12.9  3.56  1.27  73.2  0.54  8.38     0  0.17 1
15  1.52  12.6  3.59  1.31  73.3  0.58  8.5      0  0    1
```

```
tibble(glass_sim)
# A tibble: 10 × 10
      RI    Na    Mg   Al    Si     K   Ca    Ba    Fe Type
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>
 1  1.52101 13.64 4.49 1.10 71.78 0.06  8.75 0     0      1
 2 1.51761 13.89 3.60 1.36 72.73 0.48  7.83 0     0      1
 3 1.51618 13.53 3.55 1.54 72.99 0.39  7.78 0     0      1
 4 1.51766 13.21 3.69 1.29 72.61 0.57  8.22 0     0      1
 5 1.51742 13.27 3.62 1.24 73.08 0.55  8.07 0     0      1
 6 1.51596 12.79 3.61 1.62 72.97 0.64  8.07 0     0.26   1
 7 1.51743 13.30 3.60 1.14 73.09 0.58  8.17 0     0      1
 8 1.51756 13.15 3.61 1.05 73.24 0.57  8.24 0     0      1
 9 1.51918 14.04 3.58 1.37 72.08 0.56  8.30 0     0      1
10 1.51755 13.00 3.60 1.36 72.99 0.57  8.40 0     0.11   1
# ... with 5 more rows
```

When examining a tibble, you'll see only the first 10 rows and as many columns as can be displayed on your screen without wrapping into another column. If there are more columns than can fit, they'll be shown at the bottom. Additionally, the datatype for each column is indicated at the top beneath the column headers. Below are some advantages of using tibble in comparison to base R data imports: a tibble preserves the input type, allows for nonstandard variable names, only recycles vectors of matching length, and does not generate row names.

reader: The reader R package offers streamlined tools for importing data from various files. It is crafted to flexibly handle multiple data types, even if some information is missing. Reader is compatible with the following file formats:

- • read_CSV(): files that are comma-separated (CSV)
- • read_tsv(): files separated by tabs
- • read_delim(): files with general delimiters
- • read_fwf(): files with fixed widths
- • read_table(): tabular files where columns are divided by whitespace

- • read_log(): files containing web logs

4. Functional programming:

R is a language based on functional programming that offers various tools for creating and manipulating functions. The purr package includes a range of helpful functions that enable you to utilize functional programming in R by providing a comprehensive and consistent set of tools for handling functions and vectors. The primary function in purr is referred to as map(). The map functions change their input by applying a specific function to every element, yielding a vector of the same length as the original input.

```
> # purr package
> #Map(): Utilize this when you want to apply a function to every element in the list or vector.
> library(purrr)
library(purrr)
>list_data <- list(5, 15, 25)
>list_data %>%
  map(function(x) x * 2)
[[1]]
[1] 10

[[2]]
[1] 30

[[3]]
[1] 50

> # map2: applying a function to a pair ofelements from a list of lists
>x <- list(5, 10, 15)
>y <- list(2, 4, 6)
>map2(x, y, ~ .x * .y)
[[1]]
[1] 10

[[2]]
[1] 40
```

[[3]]
[1] 90

> #pmap(): executing a function on a set of elements from a collection of lists.

>x <- list(2, 3, 4)

>y <- list(10, 20, 30)

>z <- list(100, 200, 300)

>pmap(list(x, y, z), sum) [[1]]

[1] 111

[[2]]
[1] 221

[[3]]
[1] 331

In addition to the functions described in these notes, there are numerous helpful lists of functions within the tidyverse. For further information, check out the following links and dive into the vast world of data science using R tidyverse.

Suggested Readings for future use:

- https://www.tidyverse.org/
- https://joeystanley.com/downloads/171110-tidyverse_handout.pdf
- https://www.guru99.com/r-dplyr-tutorial.html
- https://forcats.tidyverse.org/
- https://www.rdocumentation.org/packages/stringr/versions/1.4.0
- https://www.r-bloggers.com/2020/06/working-with-factors-in-r-tutorial-forcats-package/
- Search for cheat sheets in R https://www.rstudio.com/resources/cheatsheets/

Data Visualization using R

Yashavanth BS
ICAR-National Academy of Agricultural Research Management, Hyderabad
yashavanthbs@gmail.com

Data Visualization

Data visualization is the practice of displaying data in a visual or graphical format, which helps individuals better understand and interpret complex information. This process involves creating visual forms of data, such as charts, graphs, maps, and other visual elements, to highlight patterns, relationships, and trends that may not be easily identifiable in raw data alone. It plays a vital role across various sectors, including business, science, research, and journalism, by facilitating clear communication and promoting data-driven decisions. The main objective of data visualization is to offer insights and improve comprehension by converting abstract numbers and statistics into meaningful visuals. By representing data visually, complicated information becomes simpler and more accessible to a wider audience, including those who may not have specialized technical expertise. This method allows individuals to recognize patterns, spot anomalies, and quickly understand the overall structure and context of the data.

Depending on the type of data and the insights needed, there are different forms of data visualization. Common types include bar charts, column charts, line graphs, scatter plots, pie charts, and geographical maps.

Data Visualization in R

Data visualization in R represents a popular and effective method for producing visual displays of data. R is a software environment and programming language designed specifically for statistical analysis and graphics. It offers a diverse range of libraries and packages that allow users to generate various types of both static and interactive visualizations.

R includes several libraries that focus on data visualization, with ggplot2 being one of the most commonly used and adaptable packages. ggplot2 employs a layered method for constructing visualizations, where different elements are progressively added to create the final graphic. This package enables users to generate visually appealing and customizable graphics using relatively straightforward syntax..

Data Visualization using ggplot2

ggplot2 is an R package that focuses on data visualization and exploratory data analysis. It streamlines the creation of visually attractive plots and manages finer aspects like legends. Using ggplot2, plots can be constructed incrementally and readily adjusted afterwards.

This package adopts a layered strategy, enabling users to begin with a layer that reflects the raw data obtained from exploratory data analysis in R. Following that, users can incorporate annotations and statistical summaries to enhance the visual representations further.

Even those who are seasoned R users often look for help in producing visually appealing graphics. ggplot2 is an exceptional tool for creating graphics in R, even though it's common for users to consult cheat sheets after years of regular use. The package is founded on an extensive grammar known as "Graphics Grammar," which includes a set of independent components that can be mixed and matched in different ways. This grammar entails a core set of rules and principles..

The components

- *- Data: In ggplot2, information needs to be organized as an R data frame.*
- *- Coordinate system: Defines the 2-D space onto which the data is mapped (including Cartesian coordinates, polar coordinates, map projections, etc.).*
- *- Geoms: Represent the various types of geometric entities that illustrate the data (such as points, lines, polygons, etc.).*
- *- Aesthetics: Indicate the visual properties that represent the data (like position, size, color, shape, transparency, and fill).*
- *- Scales: For each aesthetic, explain how the visual property is transformed into display values (including log scales, color scales, size scales, shape scales, etc.).*
- *- Stats: Outline statistical transformations that typically provide a summary of the data (such as counts, means, medians, regression lines, etc.).*
- *- Facets: Specify how the data is divided into subsets and shown as several small graphs.*

Let's attempt to create some charts utilizing a dataset. For this purpose, we will make use of the hsb (high school and beyond) dataset found in the descriptr package.

To begin, we should set up and import the descriptr package by using the commands below:

```
install.packages("descriptr")
library(descriptr)
```

Next, we can import the hsb dataset and store it as a dataframe with the same title.

```
hsb <- hsb
```

The hsb dataset includes demographic information and standardized test scores for high school students. It has 200 rows and 10 variables. Here are the details of the variables:

id: id of the student
female: gender of the student
race: ethnic background of the student
ses: socio-economic status of the student
schtyp: school type
prog: program type
read: scores from test of reading
write: scores from test of writing

math: scores from test of math
science: scores from test of science
socst: scores from test of social studies

Out of these, the variables female, race, ses, schtyp, and prog are categorized as factor variables, while the remaining ones are continuous variables. Now, let's proceed to label some of these factor variables.

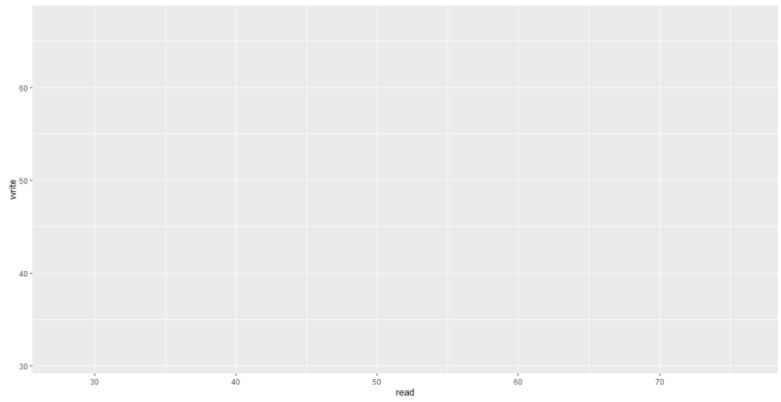
```
hsb$female <- factor(hsb$female, labels = c("Male", "Female"))
hsb$race <- factor(hsb$race, labels = c("American", "Asian", "African", "European"))
hsb$ses <- factor(hsb$ses, labels = c("Poor", "Average", "Rich"))
```

Let's utilize this data to create various plots. The commands below will install and load the ggplot2 package necessary for this purpose.

```
install.packages("ggplot2")
library(ggplot2)
```

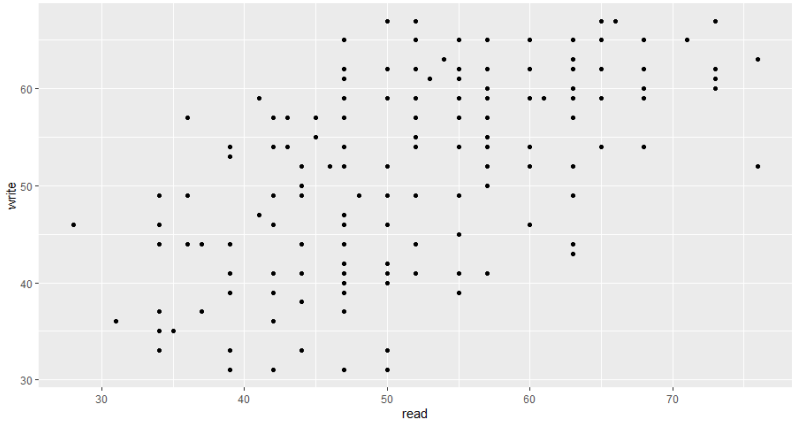
We will now explore how the ggplot2 package functions by using an example of creating a scatter plot. The scatter plot is employed to illustrate the relationship between two continuous variables. Using the hsb dataset, we will attempt to create a scatter plot that compares the 'read' and 'write' variables. In the command that follows, we are assigning the 'read' variable to the x-axis and the 'write' variable to the y-axis. Please note that this command will merely generate a canvas onto which the points representing each pair of read and write values will be plotted.

```
ggplot(hsb, aes(x=read, y=write))
```



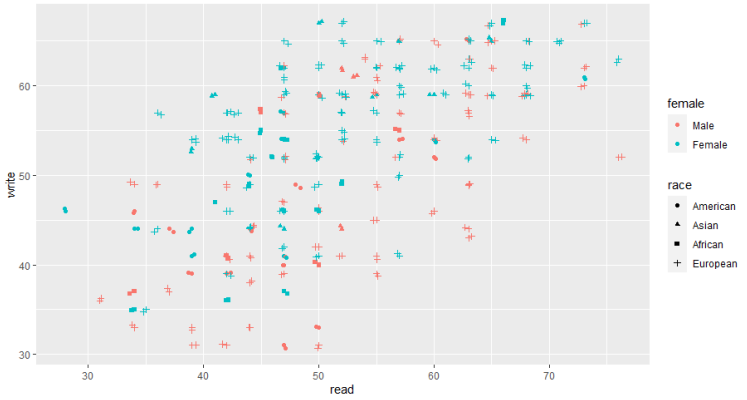
Now, let's incorporate the dots into the canvas created by utilizing the geometry function geom_point().

```
ggplot(hsb,aes(x=read,y=write)) + geom_point()
```



At this point, we observe that a layer of dots has been incorporated into the canvas. In the same way, additional layers can be added later on by utilizing the plus symbol. Now, let's apply some additional functions to alter the plot.

```
ggplot(hsb,aes(x=read,y=write, shape = ses)) + #changes shape based on ses
  geom_point()
ggplot(hsb,aes(x=read,y=write,
               shape = race, #changes shape based on race
               col= female)) + #changes color based on gender
  geom_point() +
  geom_jitter() # adds the jitter effect to avoid overlapping.
```



- In a similar fashion, we can utilize various functions to create a suitable plot that aligns with our vision or needs. Now, let’s attempt to produce some additional fundamental plots, starting with a bar chart.

```
ggplot(data=hsb,aes(x=race)) +
  geom_bar(stat="count")

ggplot(data=hsb,aes(x=race))+
  geom_bar(stat="count", # gives the count
                  fill="steelblue", # adds steelblue color to the bars
```

```
                width = 0.5) # defines the bar width to 0.5
```

- *Bar plot for average Read Score*

```
ggplot(data=hsb,aes(x=race,y=read)) +
 geom_bar(stat="summary", fun.y="mean", fill="steelblue")
```

- *Stacked Bar Chart*

```
ggplot(data=hsb,aes(x=race, fill=ses)) +
 geom_bar(stat="count")
```

- *Grouped Bar Chart*

```
ggplot(data=hsb,aes(x=race,fill=ses)) +
 geom_bar(stat="count", position="dodge")
```

- *Percentage Stacked Bar Chart*

```
ggplot(data=hsb,aes(x=race,fill=ses)) +
 geom_bar(stat="count", position="fill")
```

- *Horizontal Barplot*

```
ggplot(data=hsb,aes(x=race)) +
 geom_bar(stat="count", fill="steelblue", width = 0.5) +
 coord_flip() # this will flip the coordinates
```

- *Pie Chart*

```
ggplot(data=hsb,aes(x="", fill=female)) +
 geom_bar(stat="count")+
 coord_polar("y", start=0) # for polar coordinates
```

- *Histogram*

```
ggplot(data=hsb,aes(read)) +
 geom_density()
```

- *boxplot*

```
ggplot(data=hsb,aes(x="", y=read)) +
 geom_boxplot()
```

- *Boxplot showing two levels*

```
ggplot(data=hsb,aes(x=race,y=read)) +
 geom_boxplot()
```

- *Boxplot showing two levels with mean*

```
ggplot(data=hsb,aes(x=race,y=read)) +
 geom_boxplot() +
```

```
 stat_summary(fun=mean, geom="point") # this will add a point to denote the mean
```

To access the available geometry options, one can execute the following command.

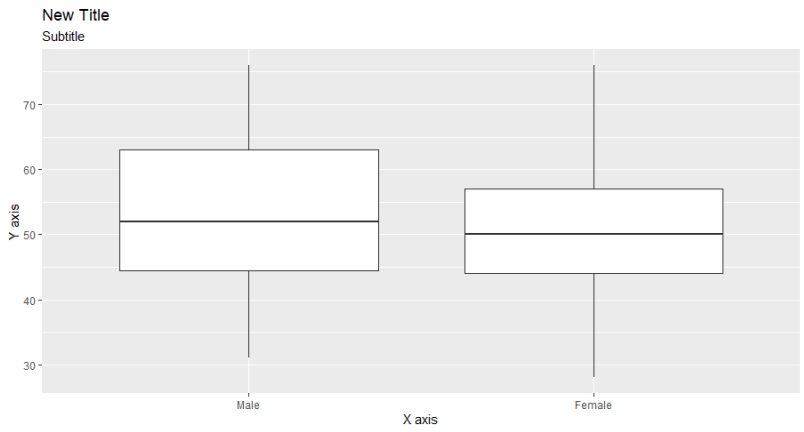
```
geoms <- help.search("^geom_", package = "ggplot2")
geoms$matches[, 1:2]
```

Titles and axes labels

The labels for the x-axis and y-axis, as well as the plot's title and subtitle, can be included in two different methods. One method involves using individual functions for each element, while the other utilizes a single function called labs().

```
ggplot(data=hsb,aes(x=female,y=read))+geom_boxplot() +
  ggtitle("Boxplot for Read") +
  xlab("Gender") +
  ylab("Marks")

ggplot(data=hsb,aes(x=female,y=read))+geom_boxplot() +
  labs(title="New Title", subtitle="Subtitle", x="X axis", y="Y axis")
```



Faceting

Faceting is a crucial feature that allows us to create multiple mini plots for enhanced data visualization.

For instance, the command below produces a scatterplot depicting the variables read and write. Additionally, the shape of the points is determined by the student's race, while the color of the points indicates the student's socio-economic status.

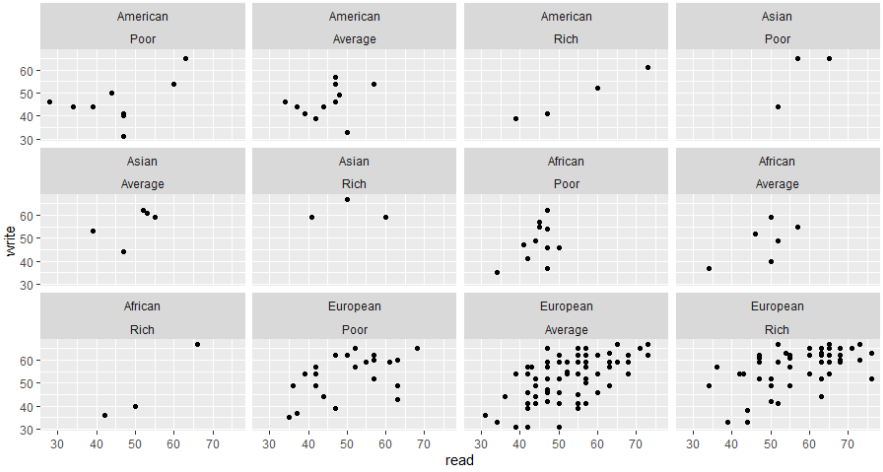
```
ggplot(hsb,aes(x=read,y=write, shape = race, col=ses)) +
  geom_point()
```

While the plot provides useful information, an improved visualization would involve dividing it into separate panels. This can be accomplished by employing the facet_wrap() and facet_grid() functions.

```
ggplot(hsb,aes(x=read,y=write, col=ses)) +
  geom_point() +
  facet_wrap(~race)

ggplot(hsb,aes(x=read,y=write)) +
  geom_point() +
  facet_wrap(~race*ses)

ggplot(hsb,aes(x=read,y=write)) +
  geom_point() +
  facet_grid(race~ses)
```



Themes

By default, ggplot2 applies a gray theme to all generated plots. The ggplot2 package offers various themes that can be easily utilized based on your needs.

Boxplot is generated by this commands:

```
ggplot(data=hsb,aes(x=race,y=read, fill=race)) +
  geom_boxplot() +
  facet_wrap(~female) +
  labs(title="Race-wise Boxplot for Marks",
       subtitle="(for the course 'Read')",
       x="Race", y="Marks obtained", fill="Race") -> p1
```

The plot has been saved as an R object called 'p1'. We can make modifications to this saved image until we reach the desired appearance. Keep in mind that once a plot is stored as an R object, it will not appear in the plot pane.

For the plot 'p1', you can explore various themes provided by the ggplot2 package. Below are some commands to try different themes.

```
p1 + theme_minimal()
```

```
p1 + theme_bw()
p1 + theme_void()
p1 + theme_classic()
p1 + theme_dark()
p1 + theme_gray()
p1 + theme_light()
p1 + theme_linedraw()
p1 + theme_test()
```

There are various other packages available, including hrbrthemes, cowplot, and ggthemes, which can be utilized for distinct themes.

Customized themes

In addition to the available themes, users have the ability to customize their own theme utilizing the theme() function. There are options to modify the panel color, grid color, background color, as well as text size and color, among others. Several of these features have been demonstrated in the commands below.

```
p1 + theme(panel.background=element_rect(fill="aquamarine1")) -> p2
p2 + theme(plot.title = element_text(hjust = 0.5,
                                     face="bold",
                                     colour="darkblue")) -> p3
p3 + theme(legend.background = element_rect(fill="lemonchiffon1")) -> p4
p4 + theme(plot.background = element_rect(fill="lightcoral")) -> p5
p5 + theme(legend.position = "top") +
          theme(legend.title=element_blank()) -> p6
p6 + theme(plot.subtitle = element_text(hjust=1, color ="yellow")) -> p7
p7 + theme(strip.background = element_rect(fill="skyblue"))
```

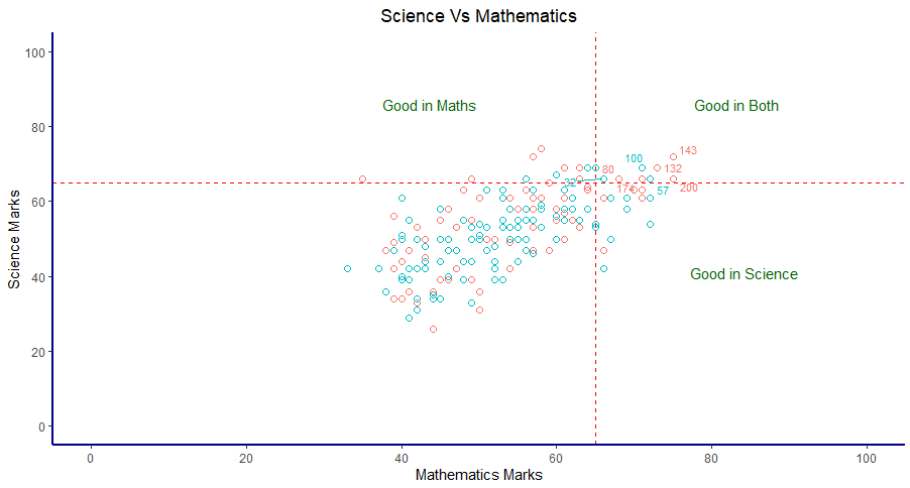
The initial and concluding results are presented in the image.

Annotating, horizontal and vertical lines

The example provided illustrates how horizontal lines, vertical lines, and annotations can enhance the informative quality of a plot.

```
ggplot(data=hsb,aes(x=math,y=science,colour=female))+
  geom_point(shape=1, size=2) +
  #add a horizontal line
  geom_hline(yintercept = 65, color="red", size=0.7, linetype = "dashed") +
  #add a vertical line
  geom_vline(xintercept = 65, color="red", size=0.7, linetype = "dashed") +
  #customize the x and y axis limits
  scale_x_continuous(limits = c(0,100), breaks = seq(0,100,20)) +
  scale_y_continuous(limits = c(0,100), breaks = seq(0,100,20)) +
```

```
#add labels
labs(x="Mathematics Marks", y="Science Marks",
    title = "Science Vs Mathematics") +
#customize theme
theme_classic() + theme(legend.position = "none") +
theme(plot.title = element_text(hjust = 0.5),
     axis.line = element_line(colour = "darkblue",
                              size = 1, linetype = "solid")) +
#Annotate
annotate("text", label="Good in Both",
        x=80,y=85,hjust=.2,vjust=.2,color="darkgreen") +
annotate("text", label="Good in Science",
        x=80,y=40,hjust=.2,vjust=.2,color="darkgreen") +
annotate("text", label="Good in Maths",
        x=40,y=85,hjust=.2,vjust=.2,color="darkgreen") +
#add labels for points
#geom_text_repel(aes(label=id)) +
geom_text_repel(aes(label=ifelse(math > 65 & science > 65,id,"")),
            size=3)
```



The example provided illustrates how horizontal lines, vertical lines, and annotations can enhance the informative quality of a plot.

qplot

The qplot() function can also create plots. Nevertheless, not all the features and functionalities that are present in the ggplot() function are available in the qplot() function.

```
qplot(read, write, data=hsb, geom="point")
qplot(read, write, data=hsb, geom=c("point","smooth"))
qplot(read, write, data=hsb, geom=c("point","smooth"), color=female)
```

Saving/ exporting plots

You can save plots created with ggplot2 in your working directory or any specified folder by using the ggsave function. This function allows you to save the most recently displayed plot in different formats, including .jpg and .png.

```
ggsave(file="Plot1.jpg")
```

Alternatively, similar to any other plots, the created visuals can be exported as images by accessing the plots pane. The images can also be saved in PDF format. Additionally, these visuals can be copied to the clipboard and inserted into any Word or PowerPoint document.

Further Reading:

Hadley Wickham (2016). ggplot2: Elegant Graphics for Data Analysis, Springer International Publishing

Chang W. (2018). R Graphics Cookbook, O'Reilly Media.

Developing R Package and Interactive Shiny Applications

Nobin Chandra Paul[1], Bijoy Chanda[2], Ponnaganti Navyasree[1], K. Ravi Kumar[1], Santosha Rathod[1] and Prabhat Kumar[1]

[1]ICAR-National Institute of Abiotic Stress Management, Baramati, Pune-413115

[2]ICAR- Central Agroforestry Research Institute, Jhansi, Uttar Pradesh-284003

Email: ncp375@gmail.com

1. Introduction

R is a powerful statistical programming language widely used in data science, machine learning, and research. Its extensibility allows users to develop R packages to share code, functions, and datasets efficiently. Developing an R package is essential for creating reusable, maintainable code, contributing to the open-source community, and enhancing reproducibility in research. This lecture provides a step-by-step guide to developing an R package named DescStats for calculating descriptive statistics, including mean, variance, minimum, maximum, skewness, kurtosis, standard deviation, and range, using RStudio. The package will include well-documented functions and tests to ensure reliability.

2. Why Develop an R Package and Shiny app?

Developing an R package offers numerous advantages, making it a valuable endeavour for researchers, data scientists, and developers. By bundling code, data, documentation, and interactive applications like Shiny apps into a standardized format, R packages enhance the efficiency, accessibility, and impact of your work. Below are key reasons to develop an R package, with an emphasis on its benefits for reproducibility, collaboration, intellectual property protection, and community contribution:

- ✓ ***Code Reusability:*** R packages allow you to encapsulate functions, datasets, and scripts into a single, reusable unit. This enables you to apply the same code across multiple projects or share it with others, saving time and reducing redundancy. For example, a package for descriptive statistics can be reused in various analyses without rewriting code.
- ✓ ***Collaboration:*** Packages promote standardized coding practices, making it easier for teams to collaborate. By sharing a package, team members can access consistent

functions and datasets, ensuring uniformity in analyses and reducing errors from ad-hoc scripts.

- ✓ ***Documentation:*** Well-documented packages, using tools like roxygen2, provide clear instructions and examples for users. Comprehensive documentation improves usability, helps users understand the package's functionality, and ensures long-term maintainability, especially for complex tasks like statistical modeling or data visualization.
- ✓ ***Copyright Protection and Attribution:*** R packages can be licensed under open-source licenses (e.g., MIT, GPL) that legally protect your intellectual property. These licenses specify how others can use, modify, or distribute your work, ensuring you receive proper attribution while allowing controlled sharing. This is particularly important for maintaining ownership of novel algorithms or methodologies.
- ✓ ***Contribution to the Community:*** Publishing packages on platforms like CRAN or GitHub makes your work accessible to a global audience. This fosters collaboration, encourages feedback, and establishes you as a contributor to the R community, enhancing your professional reputation and impact.
- ✓ ***Reproducibility and Transparency:*** Packages promote reproducible research by bundling code, data, and documentation together. Users can replicate your analyses exactly as intended, which is critical in fields like statistics, where transparency is essential for validating results.
- ✓ ***Integration with Shiny Apps:*** R packages can include Shiny applications, enabling the creation of interactive web-based tools for data exploration and visualization. By embedding Shiny apps in a package, you can provide users with user-friendly interfaces to interact with your functions, such as exploring descriptive statistics or visualizing results dynamically.
- ✓ ***Version Control and Maintenance:*** Packages integrate seamlessly with version control systems like Git, allowing you to track changes, manage updates, and maintain a stable codebase. This ensures that users always access the latest, bug-free version of your tools.
- ✓ ***Extensibility and Modularity:*** Packages allow you to build modular code that can be extended over time. For instance, a descriptive statistics package can later incorporate advanced visualizations or new statistical methods, making it adaptable to future needs.

- ✓ ***Professional Development:*** Developing an R package hones your programming skills, deepens your understanding of R's ecosystem, and demonstrates expertise to employers or collaborators. It also showcases your ability to create robust, user-friendly tools for data analysis.

By developing an R package, you create a powerful, reusable tool that enhances collaboration, protects your intellectual contributions, and supports interactive applications like Shiny apps, all while contributing to the broader R community.

3. Developing an R Package for Descriptive Statistics

Steps to Develop an R Package:

a. Setting Up the Package

To create an R package, start by installing the required packages:

```
install.packages("devtools")
install.packages("usethis")
install.packages("roxygen2")
install.packages("testthat")
library(devtools)
library(usethis)
```

Below is a concise explanation of the use of each R package mentioned-devtools, roxygen2, usethis, and testthat-in the context of R package development. These packages are essential for streamlining the creation, documentation, testing, and maintenance of R packages.

- ***devtools – Development Tools for R Packages***

Purpose:
The devtools package streamlines and simplifies the entire R package development workflow. It provides functions to create, build, install, load, test, and check packages with minimal effort.

Key Functions:
create() – Start a new package project.
load_all() – Load all functions without reinstalling the package.
document() – Generate documentation from roxygen2 comments.

install() – Install the current package.
check() – Run CRAN-style checks for completeness and errors.

Use in practice:
When you are actively developing an R package, you can use devtools::load_all() after each update to instantly use your latest code without rebuilding.

- ***roxygen2 – Documentation Made Easy***

Purpose: roxygen2 allows you to write documentation directly above your R functions using simple comments. It then automatically generates the corresponding .Rd help files that R uses internally.

Key Features:
Use #' to add documentation for each function.
Supports tags like **@param, @return, @examples**, and **@export.**
Integrates with devtools::document() to generate man/ files and NAMESPACE.

Use in practice:
Instead of writing separate help files, you write doc-comments inline, making it easier to maintain and synchronize your code and documentation.

- ***usethis – Workflow Automation***

Purpose: usethis helps automate common setup tasks in R package development. It removes the need for manual file editing and configuration, allowing you to focus on writing code.

Key Functions:
use_package('ggplot2') – Adds a dependency to your DESCRIPTION.
use_r('my_function') – Creates an R script file in the right place.
use_test('my_function') – Sets up test files for a function.
use_git() / use_github() – Sets up version control and GitHub integration.
use_mit_license() – Adds a license file to your package.

Use in practice:
Running usethis::use_testthat() will set up the entire testing framework automatically—no manual directory creation needed.

- ***testthat – Unit Testing Framework***

Purpose: **testthat** is the most popular testing framework in R. It allows you to write and run unit tests to verify that your functions behave correctly and consistently.

Key Features:

Simple syntax: expect_equal(), expect_error(), expect_type(), etc.
Organizes tests under the tests/testthat/ directory.
Integrates seamlessly with devtools::test().

Use in practice:
Write tests like:

```
test_that('mean works correctly', {
  expect_equal(mean(c(1, 2, 3)), 2)
})
```

to automatically validate your functions during development and before release.

b. Set Up the R Package Structure

Open RStudio and create a new R package project:

Go to File > New Project > New Directory > R Package.
Name the package DescStats and choose a directory.

c. Understanding R Package Structure

An R package consists of:

DESCRIPTION – Metadata about the package (name, title, author, version, dependencies).

NAMESPACE – Defines exported functions.

R/ – Holds R scripts files with functions.

man/ Holds documentation for each function

tests/ – Contains unit tests to verify the correctness of functions.

vignettes/ – Offers detailed documentation and practical use cases.

data/ – (Optional) Sample datasets

d. Writing Functions

Develop functions inside the **R/** directory. Create a file, e.g., **desc_stats.R**, with functions for descriptive statistics. Below is an example function to compute all requested statistics:

```
#' Calculate Descriptive Statistics
#'
#' Computes mean, variance, minimum, maximum, skewness, kurtosis, standard deviation, and range for a numer
ic vector.
#'
#' @param x A numeric vector.
#' @return A named list containing mean, variance, minimum, maximum, skewness, kurtosis, standard deviatio
n, and range.
#' @examples
#' x <- c(1, 2, 3, 4, 5)
#' descriptive_stats(x)
#' @export
descriptive_stats <- function(x) {
  if (!is.numeric(x)) stop("Input must be a numeric vector")
  if (length(x) == 0) stop("Input vector cannot be empty")

  list(
    mean = mean(x, na.rm = TRUE),
    variance = var(x, na.rm = TRUE),
    minimum = min(x, na.rm = TRUE),
    maximum = max(x, na.rm = TRUE),
    skewness = moments::skewness(x, na.rm = TRUE),
    kurtosis = moments::kurtosis(x, na.rm = TRUE),
    std_dev = sd(x, na.rm = TRUE),
    range = max(x, na.rm = TRUE) - min(x, na.rm = TRUE)
  )
}
```

The **moments package is required for skewness and kurtosis. Ensure it is listed in the **DESCRIPTION** file under **Depends**.*

e. Documenting Functions

Use **roxygen2** for documentation. The comments above the function (starting with #') generate documentation files. Run: **devtools::document()**

f. Testing the Package

Testing ensures reliability. Set up testing with testthat:

```
#r console
usethis::use_testthat()
usethis::use_test("desc_stats")
```

Run tests using:

```
#r console
devtools::test()
```

g. Checking and Building the Package

Before releasing, check and build the package:

Fix any errors or warnings (e.g., missing dependencies or documentation issues). Build the package:

```
#r console
devtools::check()
devtools::build()
```

This creates a .tar.gz file in the parent directory. Install and test locally:

```
install.packages("path/to/DescStats_0.1.0.tar.gz", repos = NULL, type = "source")

library(DescStats)

x <- c(1, 2, 3, 4, 5)

descriptive_stats(x)
```

h. Publish the Package

A. Publish on GitHub:

```
#r console
usethis::use_git()
usethis::use_github()
```

B. Publish on CRAN

To publish on CRAN:

Ensure devtools::check() passes with no errors or warnings.

Submit using usethis::use_cran_submission().

Follow CRAN guidelines and respond to reviewer feedback.

```
#r console
usethis::use_cran_submission()
```

Adding a Sample Dataset (Optional) Include a sample dataset for testing:

```
#r console
usethis::use_data_raw()
```

In data-raw/, create a script, e.g., dataset.R:

```
desc_data <- data.frame(
  values = c(rnorm(100, mean = 10, sd = 2))
)
usethis::use_data(desc_data, overwrite = TRUE)
```

Example Usage:

```
library(DescStats)
data(desc_data)
descriptive_stats(desc_data$values)
```

4. Developing an Interactive Shiny App for Descriptive Statistics Visualization and Reporting in R

To begin building a Shiny app, we need to understand that the app consists of two main components: the **UI (User Interface)** and the **server logic**, which are tied together using the **shinyApp()** function. The goal of our app is to allow users to upload or choose sample data, select a numeric column, view descriptive statistics, visualize the distribution as a histogram, and optionally download the results.

We first define a custom function called **descriptive_stats()** which accepts a numeric vector and calculates common statistical measures such as mean, variance, minimum, maximum, skewness, kurtosis, standard deviation, and range. This function uses the moments package for skewness and kurtosis, and the stats package for the rest.

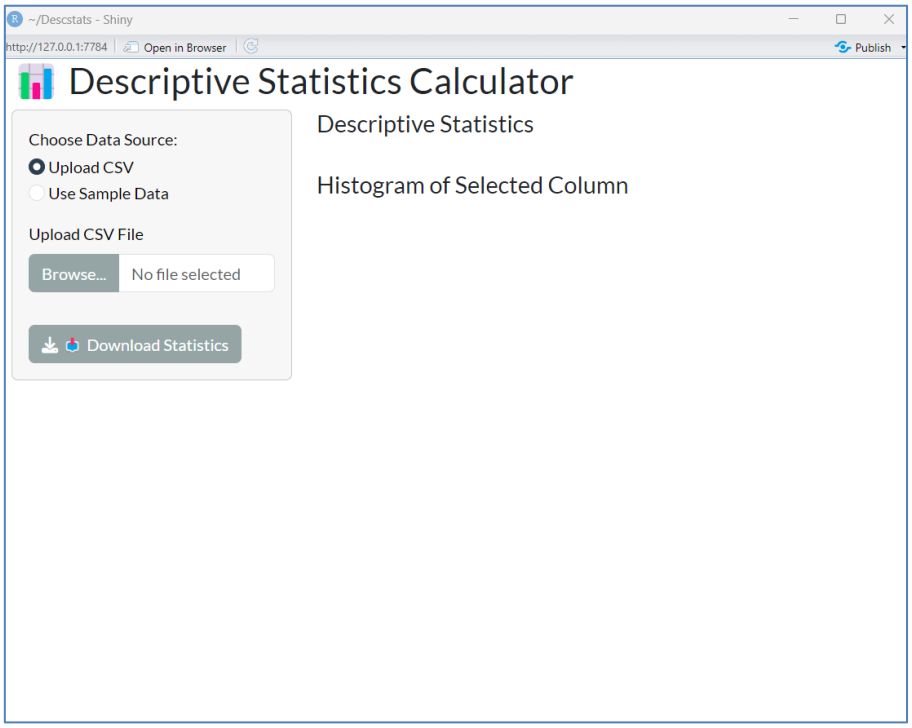
Next, we design the UI using **fluidPage()**, which organizes our app layout. The sidebar contains input elements such as radioButtons to choose between uploading a CSV file or using a sample

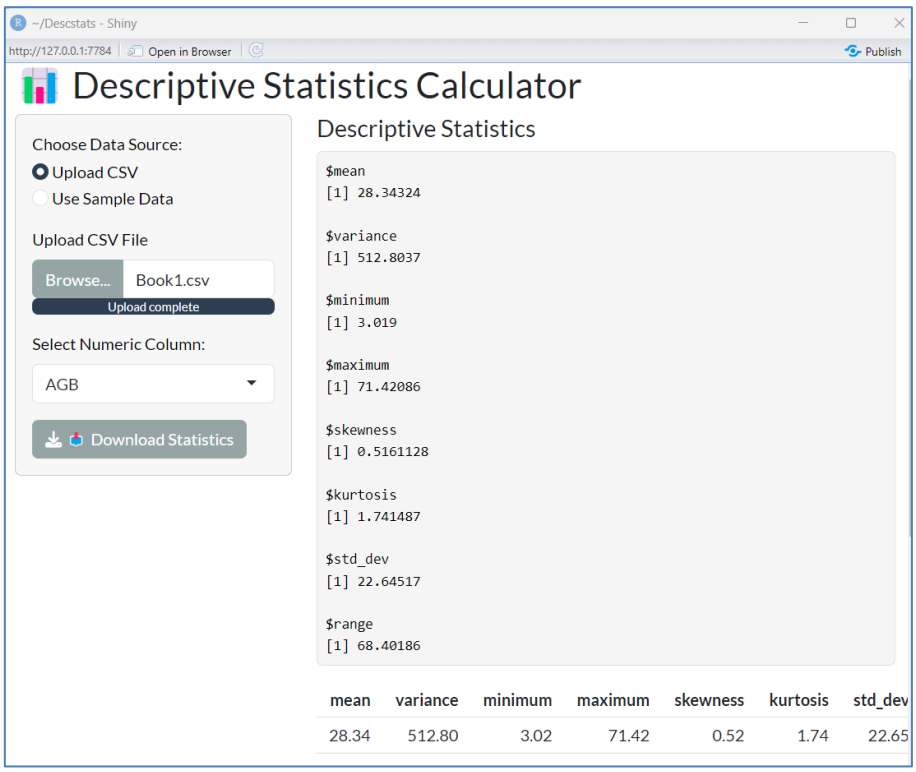
dataset, fileInput to upload the file, and a selectInput (rendered dynamically) to let users choose from available numeric columns. There's also a downloadButton to allow downloading the computed statistics as a CSV file.

On the main panel, we display the statistics in two formats: a print-style output using **verbatimTextOutput()** and a formatted table using **tableOutput()**. Below that, we include a histogram plot created with **plotOutput()**, which visually shows the distribution of the selected variable.

In the server logic, we define reactive expressions to read the uploaded or sample dataset, filter out numeric columns, and apply our descriptive_**stats()** function to the selected column. The computed values are displayed both as text and a table. The histogram is generated using R's hist() function with custom colors for better aesthetics. Finally, a downloadHandler is used to convert the statistics into a data frame and let the user save the results locally.

To launch the app, the **shinyApp(ui, server)** function binds the user interface and server together and runs the application. Participants can run this app by saving the script as **app.R** and executing **shiny::runApp("app.R")** in the R console.


~/Descstats - Shiny
http://127.0.0.1:7784
Open in Browser
Publish
Descriptive Statistics Calculator
Choose Data Source:
Upload CSV
Use Sample Data
Upload CSV File
Browse...
No file selected
Download Statistics
Descriptive Statistics
Histogram of Selected Column


~/Descstats - Shiny
http://127.0.0.1:7784
Open in Browser
Publish
Descriptive Statistics Calculator
Choose Data Source:
Upload CSV
Use Sample Data
Upload CSV File
Browse...
Book1.csv
Upload complete
Select Numeric Column:
AGB
Download Statistics
Descriptive Statistics
$mean
[1] 28.34324
$variance
[1] 512.8037
$minimum
[1] 3.019
$maximum
[1] 71.42086
$skewness
[1] 0.5161128
$kurtosis
[1] 1.741487
$std_dev
[1] 22.64517
$range
[1] 68.40186
mean variance minimum maximum skewness kurtosis std_dev
28.34 512.80 3.02 71.42 0.52 1.74 22.65

```
# Complete Shiny R Code #
# Load required libraries
library(shiny)
library(moments)
library(bslib)

# Descriptive statistics function
descriptive_stats <- function(x) {
  if (!is.numeric(x)) stop("Input must be a numeric vector")
  if (length(x) == 0) stop("Input vector cannot be empty")

  list(
    mean = mean(x, na.rm = TRUE),
    variance = var(x, na.rm = TRUE),
    minimum = min(x, na.rm = TRUE),
    maximum = max(x, na.rm = TRUE),
    skewness = moments::skewness(x, na.rm = TRUE),
    kurtosis = moments::kurtosis(x, na.rm = TRUE),
    std_dev = sd(x, na.rm = TRUE),
    range = max(x, na.rm = TRUE) - min(x, na.rm = TRUE)
  )
}

# UI
ui <- fluidPage(
  theme = bs_theme(version = 5, bootswatch = "flatly"),
  titlePanel("📊 Descriptive Statistics Calculator"),

  sidebarLayout(
    sidebarPanel(
      radioButtons("data_source", "Choose Data Source:",
                   choices = c("Upload CSV" = "upload", "Use Sample Data" = "sample")),

      conditionalPanel(
        condition = "input.data_source == 'upload'",
        fileInput("file", "Upload CSV File", accept = ".csv")
      ),

      uiOutput("column_selector"),

      downloadButton("download_btn", "📥 Download Statistics")
    ),

    mainPanel(
      h4("Descriptive Statistics"),
      verbatimTextOutput("stats_output"),
      tableOutput("stats_table"),
```

```
      br(),
      h4("Histogram of Selected Column"),
      plotOutput("hist_plot", height = "300px")
    )
  )
)
# Server
server <- function(input, output, session) {

  # Load data based on user selection
  dataset <- reactive({
    if (input$data_source == "sample") {
      data.frame(Sample = c(10, 12, 15, 18, 20, 25, 30))
    } else {
      req(input$file)
      read.csv(input$file$datapath)
    }
  })
  # Dynamically show numeric columns for selection
  output$column_selector <- renderUI({
    df <- dataset()
    num_cols <- names(df)[sapply(df, is.numeric)]
    if (length(num_cols) == 0) return(h6("No numeric columns available."))
    selectInput("column", "Select Numeric Column:", choices = num_cols)
  })
  # Compute Descriptive Statistics
  stats_result <- reactive({
    req(input$column)
    col_data <- dataset()[[input$column]]
    descriptive_stats(col_data)
  })
  output$stats_output <- renderPrint({
    stats_result()
  })
  output$stats_table <- renderTable({
    as.data.frame(t(unlist(stats_result())), row.names = FALSE)
  })
  # Plot Histogram
  output$hist_plot <- renderPlot({
    req(input$column)
    col_data <- dataset()[[input$column]]
    hist(col_data, col = "#69b3a2", border = "white",
         main = paste("Histogram of", input$column),
         xlab = input$column)
  })
  # Download handler
  output$download_btn <- downloadHandler(
```

```
    filename = function() {
      paste0("descriptive_stats_", input$column, ".csv")
    },
    content = function(file) {
      write.csv(as.data.frame(t(unlist(stats_result()))), file, row.names = FALSE)
    }
  )
}
# Run the app
shinyApp(ui, server)
```

5. Conclusions

Developing an R package like DescStats enhances reproducibility, usability, and collaboration in statistical analysis. R packages offer a powerful ecosystem for data analysis, providing specialized tools for statistical modeling, data visualization, and machine learning, with seamless integration and reproducibility. They enable efficient workflows through modular, well-documented functions, fostering collaboration and innovation. As a culmination, Shiny apps enhance this ecosystem by delivering interactive, user-friendly interfaces for data exploration and visualization. Built on R's robust capabilities, Shiny empowers users to create dynamic web applications, making complex analyses accessible to non-technical stakeholders, thus bridging the gap between data science and practical decision-making.

Suggested Readings

https://yoshitha-akunuri-21201.medium.com/publish-your-r-package-to-cran-65a0464e8df7

https://www.r-bloggers.com/2020/07/how-to-write-your-own-r-package-and-publish-it-on-cran/

Introduction to BlueSky Statistics

Prabhat Kumar, Ponnaganti Navyasree, Nobin Chandra Paul, K. Ravi Kumar and Santosha Rathod
ICAR-National Institute of Abiotic Stress Management, Baramati, Pune-413115
Email: prabhatkkv@gmail.com

1. Introduction to BlueSky Statistics

BlueSky Statistics is a free, user-friendly statistical software built on the R language. It is designed to bridge the gap between the power of R and the simplicity of a GUI. With its spreadsheet-like interface and menu-driven commands, it is well-suited for both beginners and experienced users. BlueSky supports a wide range of statistical analyses and graphical outputs. Its modular design also allows for the integration of additional R packages and tools. It is commonly used in research, education, agriculture, and business analytics.

2. What is BlueSky Statistics?

BlueSky Statistics acts as a front-end GUI for R, allowing users to access the strength of R without coding. It retains the flexibility of scripting while simplifying analysis through dialog boxes and menus. Users can perform statistical procedures, model data, and visualize results easily. It's especially helpful for those transitioning from SPSS or Excel-based workflows. Since it's built on R, advanced users can write or modify scripts as needed. This makes it a versatile tool for both teaching and research.

3. Graphical User Interface of BlueSky

The interface is clean, intuitive, and similar to other point-and-click statistical software like SPSS. It includes:

- A Data Editor for viewing and editing data,
- An Output Window for displaying results,
- A Script Editor for viewing and running R code,
- Navigator pane for managing variables and data frames. Menus are arranged clearly across the top, making it easy to access tools and analysis options. This GUI design reduces the learning curve for new users.

4. Download and Installation

BlueSky Statistics is available at www.blueskystat.com under the Free Desktop Edition. Download the installer suitable for Windows OS. The setup includes a built-in version of R, so no separate installation is needed. After installation, you can launch the application directly from your desktop. The software does not require an internet connection to run. Updates and newer plugins can be added later through the Add-on Manager.

5. Data Types in BlueSky Statistics

BlueSky supports a variety of data types necessary for statistical analysis. These include:

- Numeric: for continuous and integer variables
- Factor (Categorical): for grouped or class-based data
- Character (Text): for labels or string-based data
- Logical and Date/Time: for boolean and temporal data. Users can modify variable types directly through the Variable View or Data menu. Recognizing the correct data type ensures accurate statistical results and appropriate visualizations.

6. Key Menu Options and Tools in BlueSky

BlueSky provides several menus with categorized tools:

- **File Menu**: for creating, opening, importing, and saving data files.
- **Data Menu**: includes sorting, filtering, transforming, and recoding variables.
- **Analysis Menu**: hosts a variety of procedures like t-tests, ANOVA, regression, chi-square tests, etc.
- **Graphs Menu**: enables visualizations using ggplot2 such as bar charts, histograms, boxplots, and scatter plots.
- **Add-ons Menu**: lets you load additional R packages and tools. Each menu is supported by dialog boxes that guide the user through input selection and result interpretation.

7. Data Import and Export in BlueSky

Data can be easily imported from multiple formats including:CSV, Excel, SPSS (.sav), and native R datasets. The Import Wizard guides users step-by-step in selecting files, setting headers, delimiters, and variable types. Export options include saving data frames to CSV, Excel, or R Data formats. You can also export analysis results and R scripts for reproducibility or reporting. This makes BlueSky flexible for use in data cleaning, sharing, and publication-ready outputs.

8. Installing Plugins in BlueSky

BlueSky Statistics allows you to install additional R packages or plugins via its Add-on Module Manager. This enables you to extend the software's functionality beyond the default tools. You can add modules for advanced statistical models, machine learning, or specialized graphical analysis. Installation is simple just select the desired add-on from the list and click install. Once installed, the tools will appear under the appropriate menu. This plugin system helps tailor BlueSky to your specific analytical needs.

9. BlueSky Analysis Tools (Processing Toolbox)

The Analysis or Processing Toolbox in BlueSky includes a wide range of statistical tests and procedures accessible via menu. These include:

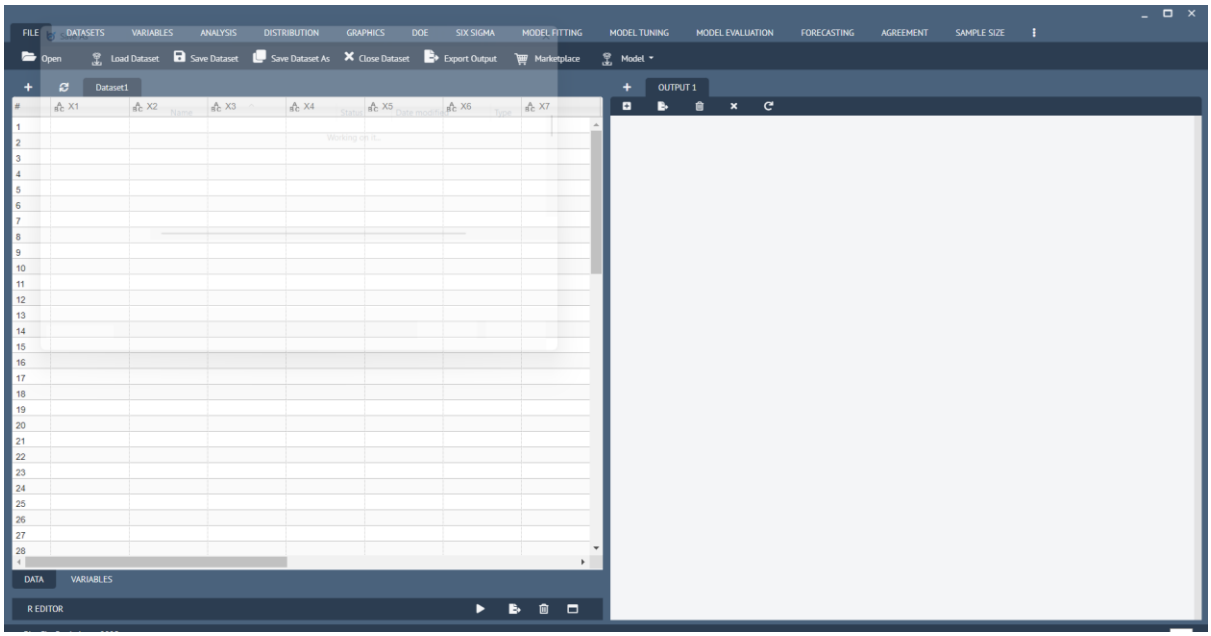
- Descriptive statistics
- Parametric/non-parametric tests (t-tests, ANOVA, chi-square)
- Regression models (linear, logistic)
- Time series analysis and correlation each tool is organized under intuitive dialog boxes with clear options for variable selection and test parameters. This toolbox helps users perform quick, accurate, and replicable analyses without writing code, while still allowing the R script to be saved or modified for advanced users.

10. Workspace Initialization in BlueSky Statistics

Description:

This is the default interface when BlueSky Statistics is launched. The left panel (Dataset

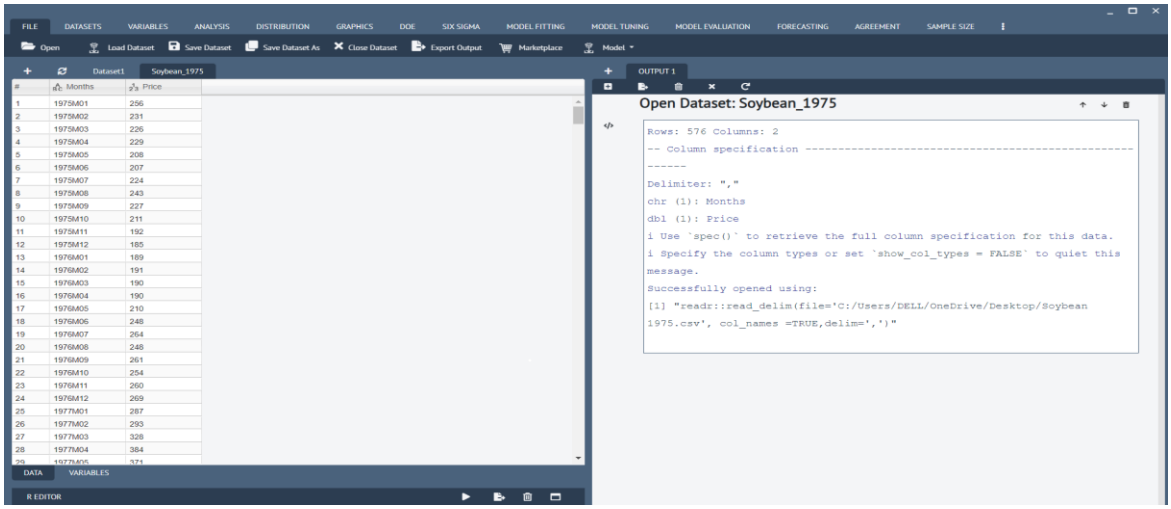
Window) allows for data input, either manually or through file import. The right panel (Output Window) is for displaying results, output text, and graphical summaries. The ribbon menu above offers categorized options such as DATASETS, ANALYSIS, FORECASTING, and more for executing statistical operations.



11. Dataset Import: Loading the Soybean_1975 File

Description:

This screen shows a CSV file (Soybean_1975.csv) successfully loaded into BlueSky Statistics. The dataset contains two columns: "Months" and "Price", with 576 rows. On the right, the output log confirms the data was read using the readr::read_delim() function from R, indicating successful data parsing with comma delimiters.



12. Display of Dataset: Soybean_1975

Description:

The left panel now shows the loaded data with a preview of monthly soybean prices from 1973 onwards. This tabular view is interactive, enabling users to inspect and prepare data before applying any statistical tests or models.

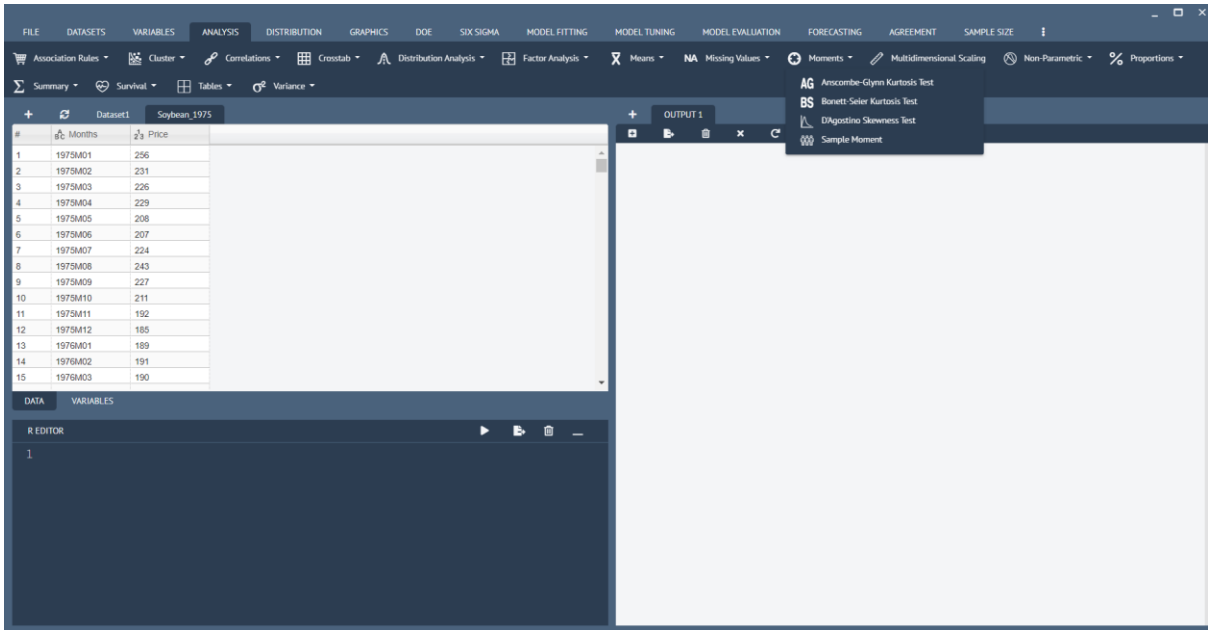
13. Moments Analysis: Skewness and Kurtosis Tests

Description:

Under the **ANALYSIS > Moments** tab, BlueSky offers various tests including:

- **Anscombe-Glynn Kurtosis Test**
- **Bonett-Seier Kurtosis Test**
- **D'Agostino Skewness Test**
- **Sample Moments**

These tests help assess distribution shape, asymmetry, and tailedness, providing a statistical basis for determining normality or identifying outliers.



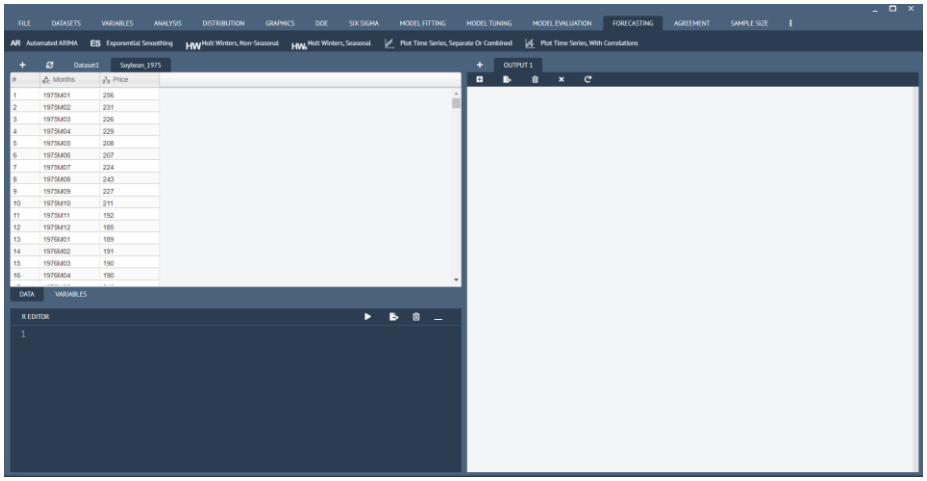
14. Time Series Forecasting Options

Description:

Under the FORECASTING tab, users can access a suite of tools including:

- ARIMA (Auto and Manual)
- Exponential Smoothing
- Holt-Winters Seasonal & Non-Seasonal models
- Time Series Plots with/without Correlation

These models support short- and long-term forecasting of price trends and are particularly useful in agroeconomic time series analysis.



15. Summary Statistics Panel

Description:

This section provides summary statistics (mean, median, variance, standard deviation, min/max values). These are essential for understanding the central tendency and dispersion in the dataset, and serve as the first step in exploratory data analysis.

16. Distribution Plot & Histogram View

Description:

A histogram and related distributional view (possibly with density plot overlay) helps visualize the shape of the price data. This is vital in determining whether parametric assumptions (like normality) are met before further statistical modeling.

Time Series Plot for Price Data

Description:

This graphical representation of the soybean price over time reveals trends, patterns, or cyclic behavior. Such visualizations are essential before applying ARIMA, Holt-Winters, or other time series models to understand underlying structures.

17. Result of D'Agostino Skewness and Sample Moments

Description:

Here, results from skewness and moment tests are shown. The D'Agostino test assesses

skewness (asymmetry), and the sample moments (mean, variance, skewness, kurtosis) provide a detailed quantitative summary of the price distribution.

Suggested Readings

Ashour, L. (2024). A review of user-friendly freely-available statistical analysis software for medical researchers and biostatisticians. *Research in Statistics*, 2(1), 2322630.

VassarStats: Website for Statistical Computation

Prabhat Kumar, Ponnaganti Navyasree, Nobin Chandra Paul, Bijoy Chanda, K. Ravi Kumar and Santosha Rathod
ICAR-National Institute of Abiotic Stress Management, Baramati, Pune-413115
Email: prabhatkkv@gmail.com

1. Introduction to VassarStats:

VassarStats is a free, web-based statistical computation toolbox developed by Richard Lowry of Vassar College. It provides easy-to-use calculators for a wide range of statistical analyses, including t-tests, ANOVAs, correlation, regression, and probability functions. VassarStats: is a user-friendly statistical software developed specifically for analyzing agricultural field experiment data. It is widely used in Indian agricultural universities for conducting ANOVA, CD (Critical Difference) calculations, and other design-based analyses. The software supports various experimental designs such as RCBD, Split Plot, Factorial, and more. It is tailored to meet the needs of agronomists, soil scientists, and plant breeders. The interface is simple and ideal for students, researchers, and faculty in the agricultural sciences.

2. What is VassarStats:?

VassarStats: is an offline statistical tool often provided by agricultural institutions or bundled with textbooks. It is commonly used for performing design-based statistical analysis like Randomized Block Design (RBD), Latin Square, Split-Plot, and other layouts used in agricultural research. Unlike general statistical software, VassarStats: focuses on predefined templates for experiment designs and outputs in tabular formats commonly used in journals. It is straightforward and suited for practical classes and research theses.

3. Graphical User Interface of VassarStats

The GUI of VassarStats: is simple and Excel-style, often built using MS Access or Excel macros. It opens with menus for selecting experiment designs and entering data in pre-set columns. The layout includes:

- A design selection window
- Data entry grids for replications, treatments, blocks, etc.

- Result windows displaying ANOVA tables and CD values
  The user interface is limited but functional, designed for specific statistical tasks rather than flexible analysis or custom modeling.

4. Download and Installation

VassarStats: is not publicly available online for direct download. It is generally distributed through agricultural universities, especially during practical classes in BSc/MSc programs. You can request it from your statistics or agronomy department, or obtain it from academic resources, CDs bundled with books, or peer sharing. The software is light, usually runs on Windows, and requires no internet or advanced system requirements. It may be installed via a .exe file or opened directly from an Excel macro file.

5. Data Types in VassarStats:

VassarStats: uses structured data types suitable for field experiments:

- **Replications**: Numeric (usually fixed blocks or seasons)
- **Treatments**: Coded alphabetically or numerically
- **Observations**: Continuous numeric data (e.g., yield, plant height)
- **Factors**: Often input via dropdowns or codes
  It does not support advanced data types like text strings or dates. The system expects well-structured, clean datasets, usually small in size and formatted based on the design selected.

6. Key Menu Options and Tools in VassarStats

The menu structure in VassarStats is focused on experimental designs. Major menu items include:

- **Design Selection**: Choose from RCBD, Split-Plot, Latin Square, etc.
- **Data Entry**: Manual input into structured templates
- **Analysis Execution**: Generate ANOVA tables, SE(m), CD, and CV

- **Output Report**: View and print journal-style results
  There is limited flexibility for custom analysis or scripting, but the software provides accurate and fast results tailored to the selected design.

7. Data Import and Export in VassarStats

VESSAR does not support direct data import/export from Excel or CSV files. Instead, users are required to manually enter data into pre-designed templates inside the software. Results can be exported by copying and pasting from the result window or printing directly. Some versions allow saving outputs as PDF or DOC files, depending on the environment (e.g., Excel-based macros). For larger datasets, data entry must be done carefully, as import automation is generally not supported.

8. Installing Plugins in VassarStats

VassarStats: does not support plugin or module installation like modern software. It is a standalone tool with fixed functionalities focused on field experiment analysis. However, some institutions may provide updated versions with new designs or slight UI improvements. Users cannot add features, statistical tests, or R packages as in other tools. For advanced statistical modeling, users may need to shift to R, SPSS, or BlueSky. VESSAR is meant to be a simple and stable environment for teaching and routine agricultural analysis.

9. Adding Raster Layers in VassarStats

VassarStats does not support raster or spatial layers, as it is not a GIS tool. It is designed for tabular data from agricultural field trials, not remote sensing or geospatial analysis. Users needing to handle raster data (e.g., satellite images or weather maps) must use QGIS, ArcGIS, or R (using raster/terra packages). VESSAR is best used for yield data, treatment effects, and factorial experiments, without spatial mapping capabilities.

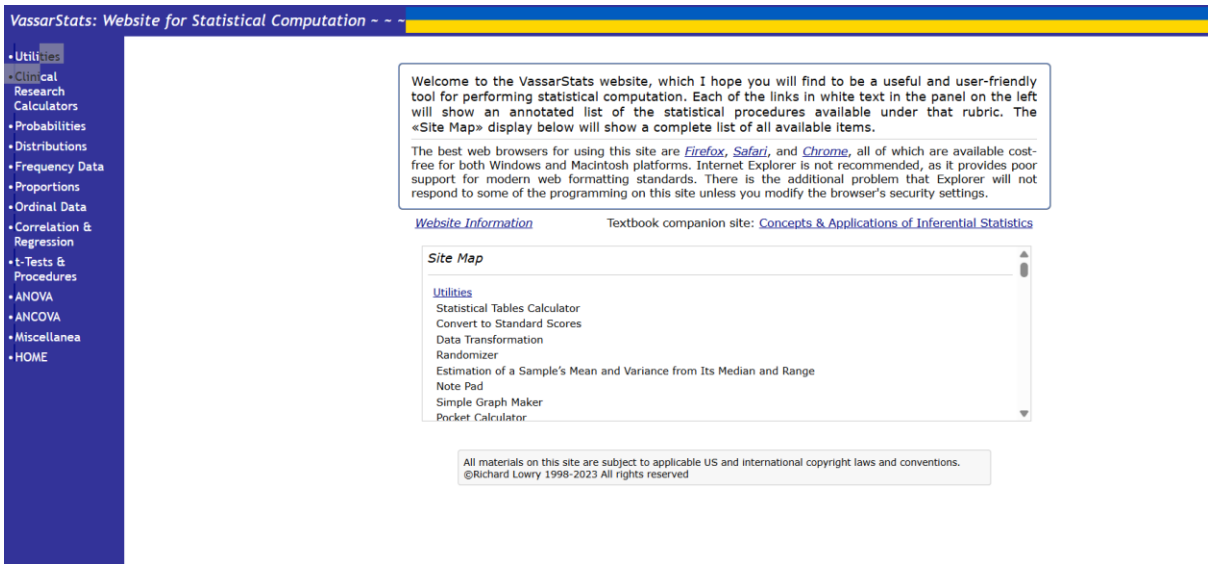
10. VassarStats Analysis Tools (Processing Toolbox)

VassarStats analysis toolbox includes core tools such as:

- One-way and two-way ANOVA
- Completely Randomized and Randomized Block Designs

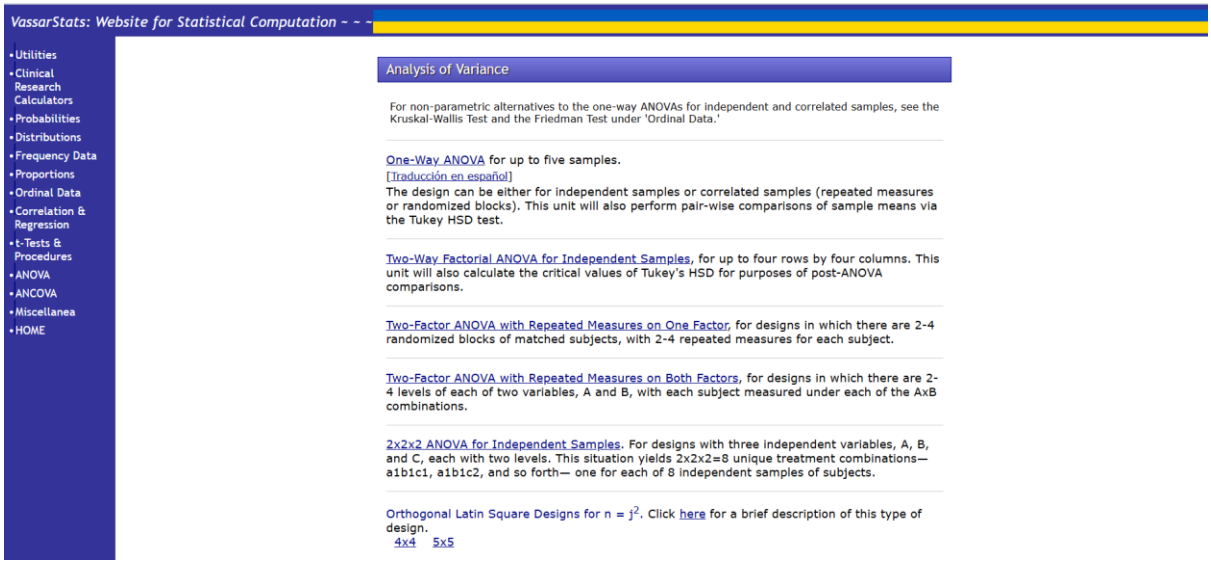
- Split-Plot and Factorial Analysis
- Critical Difference (CD), SE(d), SE(m), CV%

  These tools are automated and produce results in tables formatted for academic use. No coding is needed; analysis is triggered with one click after data entry. This makes the software ideal for quick and accurate statistical analysis in experimental agriculture.



11. Website Structure and Navigation Panel

- The homepage displays a navigation panel on the left side.
- Key categories include: *Utilities, Clinical Research Calculators, Probabilities, Distributions, ANOVA, t-Tests,* etc.
- The center of the page provides a Site Map, listing available tools like Data Transformation, Statistical Tables, Randomizer, etc.



Analysis of Variance (ANOVA)

- VassarStats supports different ANOVA types, such as:
    - **One-Way ANOVA**: For comparing means across multiple groups.
    - **Two-Way ANOVA (**with or without repeated measures).
    - **Factorial ANOVA**: Useful for experiments with multiple factors.
- Each method comes with an explanation and links to Tukey HSD post-tests.

t-Tests and Confidence Intervals

- This section offers tools for:
    - **Two-sample t-tests** for independent or correlated samples.
    - **Single-sample t-test**.
    - **Confidence Intervals** (.95 and .99) for population means.
- These tests help determine statistical significance between groups or samples.

Correlation and Regression

- This part includes:
    - Basic Linear Regression and Correlation
    - Multiple Regression
    - Partial, Rank, and Point Biserial Correlations
    - Logistic Regression

Also includes import version for large datasets

Comparisons tools

Feature	Bluesky Statistics	VassarStats:
Backend	R-based	Possibly proprietary or Excel-based
Cost	Free, Open Source	Institutional (may not be free)
GUI	Yes	Yes
Focus Area	General statistics	Field experiments in agriculture
ANOVA Support	Yes	Yes (RCBD, Split Plot, etc.)
Customization / Scripting	Yes (via R code)	No
Graphical Output	Yes (ggplot2 integration)	Limited or basic

Suggested Readings

Lowry, R. (2004). *VassarStats: website for statistical computation*. Vassar College.

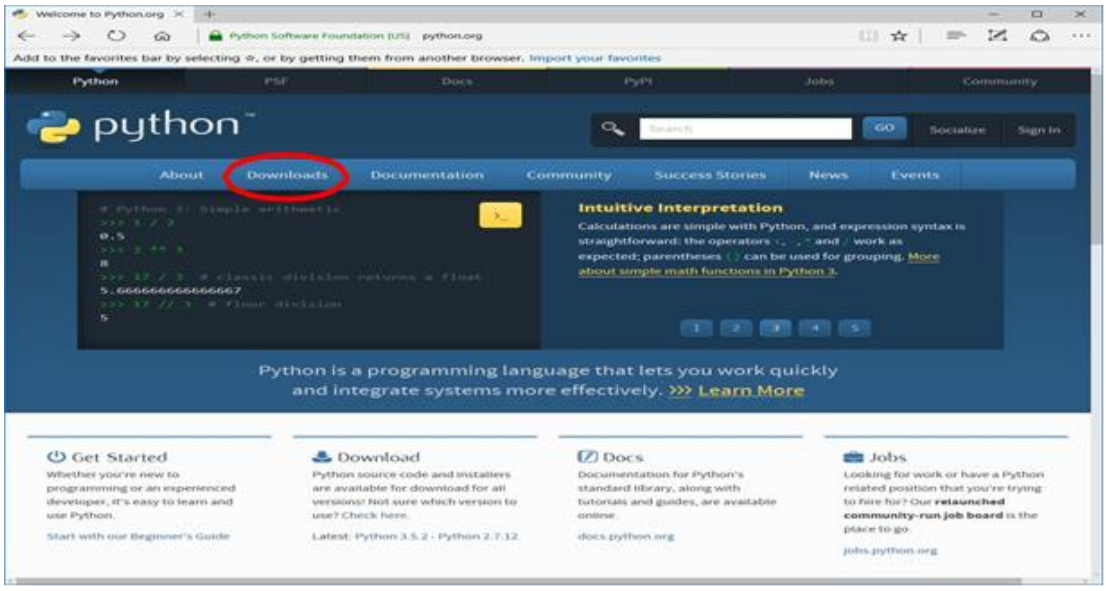
http://vassarstats.net/

Installation Guide to Python

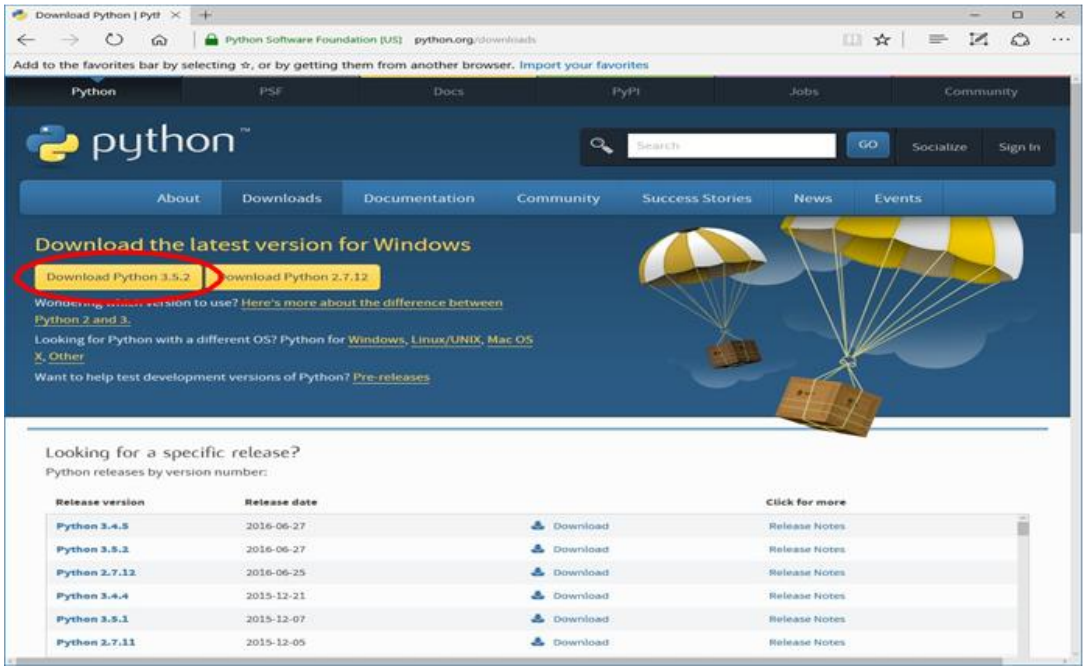
Arun Kumar
KSR College of Engineering, Kadapa, A.P.
Email: arunkumar.mtech09@gmail.com

Installing Python on Windows 10:

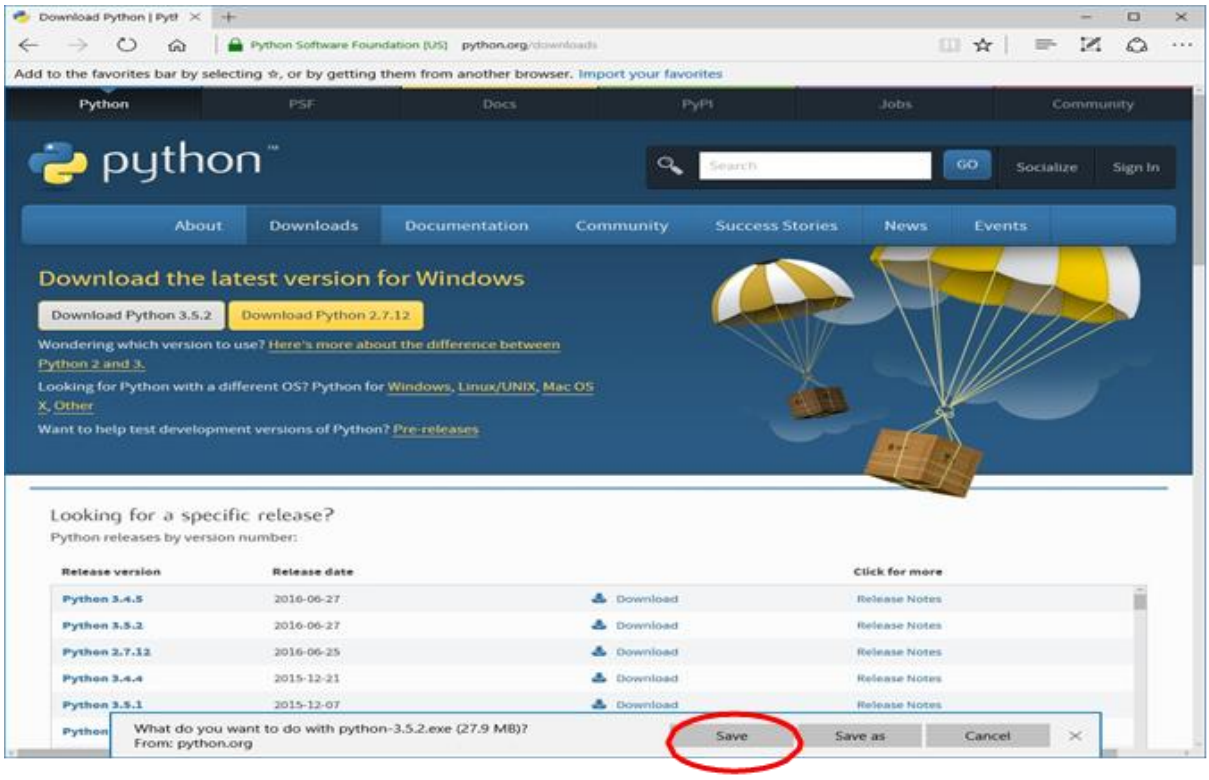
- ✓ Go to www.python.org
- ✓ Click "Downloads" Link at the top of the page

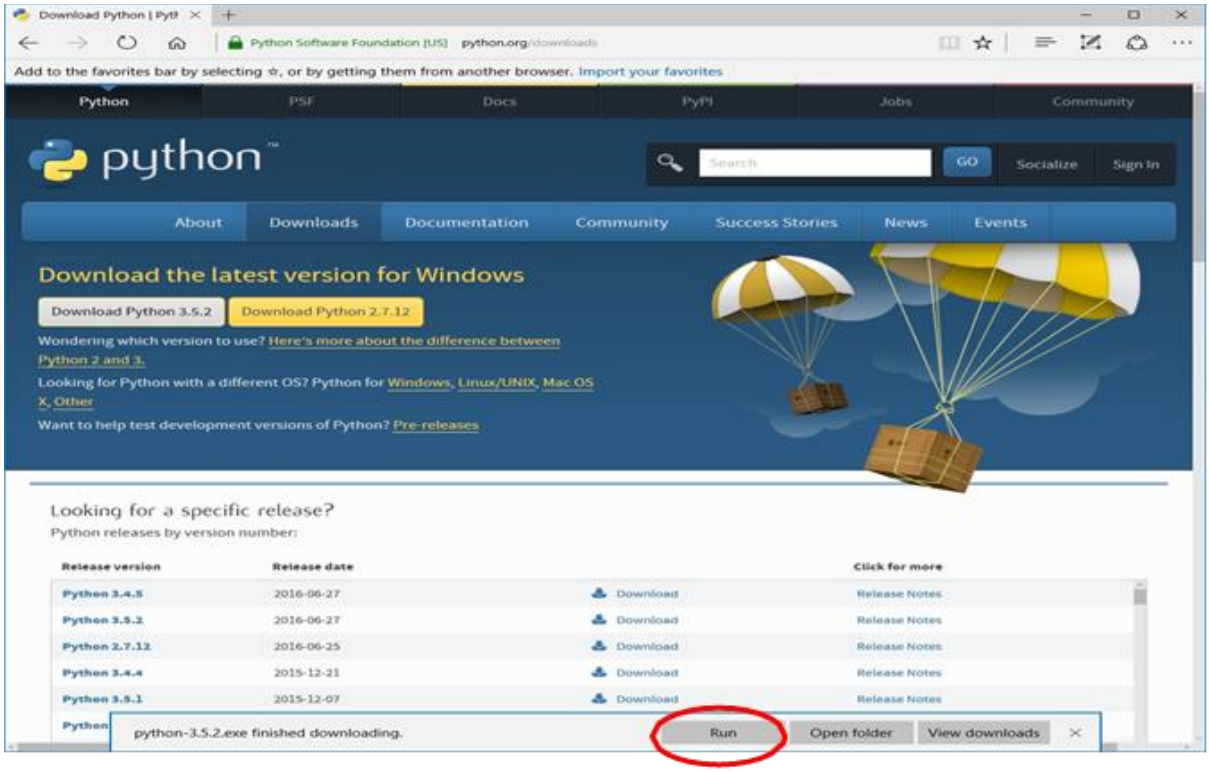


Click "Download Python3.5.2" (or whatever the 3.X version currently is:



When it asks you, “What do you want to do with python-3.5.2.exe(27.9MB)?”, click “Save” then“Run”





When the installation window comes up, click "Install Now"

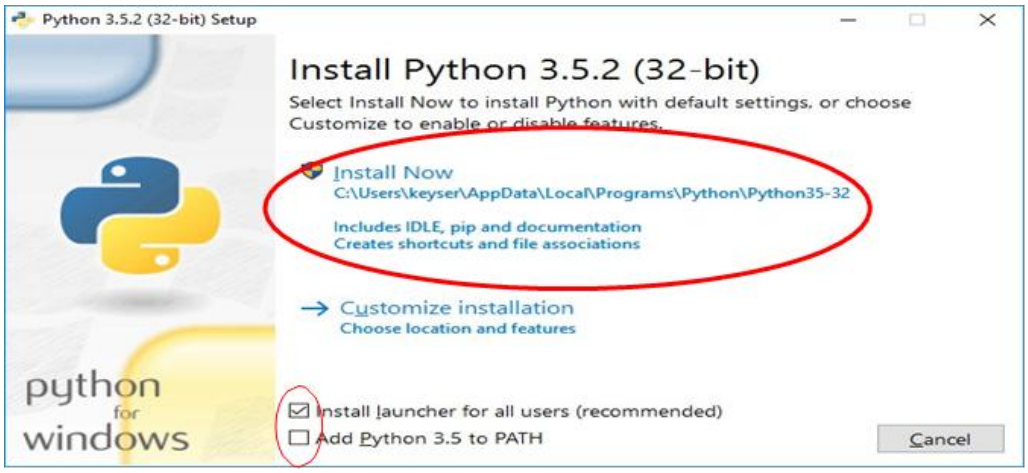
a. You can choose to check the "Install launcher for all users (recommended) or not– either way should be OK

b. You can choose to "Add Python 3.5 to PATH" or not–either way should be OK

c. Note: Depending on how Windows is set up, you might need to provide an administrator password to install on your system at this point.

d. You can choose to "Customize Installation" if you want, especially if you want to install to a location other than the default tone shown. Generally, I recommend installing to the default location unless you have a problem doing so.

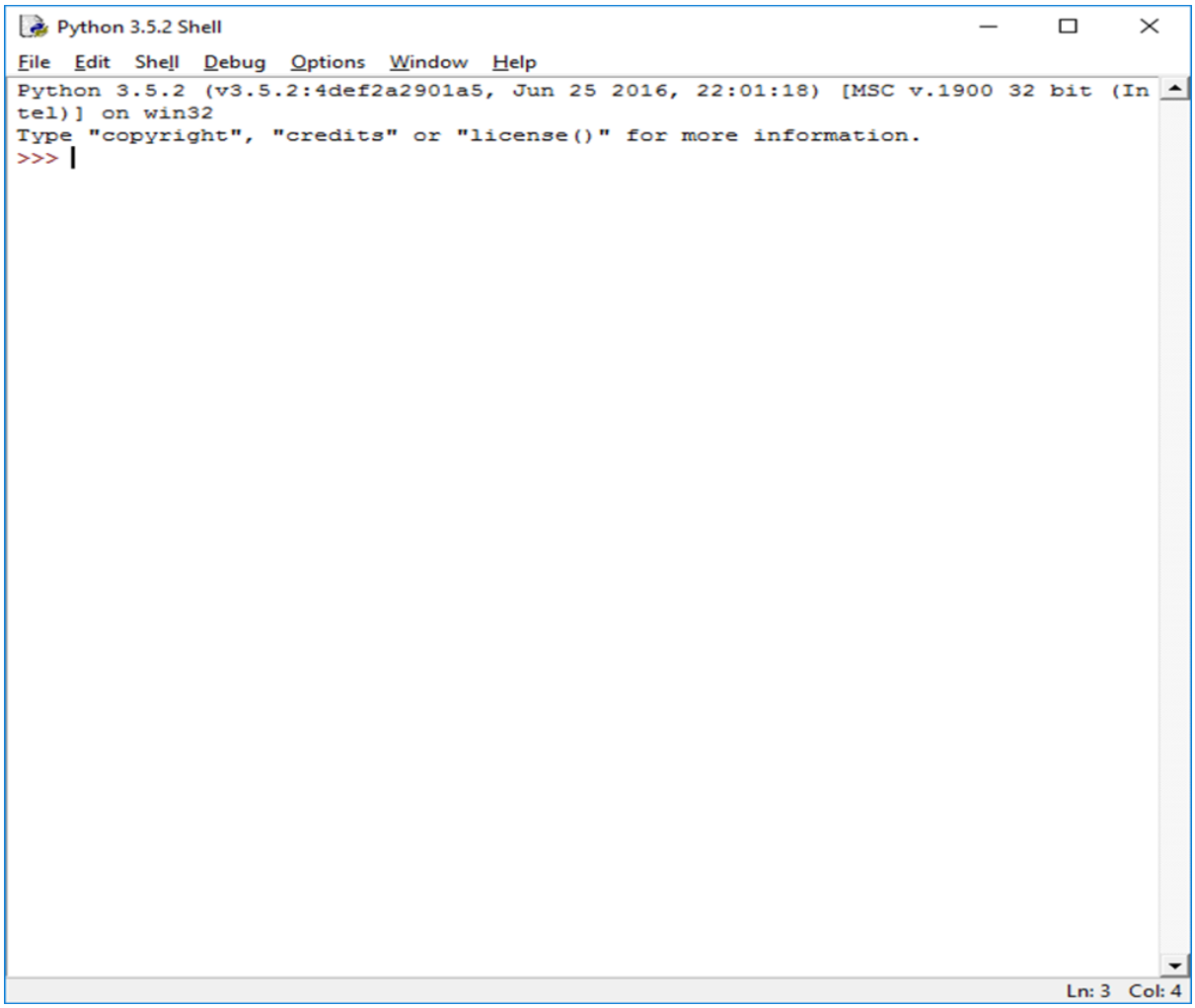
e. In any case, you might want to note the location of the installation in case you have difficulty later. If you are specifying the location yourself, put it in a location you are likely to easily find/remember.



You should see Python installing at this point. When it finishes, you should see a screen that says the installation was successful. You can click "Close"

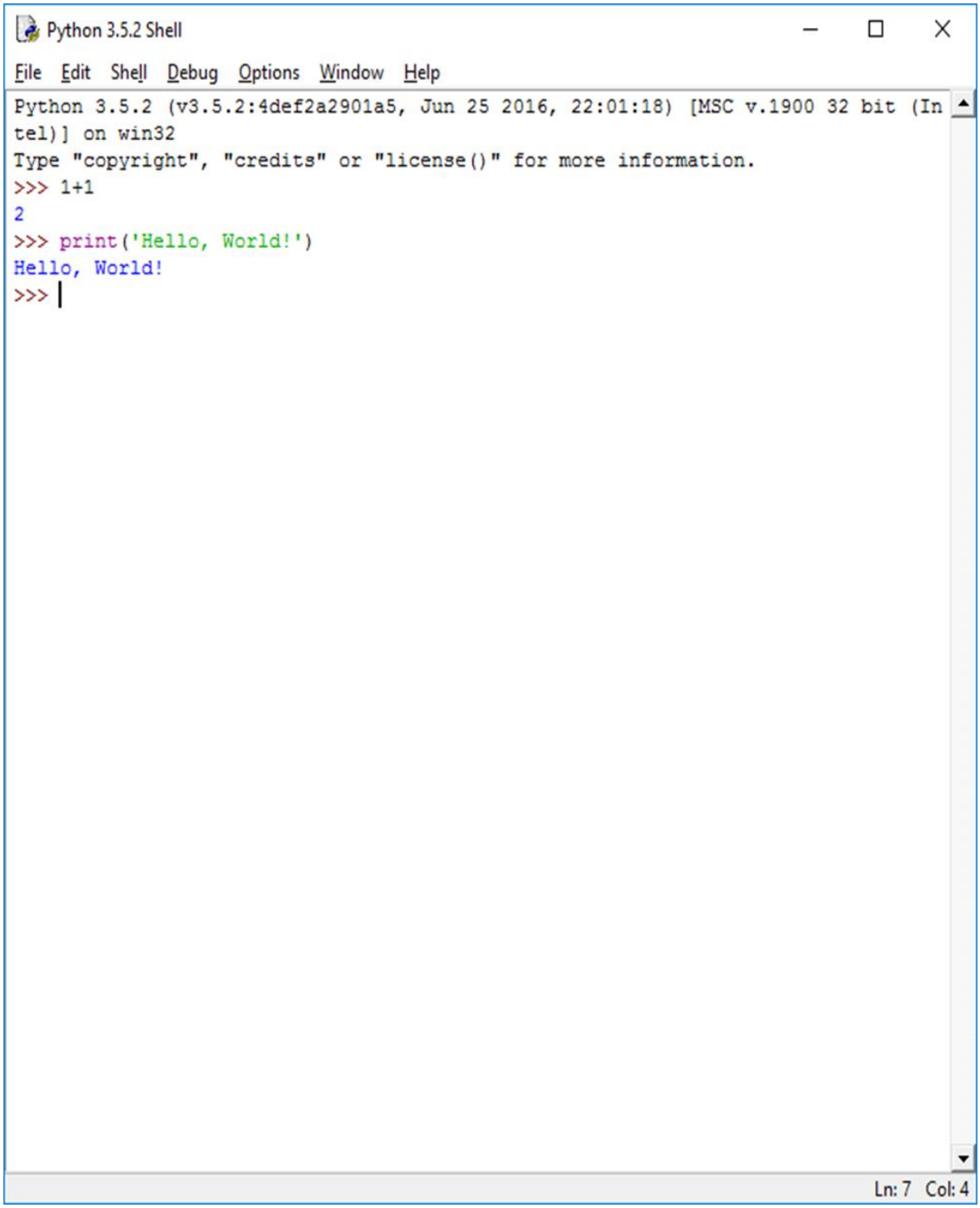
Running Python: Now you have installed Python. That means you will be able to run Python files and use the IDLE Python editor on your computer. If you want to test this quickly, you can do the following:

- ✓ Go to the text box at the lower left of your Windows desktop and type in “IDLE”. You should see a program come up from the search that says “IDLE (Python 3.5 32-bit) Desktop app” show up. Click on this.
- ✓ You should now have the “IDLE” Python shell, that looks like this:



Try typing in a short python command, like:

- ✓ A math operation : 1+1
- ✓ A print statement :print (‘Hello,World!’) It should give you the result:

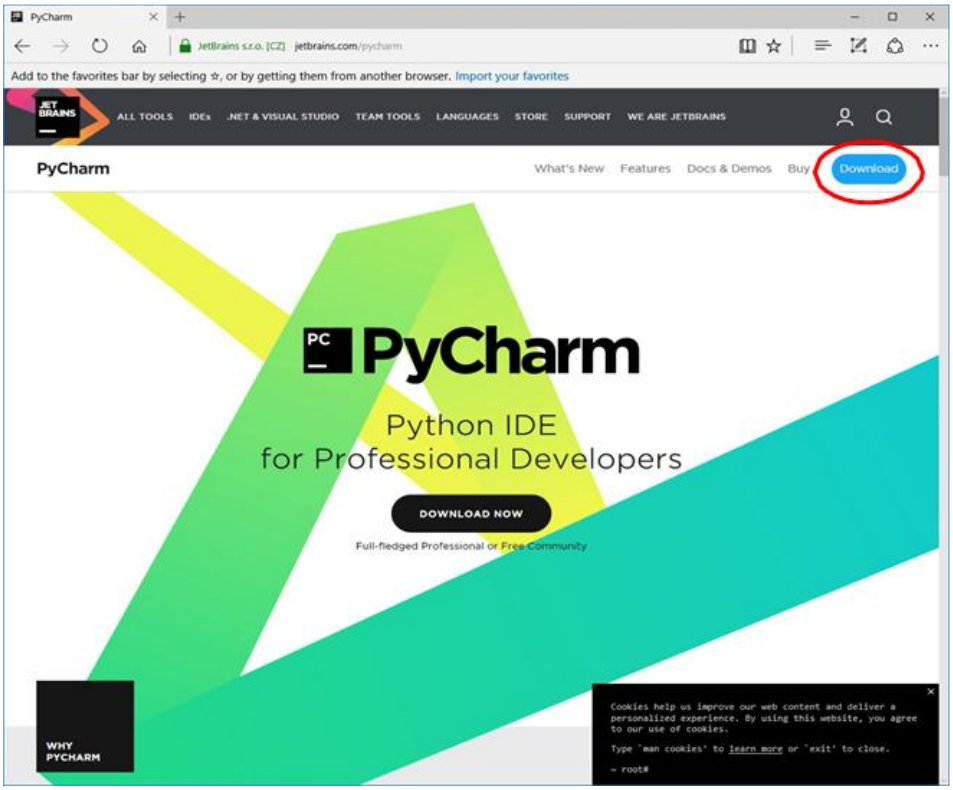


Now you are ready to install the PyCharm Integrated Development Environment (IDE).

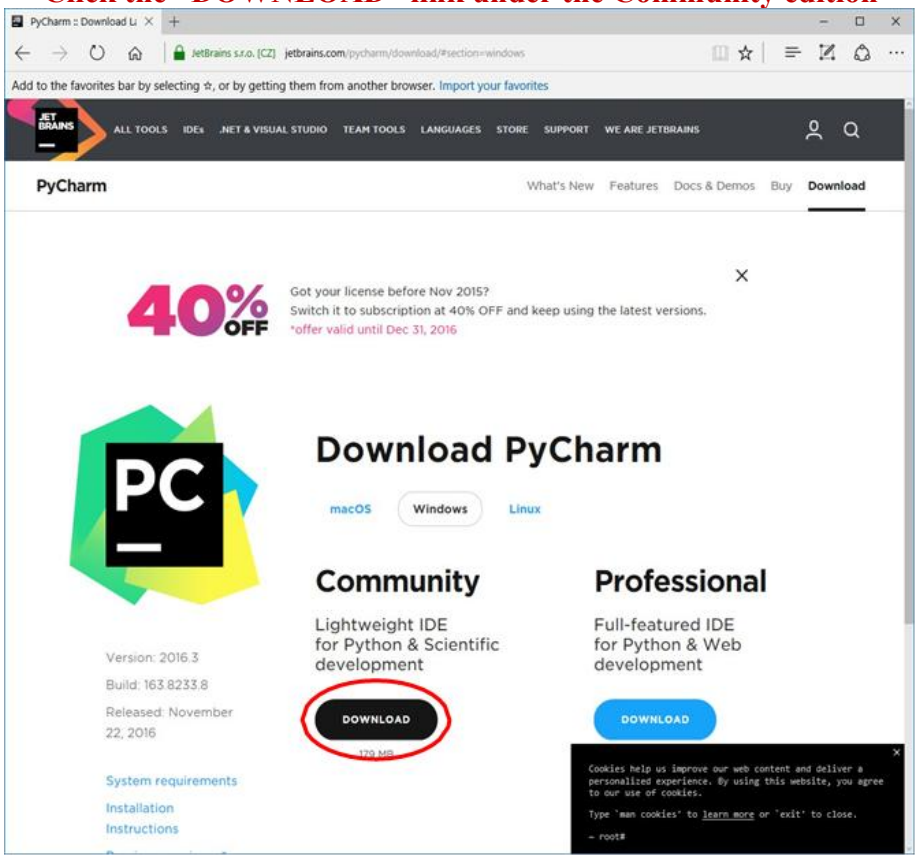
Installing PyCharm

a. Gotowww.jetbrains.com/pycharm

b. Click the “Download” button at top left (you can also scroll down the page and click the download link for the community version directly)

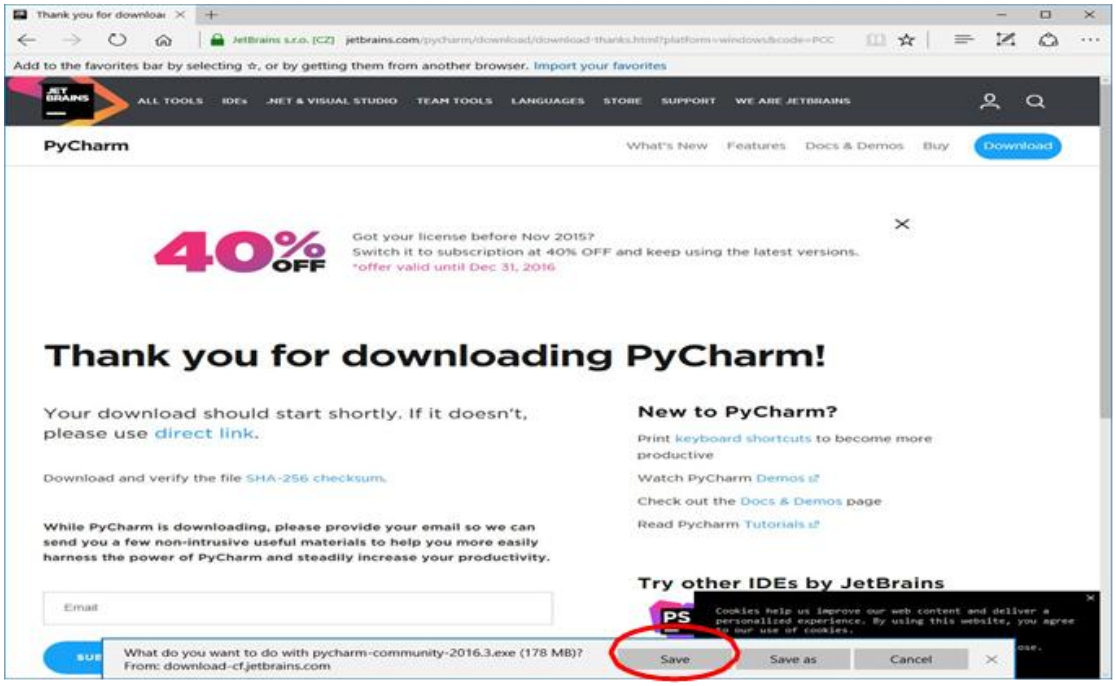


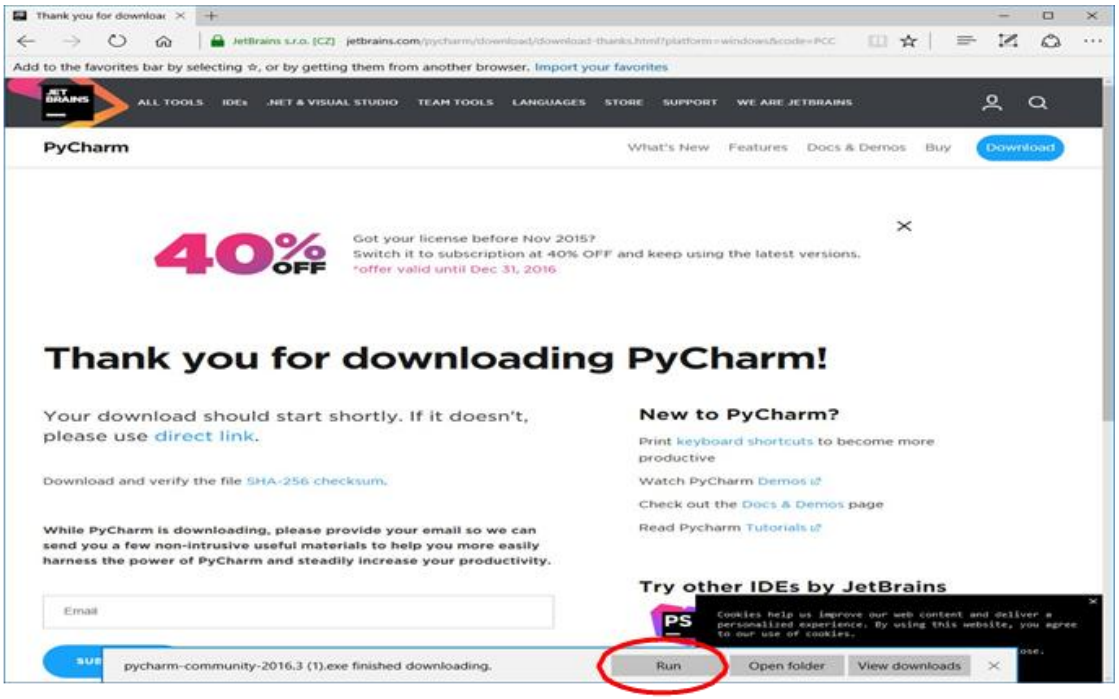
Click the "DOWNLOAD" link under the Community edition



When the download box pops up, click “Save” and then “Run”.

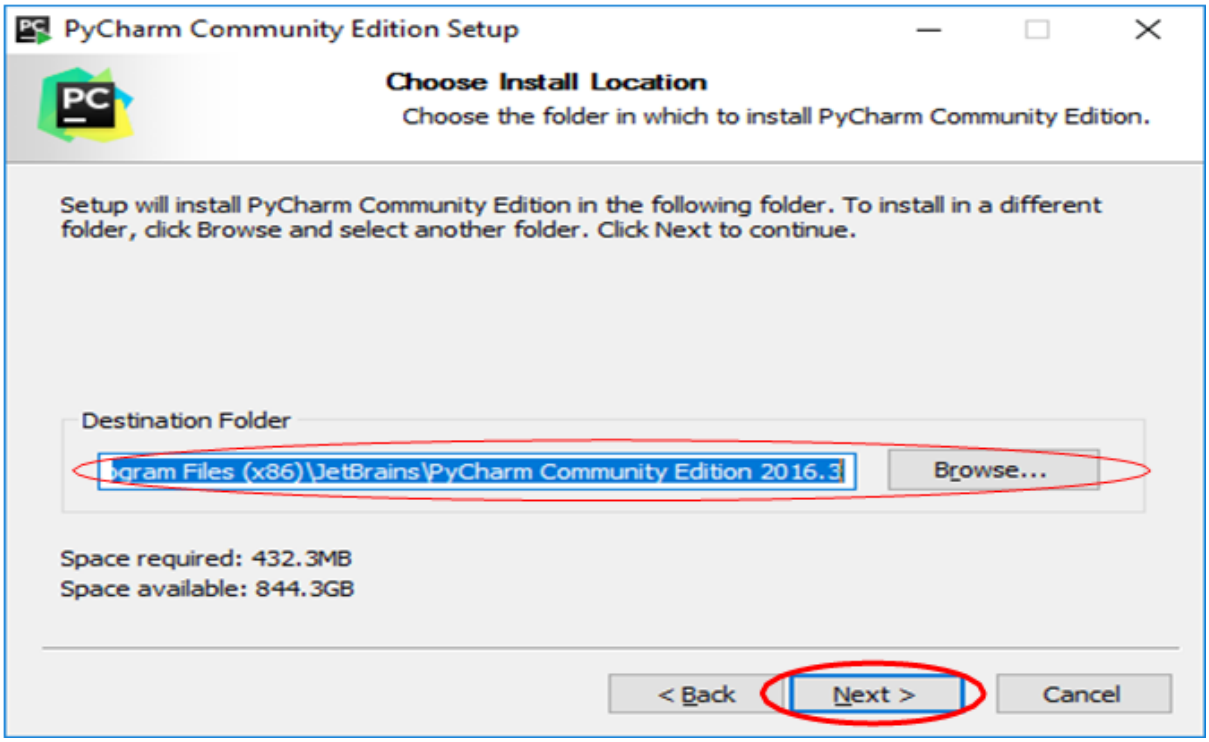
- ✓ Depending on your computer’s setup, you might need to enter an administrator password at this point.





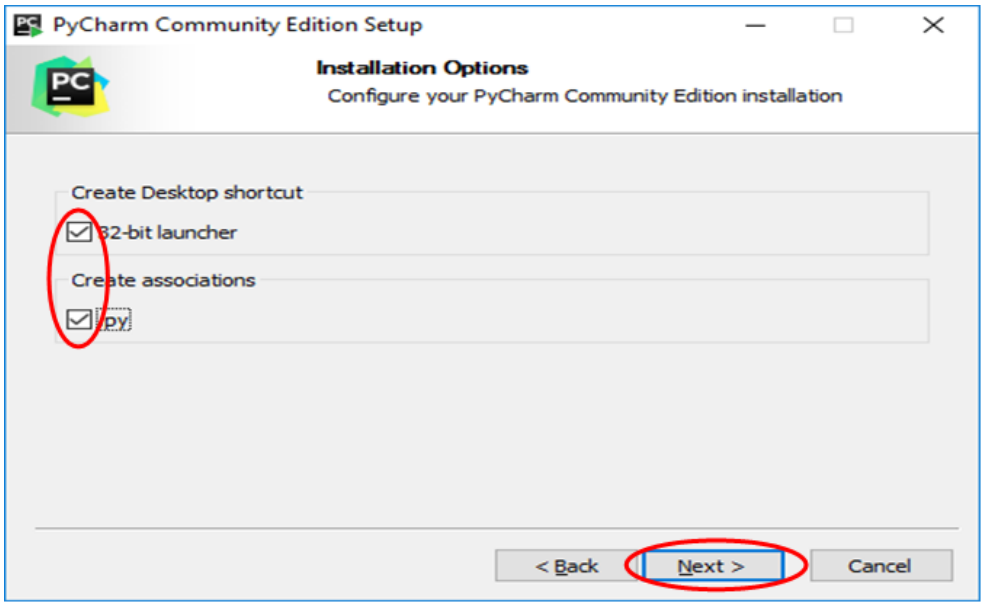
The setup wizard should have started. Click “Next”

Choose an installation location. I recommend letting it install in the suggested location. Click Browse if you want to enter a new location, and when you are done selecting the location, click “Next”.

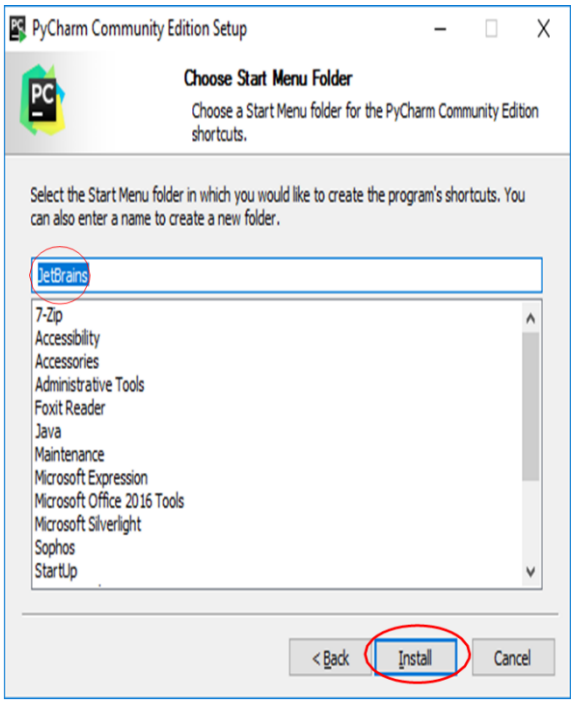


You can choose to select two options, after which you click “Next”

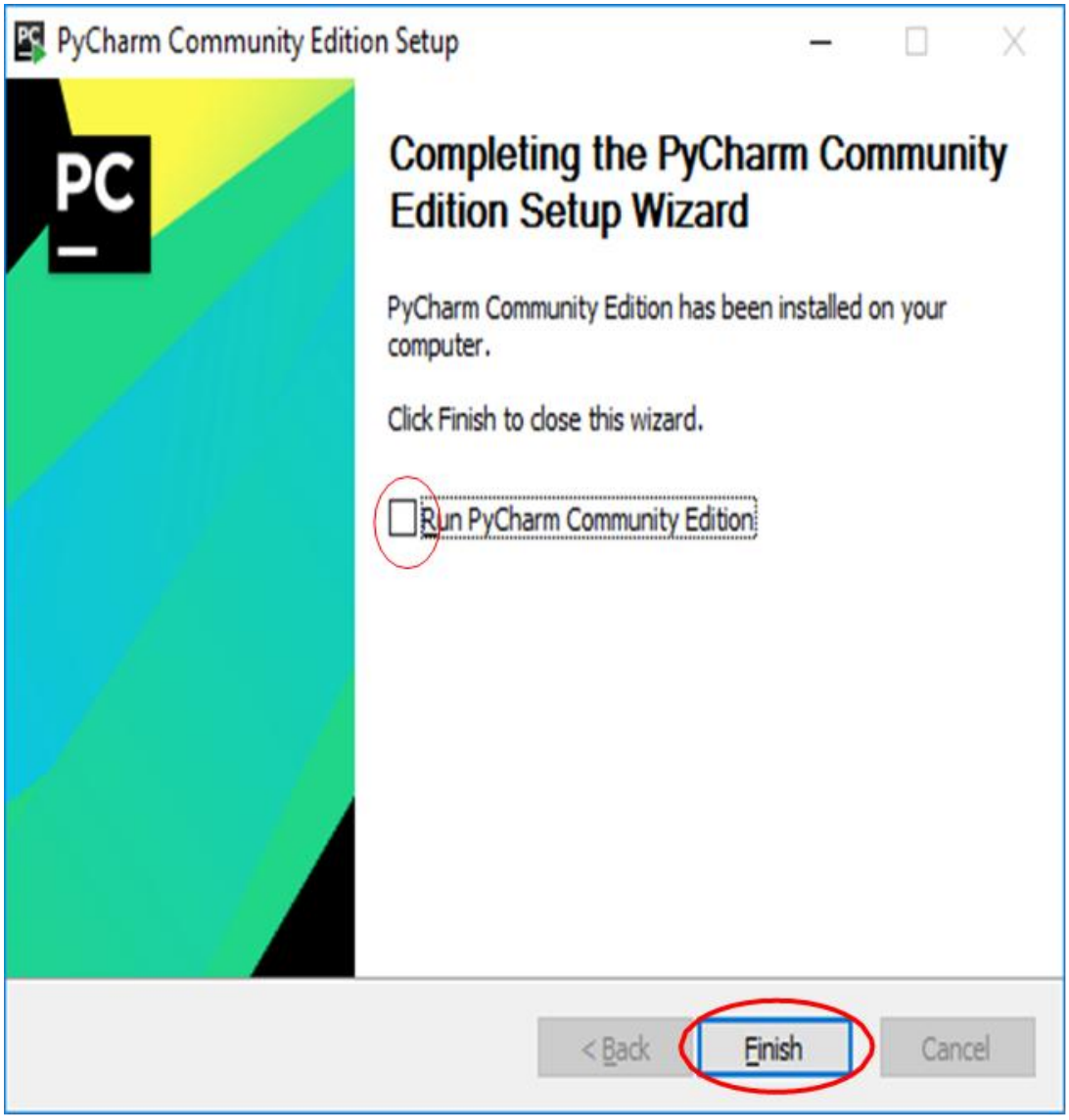
- ✓ You can create a desktop shortcut (I suggest doing so). This will put a link to PyCharm on your desktop.
- ✓ You can create associations (I suggest doing so). This means that when you open a python file (one with a .py extension), it will open in PyCharm.



Choose a start menu folder (I suggest leaving the default “Jetbrains” selected, but you can change it if you wish) and click “Install”

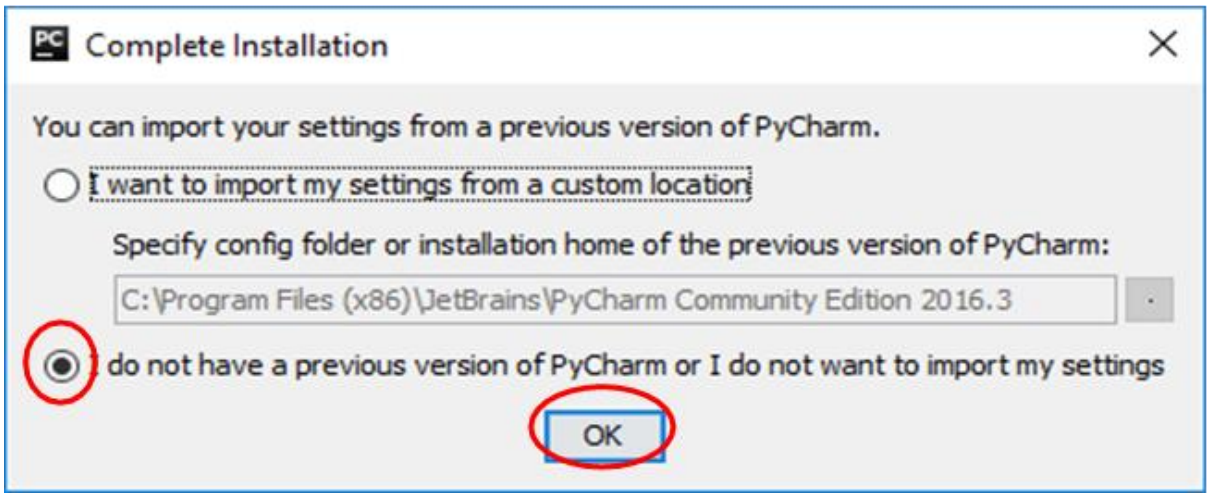


Wait for installation to finish. At the end, you should receive a message screen that PyCharm is installed. You can click “Finish”, and if you want to go ahead and run it, click the “Run PyCharm Community Edition” box first. You have now installed PyCharm.



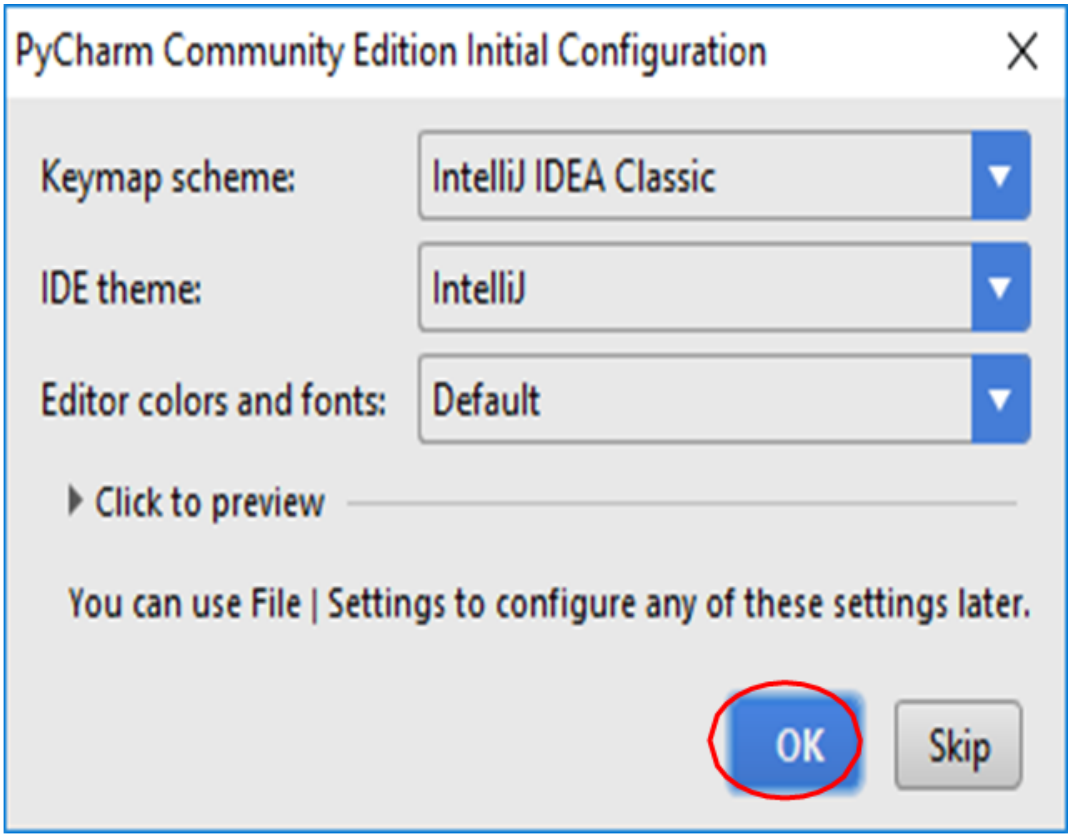
Running PyCharm

- ✓ The first time your PyCharm,you will get a message box asking about importing settings. You can select “I do not have a previous version of PyCharm or I do not want to import my settings” and click “OK”

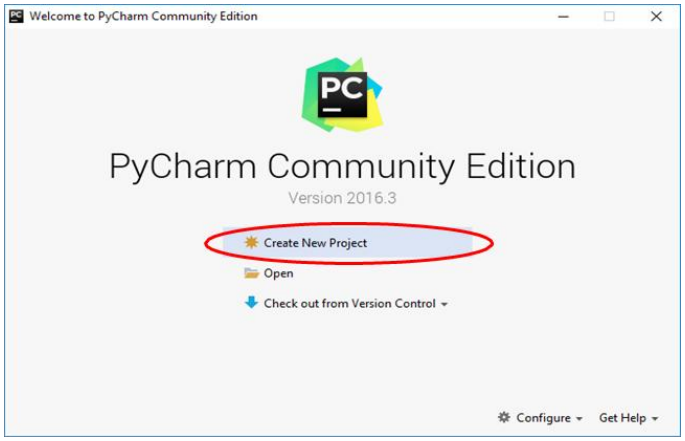


The first time you run PyCharm, you will need to accept the privacy policy.

The first time you run PyCharm, you will have some “Initial Configuration” options. Just hit “OK”



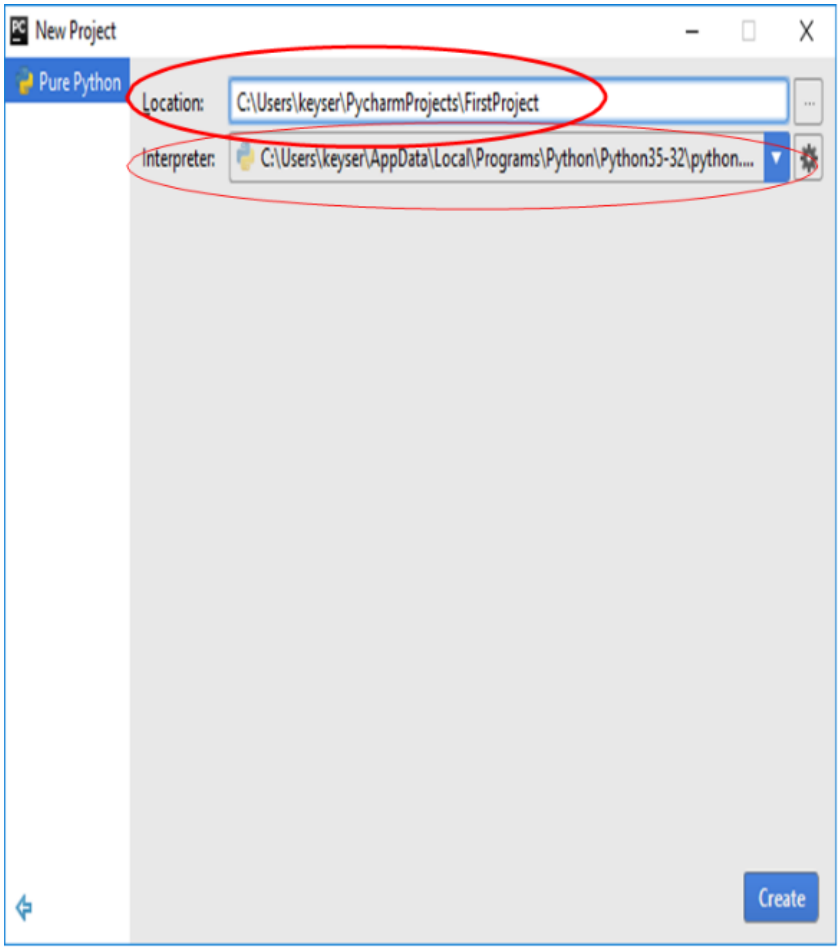
You will now be at the intro screen for PyCharm. This will also be the intro screen when you start PyCharm in the future without a recently opened project (if you do have a recently opened one you did not close before exiting, it will automatically reopen to that one).Click on "Create New Project"



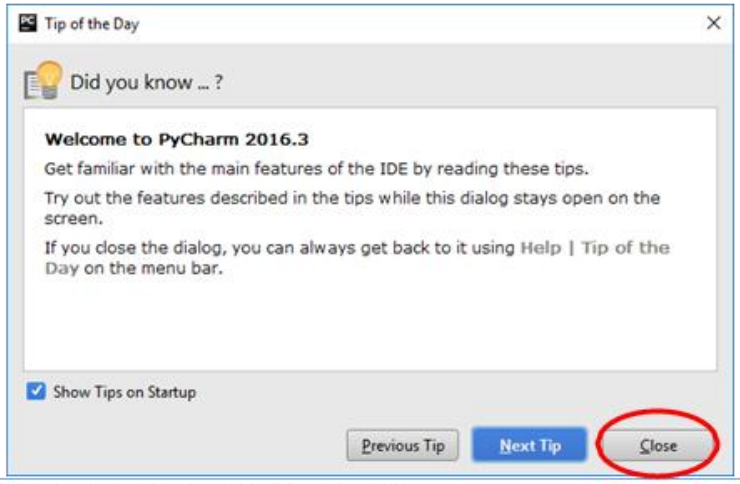
You will need to select a location.

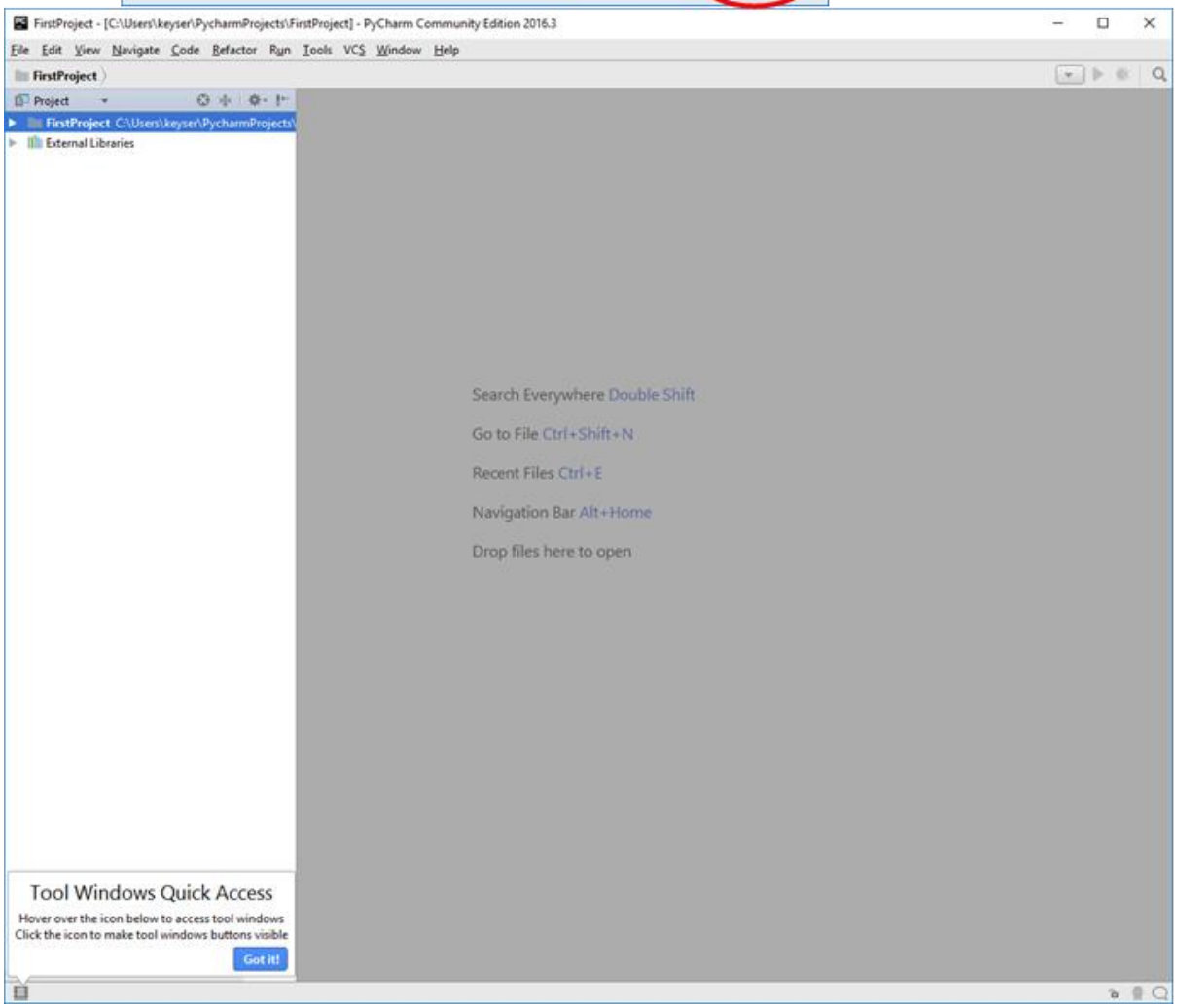
- ✓ When it asks for the location, you can select where you want the project to be created. I suggest at least changing the name from "untitled" to something more meaningful, like "First Project"

- ✓ PyCharm should have found the Python interpreter you installed earlier (when you installed Python itself). It will be selected if so. See TROUBLESHOOTING below for what to do if you do NOT have an Interpreter listed. If you donot have an interpreter listed, you need to resolve this before you can continue.
- ✓ Click the "Create" Button when you have the project setup

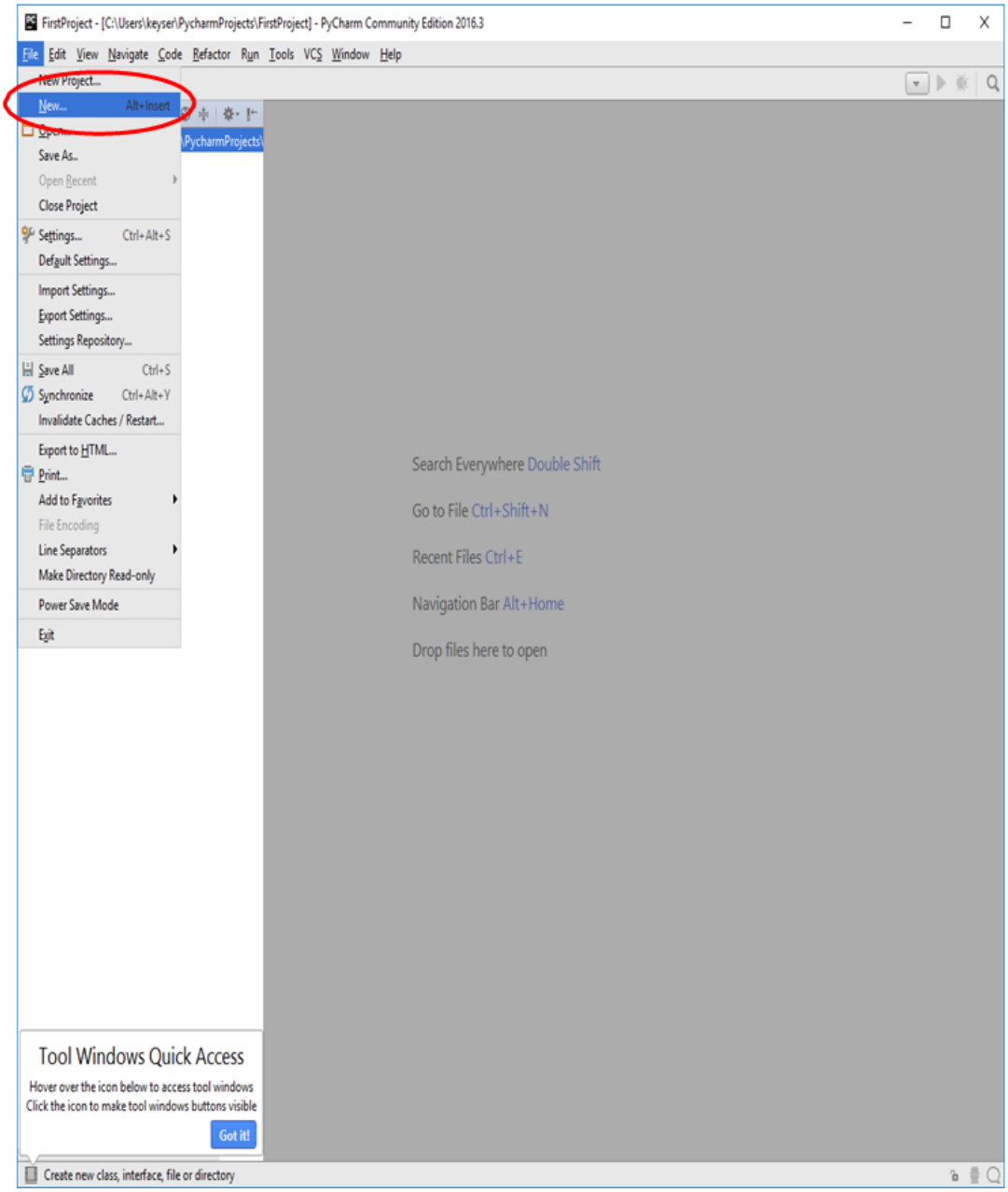


This will bring up the PyCharm environment (you can close the "Tip of the Day") box that pops up

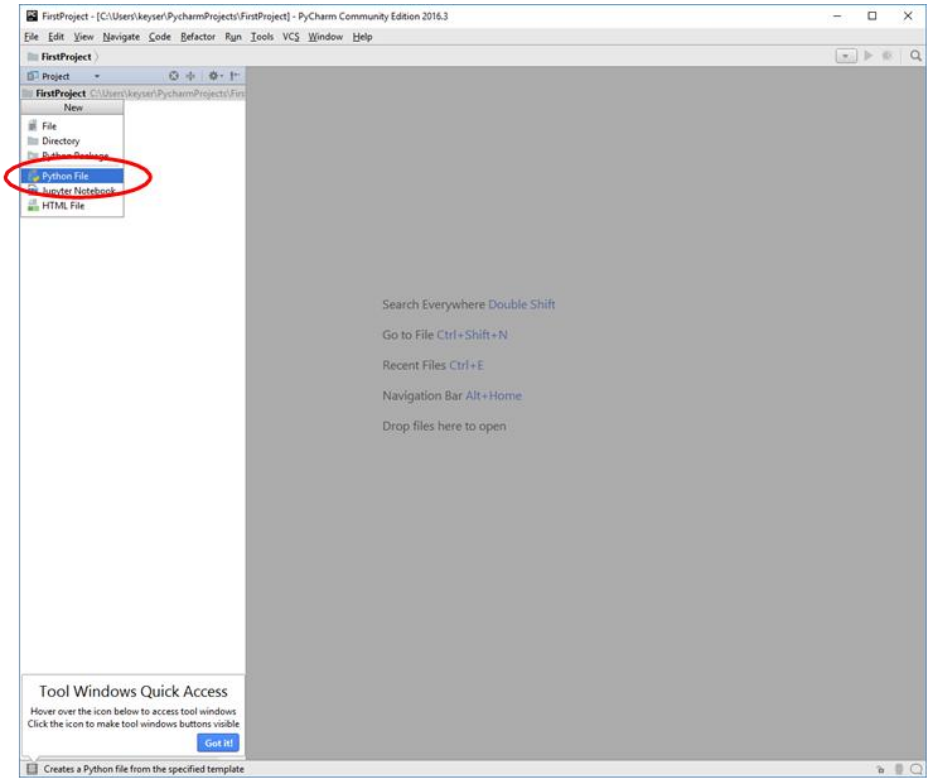




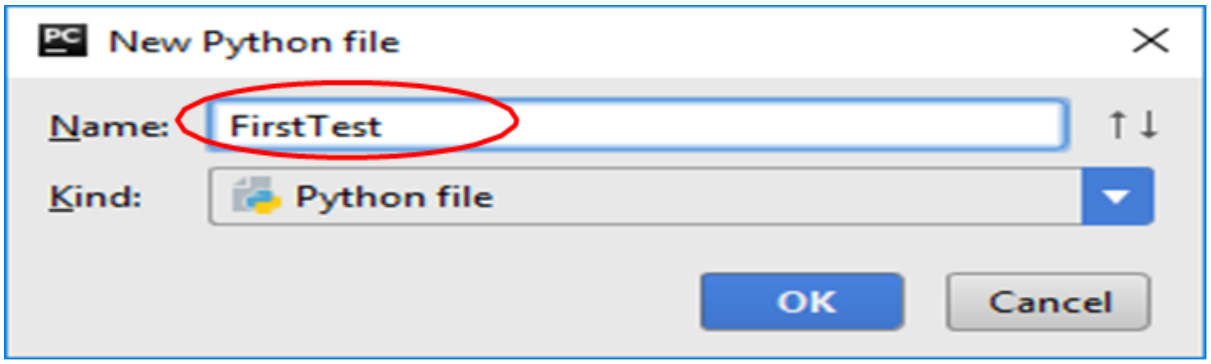
Try creating and running a program. Go upto the "File" menu and select "New"



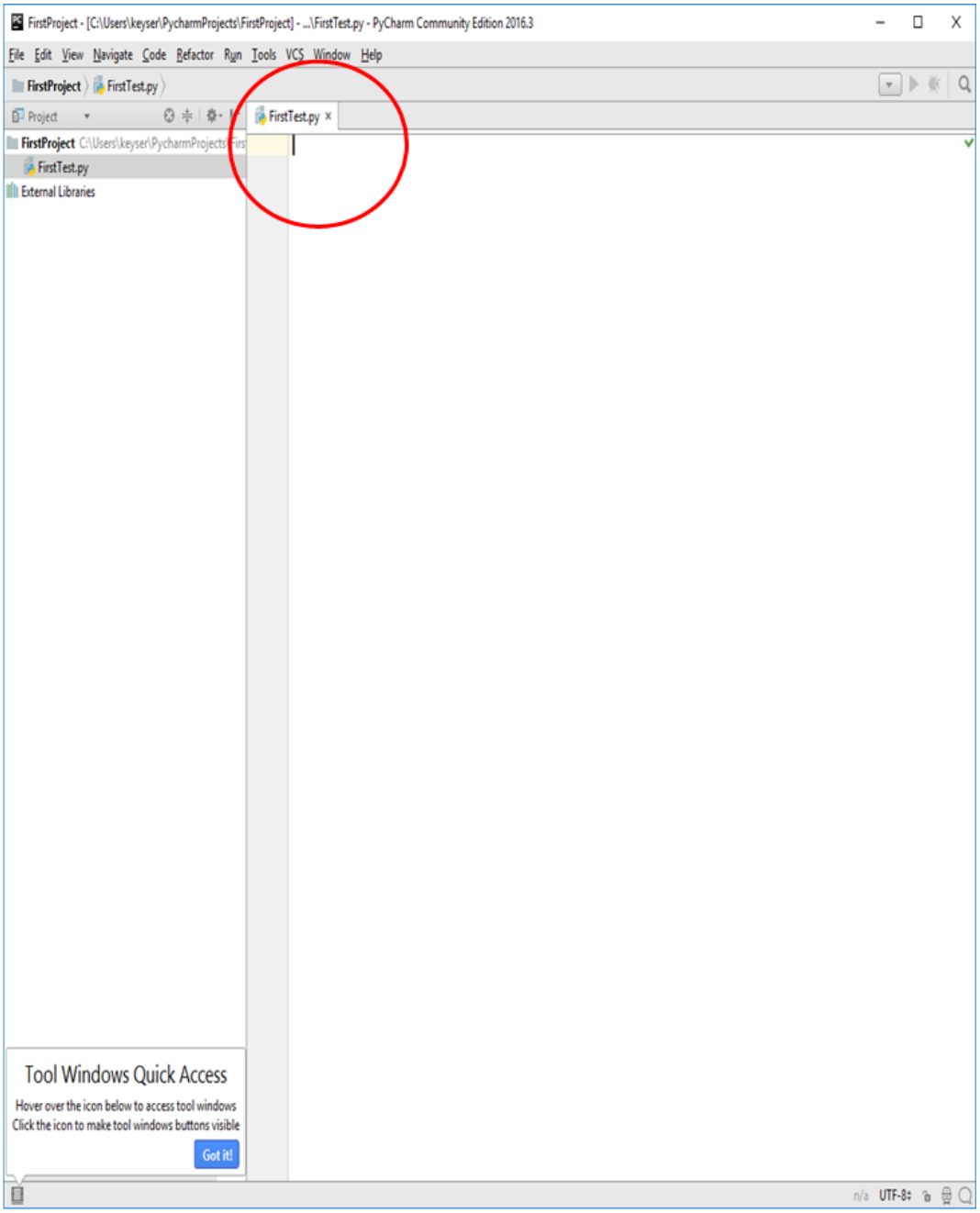
Select "Python File"



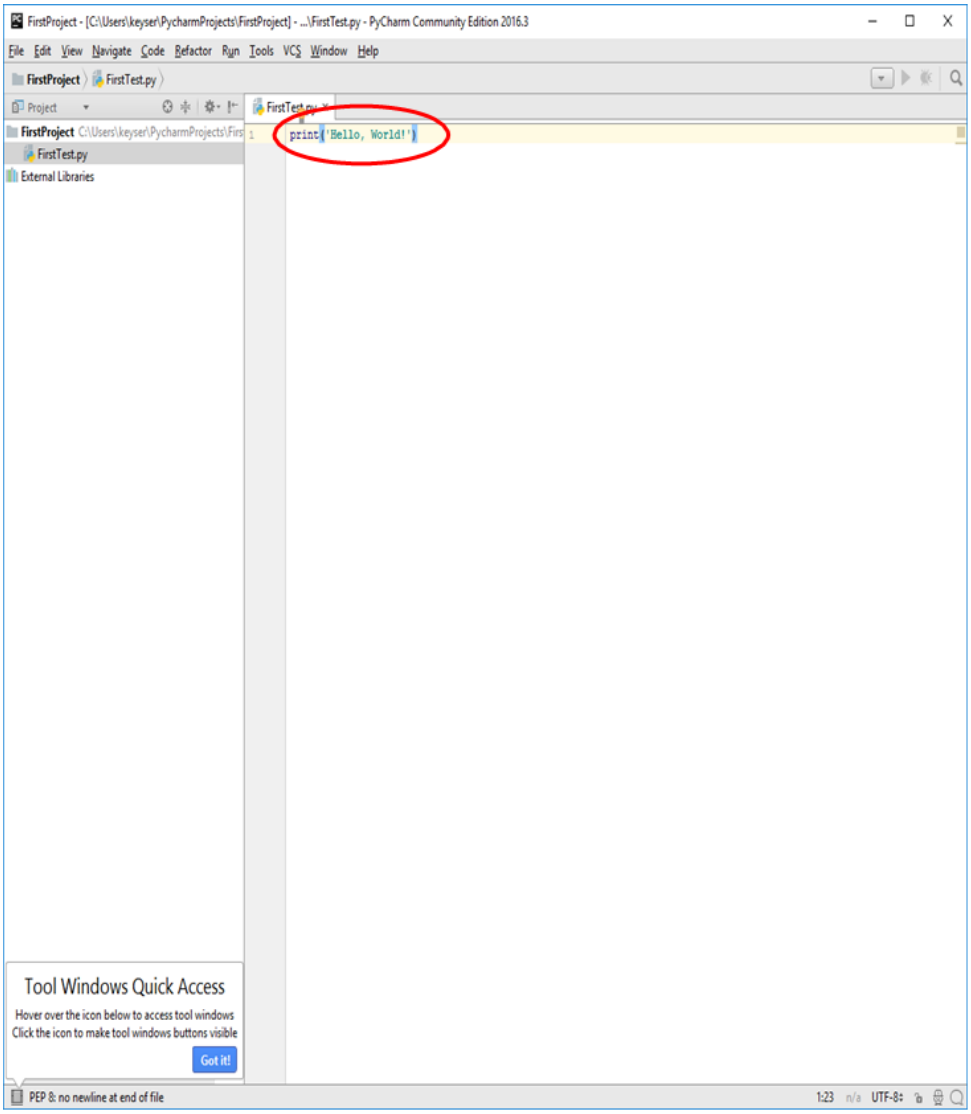
Type the name of the file you want (for example"FirstTest") and hit "OK"



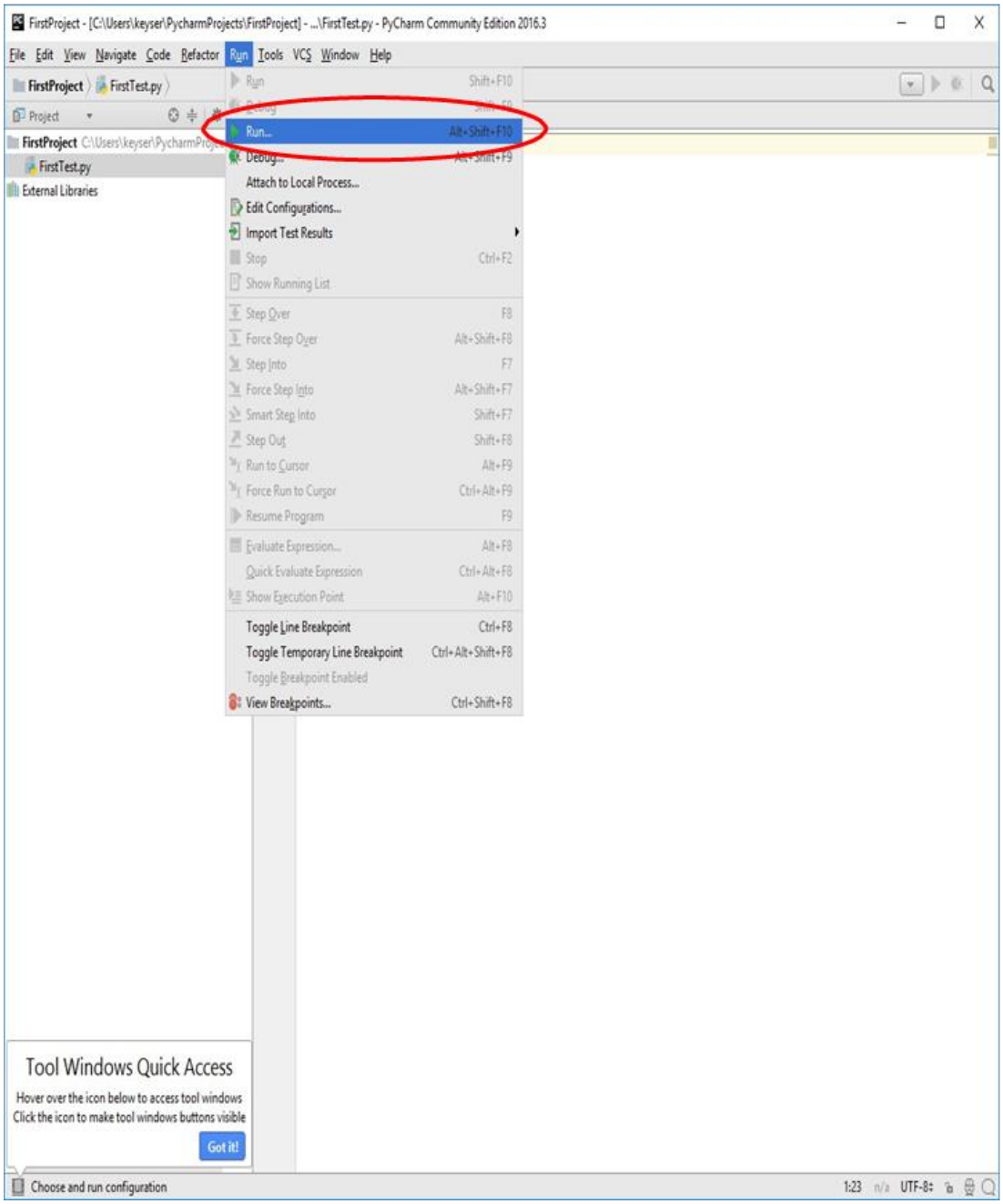
A new screen will pop up with the First Test.py file visible.



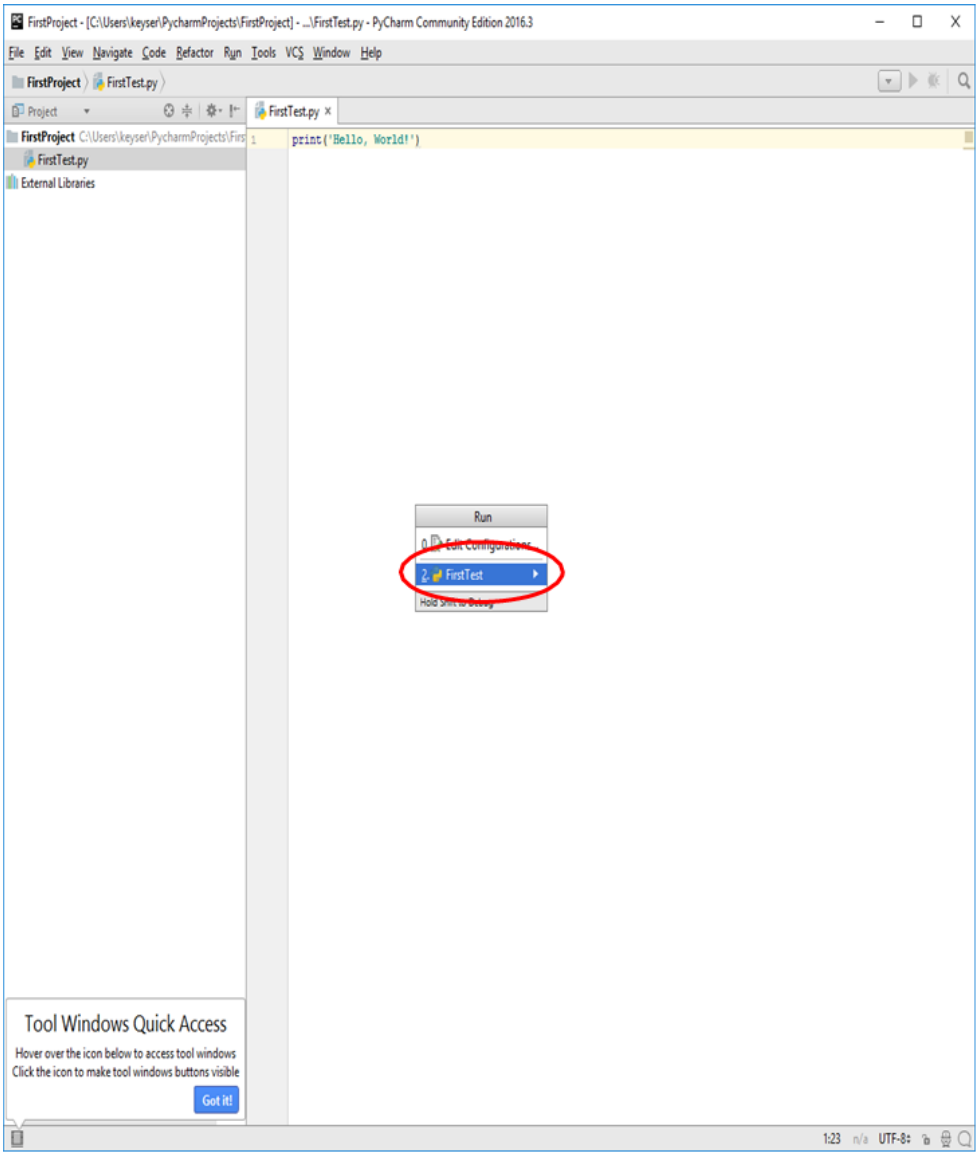
Go ahead and type in a simple program, like : print ('Hello,World!')



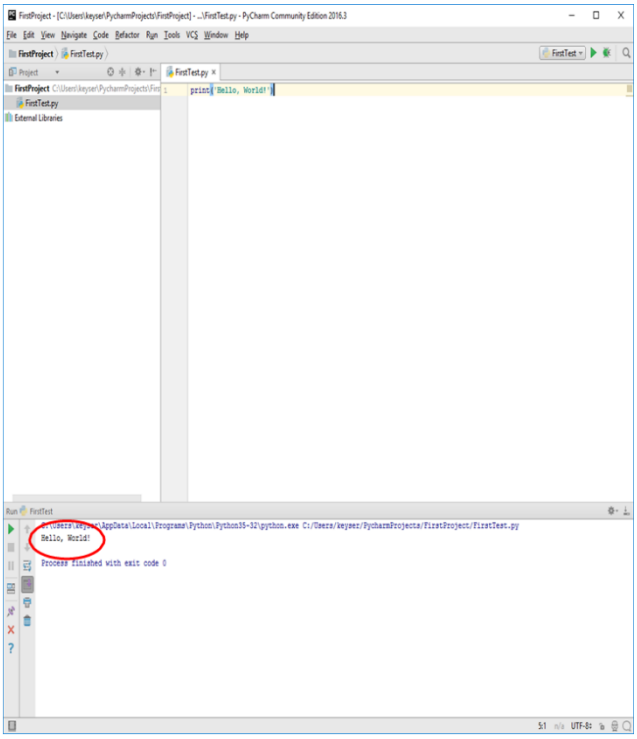
Goup to the "Run" menu and select "Run"(with the green arrow) to run your program:



Select the name of your file that you will run (in the example,that's"FirstTest"):



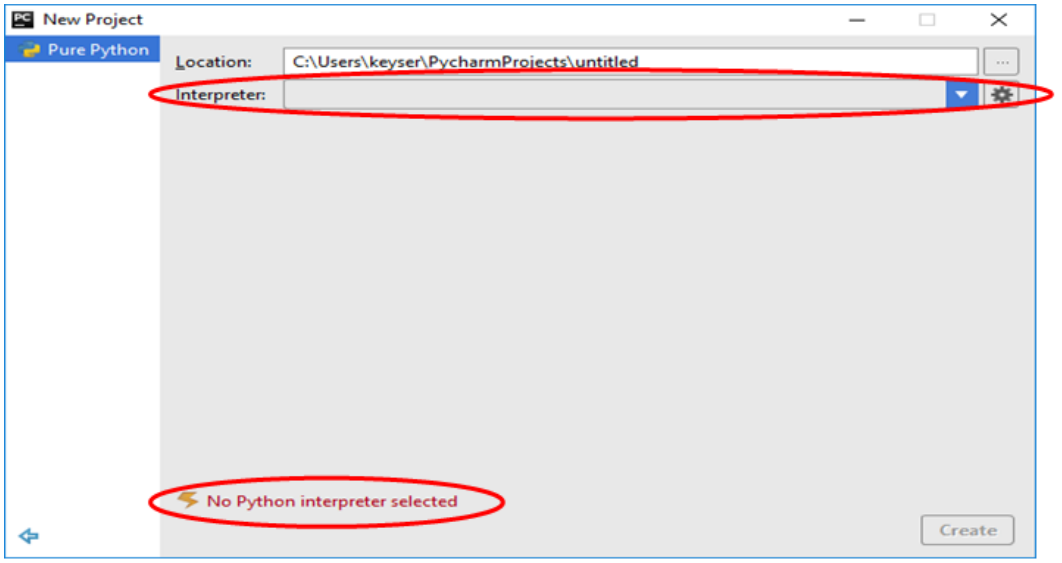
You should see the output of your program (in this case, the words Hello, World!, at the bottom of the screen:



That's all–you are ready to write programs using PyCharm.

Trouble Shooting:

If you do NOT have an interpreter listed, it means that PyCharm did not find a Python interpreter on your computer. You will see a window like this:

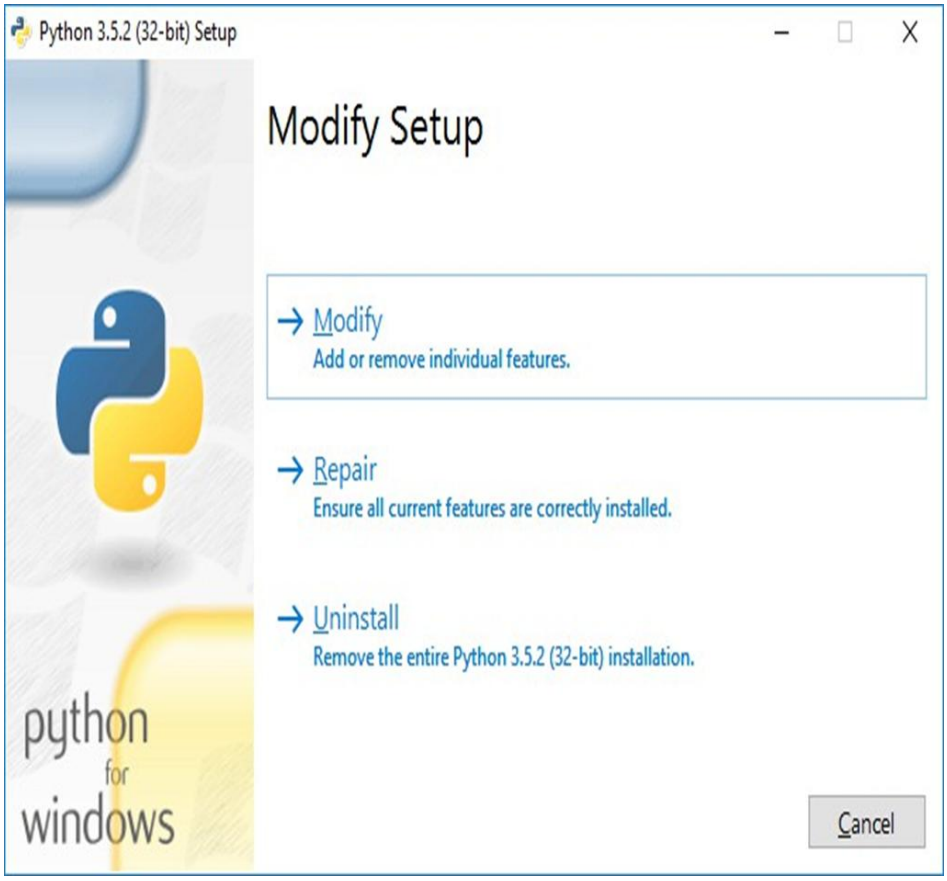


i. Did you install Python before installing PyCharm? If not, you need to go ahead and install Python! Close the PyCharm window and go back to do that first.

a) You should make sure the installation completed and that you can run IDLE

b) This is the most common reason that nothing is found. Once it is installed, go back and start PyCharm again, and it should find your Python interpreter.

ii. If you are even slightly unsure if you installed it, try installing it again. If you have it installed already, you should get a screen like this when you try to install:



iii. If you are sure you installed Python (and can test it by running IDLE), you will need to find the location where the python interpreter, usually named "python.exe"is installed. There are a few ways to do this:

a) Try looking in a standard installation location, (where***YOURUSERNAME**is your username – mine is keyser for instance) like: C:\Users***YOURUSERNAME***\AppData\Local\Programs\Python\Python35-32\python.exe

b) If you noted the location when you installed Python (or you specified your own location), you should be able to note that, and use that location

c) Do a search on your computer for python.exe:

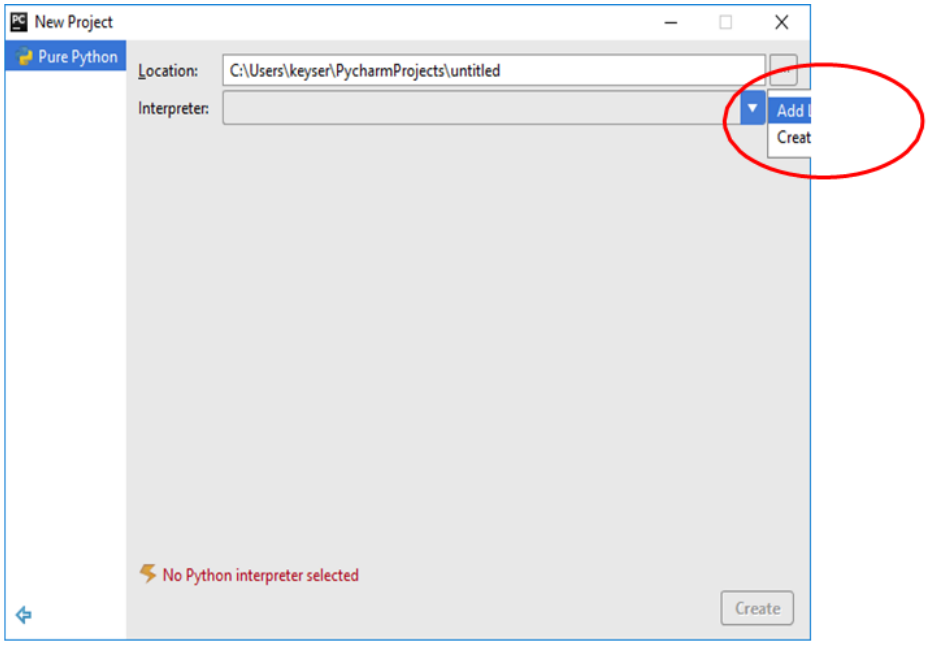
1. Open up the file explorer by typing "explorer" into the text box at lower left of your Windows system, and select File Explorer (not Internet Explorer)
2. At the far left, click on "This PC" or "Windows (C:)"
3. In the upper right, in the box labeled "Search This PC" or"Search Windows (C:)",type "python.exe"

4. There should be a file called “python” or “python.exe” found on your computer. That will be the one to use. Find the folder location for that file.

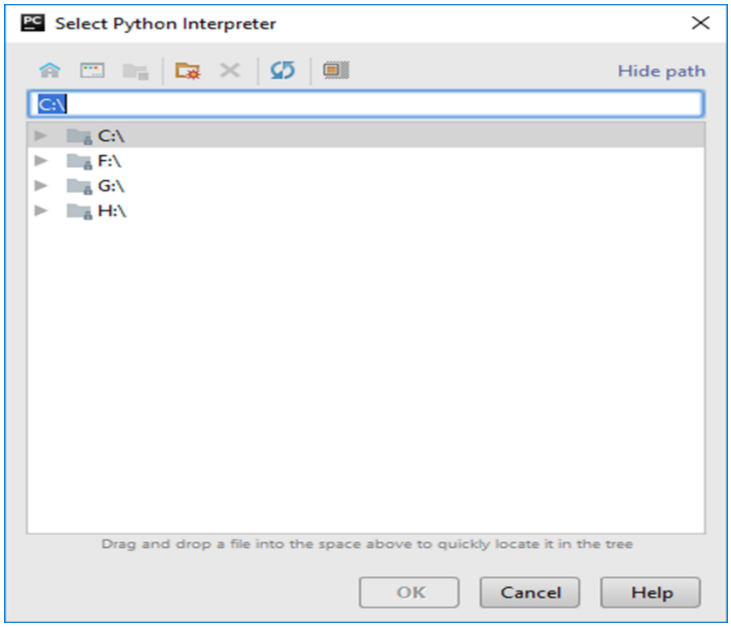
d) If you can’t find it by any of these means, try to install Python again:

1. When you first try to install, you should see the screen in(B), above.
2. Choose “uninstall” and let it uninstall your current Python installation
3. Then try to install, again. This time, choose “Customize Installation” when installing, and select a location you will be able to remember, such as C:\Program Files.
4. If you reinstall Python, try restarting PyCharm to see if it finds the interpreter automatically; if not, you will know the location to tell it to look.

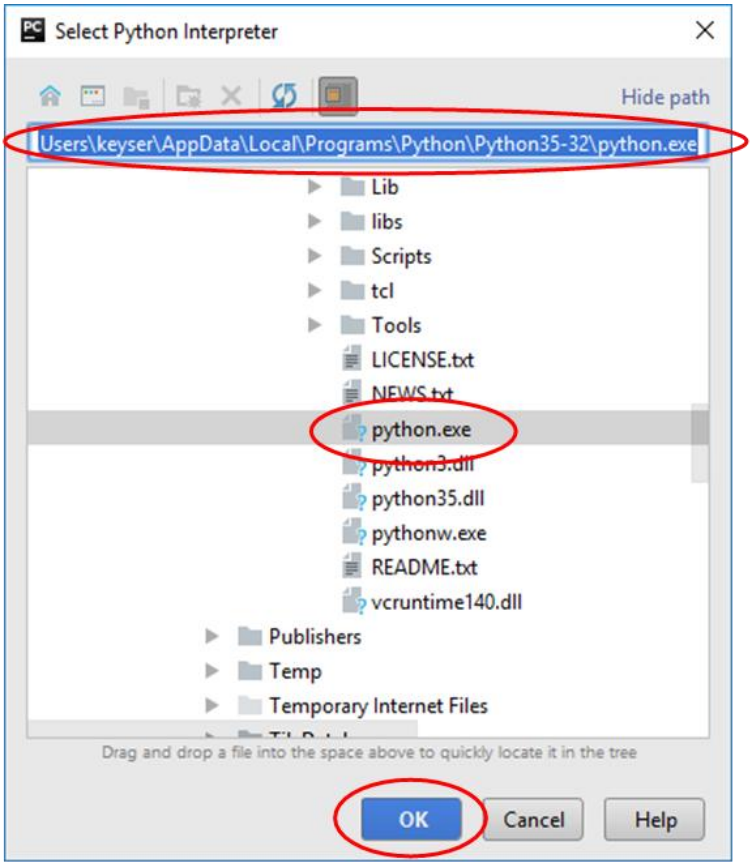
iv. Once you know the location of your python interpreter, you will click on the little settings icon (that looks like a gear wheel) next to your “Interpreter” box. There should be a link saying “Add Local” that pops up when you do that – click on it.



IV. This will bring up a file navigation window. You will need to go to the location of your python.exe file.



VI. When you have selected the location of your python.exe file, hit “OK”



VII. You are now OK to continue as above.

Introduction to Python

Arun Kumar
KSR College of Engineering, Kadapa, A.P.
Email: arunkumar.mtech09@gmail.com

Abstract

Programming languages are the sources of human interaction with machines. Advancement in the hardware and software technologies provided advancements in the programming languages like object-oriented concepts, capability to handle complex problems, and flexibility and robustness in multi-environment execution of code. In the recent years, python is one of the high-level programming languages with these characteristics. The advantages of python such as application specific libraries, executable code in different environments, and ability to process big data, makes it superior over other programming languages like Java, C++, .Net, etc. In the present literature, we introduce the basics of python programming language and emphasis the core applications of python programming.

Keywords: High-level language, Python Programming, Libraries, and applications of python programming

1. Introduction

Python is a high-level scripting language used for a wide range of tasks, including text processing, system administration, and internet-related activities. Unlike many other programming languages, Python has a small and easy-to-master core language, while allowing the addition of modules to perform countless tasks. It is a true object-oriented language and is available on various platforms. There is even a Python interpreter written entirely in Java, which enhances its suitability for internet-based problems.Python was developed in the early 1990s by Guido van Rossum, who was then at CWI in Amsterdam and is now at CNRI in Virginia. The language emerged from a project aimed at designing a programming language that would be easy for beginners to learn, yet powerful enough for advanced users. This foundation is evident in Python's clean and concise syntax, as well as the thorough implementation of concepts like object-oriented programming. Importantly, Python does not eliminate the option to program in a more traditional style.Overall, Python is an excellent choice for a first programming language, offering both accessibility for beginners and the power and capabilities that more experienced users eventually need.

There are several characteristics of Python that set it apart from other programming languages, and it's important to highlight these early to prevent confusion in later examples. More detailed information about these features will be provided later when the relevant topics are discussed thoroughly. Python statements do not require a specific ending character; the Python interpreter recognizes the completion of a statement by the presence of a newline, which is created when you press the “Return” key on your keyboard. If a statement extends beyond a single line, the

most secure approach is to place a backslash (\) at the end of the line to signal to Python that the statement will continue on the following line; additional backslashes can be used on further continuation lines as well [7]. (There are cases where backslashes are not necessary, which will be covered later.) Python offers some flexibility when writing a program, but there are essential rules that must always be followed. One such rule, which can be quite surprising to some, is that Python utilizes indentation (the amount of whitespace preceding the statement) to signify the presence of loops, rather than employing delimiters like curly braces ({}) or keywords (such as "begin" and "end") like many other programming languages do [8]. The degree of indentation you choose is not critical, but it must remain consistent at a given level of a loop, and any statements that are not indented should start in the first column [9].

2. Basics of Python

The important concepts of python programming are listed as : (i) Variables (ii) Data types (iii) Function definition (iv) Function calling (v) conditional statements (vi) Strings and Arrays (vii) Dictionaries and (viii)

2.1 Important Characters and Sets of Characters

- tab \t
- new line \n
- backslash \\
- string " " or ' '
- docstring """ """
- comparison operators == , < , > , <= , >= , !=
- Python type boolean True , False.
- Logical operators not , and , or

2.2 Order of Operations

Operator	***Description***
()	Parentheses (grouping)
f(args...)	Function call
x[index:index]	Slicing
x[index]	Subscription
x.attribute	Attribute reference
**	Exponentiation
+x, -x	Positive, negative
*, /, %	Multiplication, division, remainder
+, -	Addition, subtraction
in, not in, is, is not, <, <=, >, >=, <>, !=, ==	Comparisons, membership, identity
not x	Boolean NOT
And	Boolean AND
Or	Boolean OR

2.3 Variable Names

- case sensitive
- cannot start with a number (ex, 1_assd is not allowed)

Six Steps to Defining a Function

1. What should your function do? Type a couple of example calls.
2. Pick a meaningful name (often a verb or verb phrase): What is a short answer to "What does your function do"?
3. Decide how many parameters the function takes and any return values
4. Describe what your function does and any parameters and return values in the docstring
5. Write the body of the function
6. Test your function. Think about edge cases.

2.4 Integers and Strings

```
>>> int(45) 45
>>> int('45') 45
>>> str(45) '45'
>>> str('45') '45'
>>> int(str(45)) 45
```

2.5. CallingMethods

module_name.function_name(x)

```
math.sqrt(x)
random.randrange(2,5)
```

2.6 Conditionals and Branching

- if
- elif
- else

We have a boolean logic expression for if which works when the Boolean evaluates to True

2.7 String Operators

Description	*Operator*	*Example*	*Output*
equality	==	'cat' == 'cat'	True
inequality	!=	'cat' != 'Cat'	True
less than	<	'A' < 'a'	True
greater than	>	'a' > 'A'	True
less than or equal	<=	'a' <= 'a'	True
greater than or equal	>=	'a' >= 'A'	True
contains	in	'cad' in 'abracadabra'	True
length of str s	len(s)	len("abc")	3

2.8 String Indexing and Slicing

(s[a:b] means index a to length (b-a) or a to b index but not including b)

- s[2:3]
- s[0]

- s[:5]
- s[4:]

String is immutable (ex. s[4]='a' will not replace 'a' and index 4 of s)

2.9 String Methods

- A method is a function inside of an object.
- The general form of a method call is:
  - object.method(arguments)
  - dir(str)
  - help(str.method)

2.10 for Loops

num_vowels = 0 for char in s:
if char in 'aeiouAEIOU': num_vowels = num_vowels + 1
print num_vowels vowels = ''
for char in s:
if char in 'aeiouAEIOU': vowels = vowels + char
print vowels

2.11 Lists

Like for strings, slicing and indexing can also be used for lists

List = ['a','b',1]

- length of list len(list)
- smallest element in list min(list)
- largest element in list max(list)
- sum of elements of list (where list items must be numeric) sum(list)

>>>
>>>
'a'
>>>
'b'
>>>
'q'
>>>
a=[1,'ab',2,'pq'] a[1][0]

a[1][1]

a[3][1]

a[3][2]

2.12 List Methods

- append a value or string list.append('a')
- extended by another list list.extend(['a', 'b'])

```
>>> a = [5] + [6] + ['a',7]
>>> print (a) [5, 6, 'a', 7]
```

2.13. List Aliasing

Consider the following code:

```
>>> lst1 = [11, 12, 13, 14, 15, 16, 17]
>>> lst2 = lst1
>>> lst1[-1] = 18
>>> lst2
[11, 12, 13, 14, 15, 16, 18]
```

After the second statement executes, lst1 and lst2 both refer to the same list [11]. When two variables refer to the same objects, they are aliases. If that list is modified, both of lst1 and lst2 will see the change [12].

But be careful about:

```
>>> lst1 = [11, 12, 13, 14, 15, 16, 17]
>>> lst2 = lst1
>>> lst1 = [5, 6]
>>> lst2
[11, 12, 13, 14, 15, 16, 17]
```

And also:

```
>>> lst1 = [1,2,3]
>>> lst2 = lst1[:]
>>> lst2.remove(2)
>>> lst1 [1,2,3]
```

2.14 while Loops

```
i = 0
while i < len(s) and not (s[i] in 'aeiouAEIOU'): print(s[i])
i = i + 1

for char in s:
if not (char in 'aeiouAEIOU'): print(char)
```

The distinction between the two lies in the fact that the for loop examines each character in s, whereas the while loop stops immediately upon encountering a vowel.So the loops differ on any string where a consonant follows a vowel. while is an if statement in motion. It is a repeated loop until the boolean test evaluates to False [13].

def secret(s): i = 0 result = ''
while s[i].isdigit(): result = result + s[i] i = i + 1
print result

>>> secret('123') will give an error message when it runs the fourth time.

2.15 Global and Local Variables

Variables defined outside functions are global variables. Their values may be accessed inside functions without declaration. To modify to a global variable inside a function, the variable must be declared inside the function using the keyword global [13].

def x():
global num num = 5
def y():
num = 4

>>> num = 7
>>> print (num) 7
>>> x()
>>> print (num) 5
>>> y()
>>> print (num) 5

2.16 Dictionaries

The values within a dictionary can vary in type, but the keys need to be of an immutable data type like strings, numbers, or tuples.

- keys can be numbers, strings, Booleans
  - a list is unhashable in a dictionary (cannot be used as a key)
  - a tuple is hashable in a dictionary (can be used as a key).
- values can be dicts, strings, numbers, booleans, lists

for key in my_dict: value = my_dict[key]
This is same as:
for key, value in my_dict.items():

2.17 . Python for Agriculture Data Analysis

Visualization involves observing data from multiple perspectives. In Python, we can create visual representations of data using a variety of plots found in different libraries. In this article,

we will illustrate and forecast crop production data for various years by employing different visuals and Python libraries.

2.17 a). Importing crop dataset into python

The crop dataset in .csv, .xls and other formats can be imported in to python using different inbuilt python libraries like pandas, numpy, and matplotlib. The relevant code can be given as :

```
#Import crop datasets in to python
import pandas as pd
# load the dataset
crop = pd.read_csv('crop.csv')
# display top 5 values
crop.head()
```

2.17 b). Description of the dataset

```
# data description
crop.info()
```

2.17 Histogram of crop data

```
 # 2012 crop data in histogram analysis
crop['2012'].hist()

# 2013 crop data in histogram analysis
crop['2013'].hist()

  # display all year data
  crop.hist()

 # scatter plot 2013 data vs 2014 data
plt.scatter(crop['2013'],crop['2014'])
plt.show()

# line plot 2013 data vs 2014 data
plt.plot(crop['2013'],crop['2014'])
plt.show()
 # import required modules
import matplotlib.pyplot as plt
from scipy import stats
# assign data
x = crop['2017']
y = crop['2018']
# linear regression 2017 data vs 2018 data
```

```
slope, intercept, r, p, std_err = stats.linregress(x, y)
# function to return slope
def myfunc(x):
        return slope * x + intercept
mymodel = list(map(myfunc, x))
# scatter
plt.scatter(x, y)
# plotting the data
plt.plot(x, mymodel)
# display the figure
plt.show()

# import required modules
import matplotlib.pyplot as plt
from scipy import stats
# assign data
x = crop['2016']
y = crop['2017']
# linear regression 2017 data vs 2018 data
slope, intercept, r, p, std_err = stats.linregress(x, y)
# function to return slope
def myfunc(x):
        return slope * x + intercept
mymodel = list(map(myfunc, x))
# scatter
plt.scatter(x, y)
# plotting the data
plt.plot(x, mymodel)
# display the figure
plt.show()
```

3. Conclusion

In the present study, introduction to basic python programming is elaborated with relevant examples. The importance of python programming in various applications is enhanced with suitable code samples. Python being an important high-level programming language can be used in various fields of Science and Engineering. Furthermore, the sources of online python programming platforms are mentioned in the present study.

Sources of Online python programming

1. Python.org, https://www.python.org/
2. Learnpython, https://www.learnpython.org/
3. Python for beginners, https://www.pythonforbeginners.com/

4. Swaroopch, https://python.swaroopch.com
5. Awesome Python, https://github.com/vinta/awesome-python

References

X. Cai, H. Langtangen and H. Moe, "On the Performance of the Python Programming Language for Serial and Parallel Scientific Computations," Scientific Programming, vol. 13, no. 1,pp. 31-56, 2005.

G. V. Rossum, "Computer Programming for Everybody-A Scouting Expedition for the Programmers of Tomorrow," CNRI Proposal 90120-1a, Corporation for National Research Initiatives, 1999.

P. J. Guo. (2014) "Python is now the most popular introductory teaching language at top U.S. universities". [Online]. Available: http://cacm.acm.org/blogs/blog-cacm/176450-python-is-nowthe- most-popular-introductory-teaching-language-at-top-u-suniversities/fulltext, 2014

R. Sand. (2012) The slideshare website. [Online]. Available: https://www.slideshare.net/blackducksoftware/open-source-bythe-numbers/

G. Piatetsky. (2017) Kdnuggets website. [online]. Available: https://www.kdnuggets.com/2017/08/python-overtakes-rleader- analytics-data-science.html

S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. Seljebotn and K. Smith, "Cython: The Best of Both Worlds," Computing in Science & Engineering, vol. 13, no. 2, pp. 31-39, 2011.

S. V. D. Walt, S. Colbert and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," Computing in Science & Engineering, vol. 13, no. 2, pp. 22-30,2011.

W. McKinney, "Pandas: a foundational Python library for data analysis and statistics," Python for High Performance and Scientific Computing, pp. 1-9, 2011.

KM Lukas. (2017) The machinelearningexp website. [Online]. Available:http://machinelearningexp.com/data-scienceperformance-of-python-vs-pandas-vs-numpy/

Abadi, Martín, P. Barham, J. Chen, Z. Chen, A. Davis et al. "Tensorflow: A system for large-scale machine learning," in OSDI, 2016, p. 265-283.

Bird, Steven, and E. Loper, "NLTK: the natural language toolkit," in Proc. of the ACL 2004 on Interactive poster and demonstration sessions, 2004, p. 31.

F. Pedregosa, G. Varoquaux, A. Gramfort et al. “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning, vol. 12, pp. 2825-2830, 2011.

J. D. Hunter, "Matplotlib: A 2D Graphics Environment," in Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, May-June 2007.

A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, “Novel dataset for Fine-Grained Image Categorization: Stanford dogs,” in Proc. Computer Vision and Pattern Recognition workshop on Fine-Grained Visual Categorization (FGVC), vol. 2, p. 1, 2011.

Descriptive and Inferential Statistics in Abiotic Stress Management

Naveena K
Centre for Water Resources Development and Management, Kozhikode
naveenak@cwrdm.org

Introduction

Abiotic stress refers to the negative impact of non-living factors such as drought, salinity, extreme temperatures, and heavy metals on plant growth and crop productivity. Tackling these challenges is vital for maintaining global food security, especially given the impacts of climate change.In agricultural research, statistics play a vital role in understanding, predicting, and mitigating the effects of abiotic stress on plants.

Descriptive statistics help summarize and present data in a meaningful way. They are used to analyze measurements like plant height, chlorophyll content, biomass, yield, and stress tolerance indices. Metrics such as mean, median, standard deviation, and range allow researchers to evaluate the performance of different crop varieties or treatments under stress conditions.

On the other hand, **inferential statistics** allow researchers to draw conclusions and make predictions about larger populations based on sample data. Techniques such as ANOVA, regression analysis, t-tests, and chi-square tests help determine the significance of differences among treatments, environmental effects, or genetic traits.

Using inferential methods, scientists can test hypotheses related to stress resistance and assess the effectiveness of mitigation strategies such as bio-fertilizers, irrigation management, or genetic modifications. These statistical tools are also key in plant breeding programs aimed at developing stress-resilient varieties.

Methodology

Descriptive Statistics in Hydrology:

Descriptive statistics enable plant scientists and agronomists to summarize large datasets related to abiotic stress factors such as drought, salinity, extreme temperatures, and nutrient deficiencies into key numerical measures. These statistics simplify complex datasets and make

it easier to interpret and communicate important patterns in how crops respond to stress. Descriptive statistics help identify outliers, missing data, or inconsistencies in experimental measurements, allowing researchers to assess data quality and make necessary corrections. They are also crucial for identifying long-term changes in crop performance or environmental conditions, such as declining soil moisture trends or increasing salinity levels, which are vital for planning effective stress management strategies.

One of the most widely used measures of central tendency in abiotic stress studies is the **mean**, which represents the average response of a crop trait (e.g., leaf water content, chlorophyll concentration, biomass, or yield) under stress conditions. The mean allows researchers to quantify the average effect of a stressor across treatments or genotypes. The **median**, another important measure, indicates the middle value in a dataset and is particularly useful while working with skewed data or outliers—common in field trials where individual plant responses can vary widely. The **mode**, while less frequently used, can still provide insights into the most common response levels in certain scenarios, though biological datasets often lack a distinct mode due to continuous data distributions.

Measures of dispersion, such as **range** and **standard deviation**, are equally important because they quantify variability in plant responses. The **range** shows the span between the minimum and maximum values observed in stress experiments—such as the difference in biomass between the most and least affected plants. **Standard deviation**, a more robust measure, reflects how much individual observations deviate from the mean, and is often used to quantify the consistency or variability of plant traits under stress. A **low standard deviation** might indicate uniform stress tolerance, while a **high standard deviation** could point to genetic variability or inconsistent experimental conditions.

The **coefficient of variation (CV)**, which is the ratio of the standard deviation to the mean, is commonly used to compare the relative variability between different traits or treatment groups. For example, if leaf temperature varies more under heat stress than relative water content under drought, the CV can highlight this difference in variability. Additional measures such as the **interquartile range (IQR)**—the range between the 25th and 75th percentiles—help in identifying the central spread of data, especially when data are skewed. **Percentile ranges** also

offer valuable insights into the range of responses, especially in screening trials for extreme tolerance or sensitivity.

Skewness and **kurtosis** are advanced descriptive measures used to understand the shape of the data distribution. **Skewness** indicates whether the data are asymmetrically distributed, which is important in identifying whether extreme stress responses are more likely to be negative (e.g., yield loss) or positive (e.g., stress-induced growth in some genotypes). **Positive skewness** suggests a tendency toward extreme high responses (e.g., certain genotypes thriving under stress), while **negative skewness** indicates more frequent low responses (e.g., stress damage). **Kurtosis** measures whether the data distribution is more peaked or flatter than a normal distribution. High kurtosis might indicate a higher frequency of extreme responses, such as very high electrolyte leakage in salt stress experiments, signaling potential outliers or stress-sensitive accessions.

R code for computing basic descriptive statistics

```
# Load the s dataset
data <- c(20.2, 19.4, 23.1, 22.5, 21.7, 19.9, 20.3, 21.1, 20.8, 22.2, 19.8, 18.9, 20.7, 20.5, 22.3, 21.5, 21.9, 20.6, 23.5, 21.8)
# Compute basic descriptive statistics
mean_value <- mean(data) # Mean
median_value <- median(data) # Median
mode_value <- median(data[duplicated(data)]) # Mode (using base R)
range_value <- range(data) # Range
sd_value <- sd(data) # Standard deviation
cv_value <- sd_value / mean_value * 100 # Coefficient of variation
or
# Load the hydrostats package
library(hydrostats)
# Compute descriptive statistics using the hydrostats package
basic_stats <- hydrostats::basics(data) # Basic statistics
quantile_stats <- hydrostats::quantile(data) # Quantiles
freq_stats <- hydrostats::frequency(data) # Frequency analysis
```

Illustration:

The reference evapotranspiration (ETo) in Thiruvananthapuram District, Kerala (8.29N Latitude, 76.57E Longitude), was analyzed for a 51-year period from 1970 to 2020. The average annual ETo value in Thiruvananthapuram is 1384.71 mm. With a relatively low standard deviation of 13.87 mm, there is a moderate level of variability around the average ETo values. The positive skewness of 0.50 suggests a slight inclination towards higher ETo values, while the kurtosis of 0.16 indicates a distribution close to normal.

The range of ETo values extends from 1355.27 mm to 1422.74 mm, indicating significant variability throughout the year. Analyzing monthly data, the months of June, July, and August exhibit the lowest average ETo values, whereas January, November, and December have the highest averages. May displays a distribution with a heavy tail and potential fluctuations.

Considering the seasonal analysis, the Winter season has the highest average ETo, followed by the southwestern monsoon, northeast monsoon, and summer seasons. Across all seasons, the variability in ETo is generally low, except for the winter season which exhibits slightly higher variability. These findings provide valuable insights into the evapotranspiration patterns in Thiruvananthapuram District over the studied period, aiding in understanding the water demands and climatic conditions of the region.

Table: ET statistics of Thiruvananthapuram.

THIRUVANANTHAPURAM							
	Average (mm)	**Standard deviation (mm)**	**Coefficient of variance (%)**	**Skewness**	**Kurtosis**	**Maximum value**	**Minimum value**
Month-wise Analysis							
Jan	165.17	3.03	1.84	0.65	0.30	174.55	160.31
Feb	137.54	3.40	2.47	0.79	0.28	147.72	132.69
Mar	127.08	1.81	1.42	-0.27	0.06	131.00	122.47
Apr	95.23	1.35	1.42	0.52	-0.30	98.67	93.11
May	73.95	1.01	1.37	-1.33	4.31	76.10	69.84
Jun	59.49	0.80	1.34	0.19	0.27	61.67	57.95

Jul	65.76	0.79	1.20	0.59	-0.31	67.44	64.57
Aug	85.33	1.08	1.26	0.56	0.26	88.07	83.10
Sep	109.76	1.13	1.03	0.33	-0.26	112.25	107.50
Oct	141.02	1.59	1.12	-0.12	-0.70	143.87	137.29
Nov	155.12	1.54	0.99	-0.09	-0.06	158.36	151.47
Dec	169.26	2.69	1.59	0.76	1.63	178.45	163.98
Annual ETo	1384.71	13.87	1.00	0.50	0.16	1422.74	1355.27
Season-wise Analysis							
Summer	296.27	3.35	1.13	0.24	0.61	305.77	287.84
SW Monsoon	320.34	3.25	1.01	0.60	-0.15	327.70	314.87
NE Monsoon	296.14	2.80	0.95	0.19	-0.75	301.51	290.06
Winter	471.96	7.45	1.58	0.48	-0.15	490.81	458.53

Inferential Statistics:

Inferential statistics in hydrology utilizes statistical methods to draw conclusions or forecasts regarding a broader population based on a limited sample of data.This is important in hydrology because it is often not feasible or practical to collect data from an entire population, such as all the streams in a large watershed, so we must use statistical methods to make predictions about the population condiering the a sample of data.

Inferential statistics in hydrology can be used to test hypotheses, make predictions, and estimate parameters of interest, such as the mean or variance of a population. Typical inferential statistical methods employed in hydrology consist of hypothesis testing, constructing confidence intervals, conducting regression analysis, and performing time series analysis.Hypothesis testing involves testing a hypothesis or claim about a population parameter, such as the mean streamflow or rainfall intensity, based on a sample of data. This

can help us determine whether there is sufficient evidence to support or reject the hypothesis or claim. Confidence intervals provide a range of values within which a population parameter is likely to lie, based on a sample of data. This allows us to approximate the population parameter with a specified confidence level and evaluate the accuracy of our estimate.

T, Z, and F tests are all types of statistical tests are used in hydrology to make inferences about populations based on sample data.

T-tests are frequently utilized in hydrology to analyze the average values of two different groups or sets of data. In particular, a t-test can determine whether the disparity between the averages of two populations or samples is statistically meaningful. For instance, a hydrologist might seek to evaluate the average streamflow of two watersheds to determine if a significant difference exists between them. In order to accomplish this, the hydrologist would gather streamflow information from both watersheds, compute the average streamflow for each area, and subsequently employ a t-test to assess whether the difference in means is statistically significant. There are primarily two varieties of t-tests applicable in hydrology: the independent samples t-test and the paired samples t-test.

The independent samples t-test is used when the two samples being compared are independent of each other, meaning that there is no overlap between the two groups. For example, the hydrologist might collect streamflow data from two different watersheds that are not connected in any way. The independent samples t-test is based on the assumption that the variances of the two groups are equal, and it can be utilized to determine if the difference between the means of the two groups is statistically significant.

The paired samples t-test, applicable when the two samples are related or paired in some way. For example, the hydrologist might collect streamflow data from a single watershed at two different locations or at two different times. The paired samples t-test assumes that the differences between the pairs are normally distributed, and can be used to test whether the mean difference between the two samples is statistically significant. Before conducting a t-test, it's important to ensure that the assumptions of the test are met, such as normality of the data and equality of variances (for independent samples t-tests). Additionally, it's always a good idea to consult with a statistician or hydrologist who has expertise in the use of t-tests before conducting any statistical analysis.

Example of how to conduct an independent t-test and a paired t-test in R using hydrological data.

```
## Independent t-test example
```

Let us consider, **flow1** and **flow2** data representing river discharge measurements. Each dataset consists of 100 observations. We perform an independent t-test to compare the means of **flow1** and **flow2** and determine if there is a significant difference between the two datasets. The independent t-test is used because the samples (**flow1** and **flow2**) are independent, meaning the measurements in one dataset are not paired or related to the measurements in the other dataset.

```
.# Conduct the t-test
t.test(flow1, flow2, alternative = "two.sided", mu = 0, var.equal = TRUE)
```

Output:

data: flow1 and flow2

t = -6.0315, df = 198, p-value = 7.843e-09

interpretation:

From these findings, we can deduce that a statistically significant difference exists between the average river discharge on rainy days for flow1 and flow2.The small p-value suggests observed difference in means is unlikely to have occurred by chance alone.

```
## Paired t-test example
```

Let us consider simulated discharge data before June and after June for comparison.

```
t.test(pre_june, post_june, alternative = "two.sided", paired = TRUE)
```

Output:

Paired t-test

data: pre_june and post_june

t = -5.9658, df = 49, p-value = 2.643e-07

Interpretation:

- The calculated t-value is -5.9658, with degrees of freedom (df) equal to 49.

- The p-value is very small (2.643e-07), indicating strong evidence against the null hypothesis.

Based on these results, we can conclude that there is a statistically significant difference between the mean river discharge before and after the month of June. The negative t-value suggests that the mean river discharge after June tends to be lower than the mean river discharge before June.

Z-test

Z-tests represent a different category of statistical tests that can be utilized in hydrology to draw conclusions about populations from sample data. Specifically, the z-test is used to test hypotheses about population means or proportions when the sample size is sufficiently large and the population standard deviation is known.

For example, let's look at a situation in which we aim to assess the percentage of water sources that comply with a certain safety standard.Then hypothesis formulated in the form of

- Null hypothesis (H_0): The proportion of water sources meeting the safety standard is equal to a specified value.
- Alternative hypothesis (H_1): The proportion of water sources meeting the safety standard is different from the specified value.

Test statistic calculation (z-score): Calculate the z-test statistic using the formula:

$$z = (\hat{p} - p_0) / \text{sqrt}((p_0(1 - p_0)) / n)$$

Where p_0 is the specified proportion under the null hypothesis, $\hat{p}$ is the sample proportion, and n is the sample size.

If the calculated z-score falls within the rejection region (i.e., exceeding the critical value(s)) or the p-value is lower than the chosen significance level, reject the null hypothesis. This suggests a significant difference between the observed sample proportion and the specified proportion, indicating a difference in the proportion of water sources meeting the safety standard.

Example of how to conduct a one sample z-test in R using hydrological data:

Let's consider a scenario where we have 1000 water sources to check the safety standard, and we assuming that 75% of the water sources meet the safety standard. We will then perform a z-test for proportions to analyse the results.

R

```
# Calculate the observed sample proportion
p_hat <- sum(water_data) / n
# Specify the null hypothesis proportion
p_null <- 0.75
# Calculate the test statistic (z-score)
z_score <- (p_hat - p_null) / sqrt((p_null * (1 - p_null)) / n)
# Set the desired significance level
alpha <- 0.05
# Calculate the critical value (two-tailed test)
critical_value <- qnorm(1 - alpha / 2)
# Calculate the p-value (two-tailed test)
p_value <- 2 * (1 - pnorm(abs(z_score)))
# Hypothesis testing
if (abs(z_score) > critical_value) {
  # Reject the null hypothesis
  message("Reject the null hypothesis. There is a significant difference in the proportion of water sources meeting the safety standard.")
} else {
  # Fail to reject the null hypothesis
  message("Fail to reject the null hypothesis. There is no significant difference in the proportion of water sources meeting the safety standard.")
}
# Print the calculated test statistic, critical value, and p-value
cat("Test Statistic (z-score):", z_score, "\n")
cat("Critical Value:", critical_value, "\n")
cat("P-value:", p_value, "\n")
```

Output:

Test Statistic (z-score): -3.794733

Critical Value: 1.959964

P-value: 0.0001478023

Interpretation: Based on the results obtained, we are rejecting the null hypothesis. The data indicates that there is a notable disparity between the percentage of water sources that comply with the safety standard and the given value (null hypothesis proportion).The data indicates that the observed proportion is significantly lower than the specified value.

F-test:

F-test: The F-test is used to test if the variances of two or more populations are equal. In hydrology, F-tests can be used to compare the variances of streamflow data in two or more watersheds, for example.

The specific formulas and assumptions used in each of these tests may vary depending on the particular application and statistical software used. It's important to ensure that the appropriate assumptions and conditions are met before applying any statistical test to hydrological data. Additionally, it's always a good idea to consult with a statistician or hydrologist who has expertise in the use of these tests before conducting any statistical analysis.

Let us consider, **flow1** and **flow2** data representing river discharge measurements. Each dataset consists of 100 observations. We perform an independent f-test to compare the variences of **flow1** and **flow2** and determine if there is a significant difference between the two datasets.

```
# Calculate the variance of the two samples
var1 <- var(flow1)
var2 <- var(flow2)
# Conduct the F-test
f_stat <- var1 / var2
p_value <- pf(f_stat, df1 = length(flow1) - 1, df2 = length(flow2) - 1, lower.tail = FALSE)
# Print the results
```

```
cat("F-statistic:", f_stat, "\n")
cat("P-value:", p_value, "\n")
```

Output:

F-statistic: 0.891098

P-value: 0.7163424

Interpretation:

Given that the p-value (0.7163424) exceeds the selected significance level (commonly set at 0.05), we do not reject the null hypothesis. This indicates that there is no statistically significant difference between the variances of the two datasets (flow1 and flow2). Consequently, we lack adequate evidence to assert that the variance of flow1 differs significantly from that of flow2.Note that the F-test is used to compare the variances of two populations. In hydrology, this test can be used, for example, to compare the variability of flow data in two different watersheds. It's important to ensure that the assumptions of the F-test are met before conducting the test on hydrological data.

ANOVA:

ANOVA, or analysis of variance, is another statistical test under F that can be used in hydrology to compare the means of two or more populations or samples. Specifically, ANOVA can be used to test whether there are significant differences in the means of two or more groups or treatments. In hydrology, ANOVA can be used, for example, to compare the mean streamflow or mean rainfall depth between different watersheds or regions, or to test the effectiveness of different management practices or treatments on water quality or quantity. To perform an ANOVA, the hydrologist first calculates the F-statistic, which is a measure of the variation between the groups or treatments relative to the variation within the groups or treatments. The hydrologist then compares the F-statistic to a critical value based on the desired level of significance and the degrees of freedom for the groups or treatments. The formula for ANOVA is a bit more complex than for the t-test or F-test, but can be calculated using statistical software such as R or Python. ANOVA involves partitioning the total variation in the data into two components: the variation between the groups or treatments, and the variation within the groups or treatments. The F-statistic is then calculated as the ratio of the variation between the

groups to the variation within the groups. ANOVA assumes that the populations or samples being compared are normally distributed, and that the samples are independent of each other. Additionally, ANOVA assumes that the variances of the populations or samples are equal, although there are variations of ANOVA that can handle unequal variances. It's important to ensure that the assumptions of ANOVA are met before applying the test to hydrological data, and to consult with a statistician or hydrologist who has expertise in the use of ANOVA before conducting any statistical analysis.

Example of how to conduct an ANOVA in R using hydrological data:

```
# Conduct the ANOVA
model <- aov(flow ~ watershed, data = df)
summary(model)
```

Output:

Analysis of Variance Table

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Group	2	0.867	0.43337	2.9762	0.05251
Residuals	297	43.246	0.14561		

Interpretation:

The p-value exceeds the common significance threshold of 0.05, indicating that there is insufficient evidence to dismiss the null hypothesis. Nevertheless, it is near the significance level, which suggests a slight distinction between the group means. Additional exploration may be justified.

References:

Wilcox, R. R. (2016). Understanding and applying basic statistical methods using R. John Wiley & Sons.

Schmuller, J. (2017). Statistical analysis with R for dummies. John Wiley & Sons.

Naghettini, M. (2017). Fundamentals of statistical hydrology.

Maity, R., & Maity. (2018). Statistical methods in hydrology and hydroclimatology (Vol. 555). Singapore: Springer.

Correlation and Regression Analysis

Mohan Kumar TL, Santosha Rathod and Nirmal B
University of Agricultural Sciences, Bangalore.
ICAR- National Institute of Abiotic Stress Management, Baramati
ICAR-National Academy of Agricultural Research Management, Hyderabad
Email: monis.iasri@gmail.com

The correlation is a widely used and highly valuable statistic. It is represented by a single number that conveys the strength of the relationship between two variables.

Pearson correlation coefficient

The Pearson correlation coefficient (r) is the most common way of measuring a linear correlation. It is a number between –1 and 1 that measures the strength and direction of the relationship between two variables

The formula for the correlation is:

$$r = \frac{N\sum xy - \sum x \sum y}{\sqrt{(N\sum x2 - (\sum x)2)(N\sum y2 - (\sum y)2)}}$$

where:

- N is the number of pairs of scores,
- $\sum xy$ is the sum of the products of paired scores,
- $\sum x$ is the sum of x scores,
- $\sum y$ is the sum of y scores,
- $\sum x^2$ is the sum of squared x scores,
- $\sum y^2$ is the sum of squared y scores.

The Pearson correlation coefficient is applicable when (1) the association is linear, (2) both variables are numerical, (3) they follow a normal distribution, and (4) there are no outliers present.

cor() function is utilized to calculate the Pearson correlation coefficient in R. significance of the correlation os tested using the **cor.test()** function.

The symbol r represents the correlation. R will consistently range from -1.0 to +1.0. A negative correlation indicates a negative relationship, while a positive correlation signifies a positive relationship.

Illustration: The correlation between the height of a person and self-confidence of 20 persons.

Person	Height (x)	Self Confidence (y)	x*y	x*x	y*y
1	68	4.1	278.8	4624	16.81

2	71	4.6	326.6	5041	21.16
3	62	3.8	235.6	3844	14.44
4	75	4.4	330	5625	19.36
5	58	3.2	185.6	3364	10.24
6	60	3.1	186	3600	9.61
7	67	3.8	254.6	4489	14.44
8	68	4.1	278.8	4624	16.81
9	71	4.3	305.3	5041	18.49
10	69	3.7	255.3	4761	13.69
11	68	3.5	238	4624	12.25
12	67	3.2	214.4	4489	10.24
13	63	3.7	233.1	3969	13.69
14	62	3.3	204.6	3844	10.89
15	60	3.4	204	3600	11.56
16	63	4	252	3969	16
17	65	4.1	266.5	4225	16.81
18	67	3.8	254.6	4489	14.44
19	63	3.4	214.2	3969	11.56
20	61	3.6	219.6	3721	12.96
Sum	1308	75.1	4937.6	85912	285.45

N=20

$\sum xy$=4937.6

$\sum x$=1308

$\sum y$=75.1

$\sum x2$=85912

$\sum y2$=285.45

R=0.73

So, the correlation for our twenty cases is 0.73, which is a fairly strong positive relationship.

As in all hypothesis testing, we need to first determine the significance level. We may use the common significance level of alpha = .05. This means that we are conducting a test where the odds that the correlation is a chance occurrence is no more than 5 out of 100. We have to compute the degrees of freedom or df. The df is simply equal to N-2 or, in this example, is 20-2 = 18. Finally, we have to decide whether we are doing a one-tailed or two-tailed test. In this

example, since we have no strong prior theory to suggest whether the relationship between height and self-confidence would be positive or negative, we'll opt for the two-tailed test. With the available information – the level of significance (alpha = .05)), degrees of freedom (df = 18), and type of test (two-tailed) – we can now test the significance of the correlation we found. If we look up this value in the table at the back of any statistics book, we find that the critical value is .4438. This means that if the correlation is greater than .4438 or less than -.4438, conclude that the odds are less than 5 out of 100 that this is a chance occurrence. Since our correlation of .73 is actually quite a bit higher, we conclude that it is not a chance finding and that the correlation is "statistically significant". We can reject the null hypothesis and accept the alternative.

Spearman's rank correlation coefficient

Spearman's rank correlation coefficient, rs, indicates the relationship between two sets of ordinal data. It measures how variations in one ordinal data set correspond to changes in another ordinal data set.

The formula for the Spearman rank correlation coefficient when there are no tied ranks is:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

R Language provides two methods to calculate the correlation coefficient. By using the functions **cor()** or **cor.test()** it can be calculated. It is important to highlight that cor() calculates the correlation coefficient, while cor.test() performs a test to determine the association or correlation between paired samples. This function provides both the correlation coefficient and the significance level (or p-value) related to the correlation.

- The function cor( ) calculates the correlation coefficient.

- The cor.test() function assesses the association or correlation between two related samples, providing both the correlation coefficient and the significance level (or p-value) of the correlation.

- The syntax is cor(x, y, method = c("pearson", "spearman")).

cor.test(x, y, method=c("pearson", "spearman"))

- **x, y**: numeric vectors with same length
- **method**: correlation method

Regression Analysis

Regression Analysis is a widely used statistical method for modeling the relationships between two or more variables. It aims to predict a continuous dependent variable based on multiple independent variables. This technique helps identify significant factors, discard

irrelevant ones, and understand their interactions. Often used to model or analyze data, regression analysis aids in comprehending variable relationships to accurately predict outcomes..

In its most basic sense, regression analysis closely resembles correlation; in reality, the fundamental mathematical models are nearly the same. Nonetheless, regression analysis can be applied when there are multiple explanatory variables and when different types of data are combined. The overarching regression model is:

$Y= \alpha +bX1+cX2+….+$ error

In this equation, α represents a constant, while X1, X2, and others are the predictor variables. The error term signifies the gap between the actual value and the predicted value of γ.

The vocabulary related to regression analysis includes:

• Dependent variable or target variable: The variable that is being predicted.

• Independent variable or predictor variable: Variables used to estimate the dependent variable.

• Outlier: An observation that significantly diverges from other data points. It should be excluded as it may affect the results negatively.

• Multicollinearity: A condition where two or more independent variables exhibit a strong linear relationship.

• Homoscedasticity or homogeneity of variance: A condition where the error term remains consistent across all values of the independent variables.

Regression analysis serves two main purposes. Firstly, it is commonly utilized for prediction and forecasting, which aligns with the realm of machine learning. Secondly, it is used to determine causal relationships between independent and dependent variables. s.

- The most basic scenario of linear regression involves determining a relationship using a linear model (i.e., a line) between a single independent variable (input feature) and a dependent output variable. This situation is referred to as Bivariate Linear Regression.
- Conversely, when a linear model illustrates the relationship between a dependent output and several independent input variables, it is known as Multivariate Linear Regression.
- The dependent variable is continuous, while the independent variables may be either continuous or not. We establish the relationship between them through the best fitting line, commonly referred to as the Regression line. The formula for a line is,

$y = m * x + b$

Where,

- **x:** Independent Variable
- **y:** Dependent Variable

- **m:** Slope of Line
- **b:** y Intercept

To assess the best fit line, the most widely used approach is the Least Squares Method. This technique determines the regression line by reducing the sum of the squared differences between the regression line and the data points. An additional method for finding this line is referred to as R Squared analysis.

When the relationship between the input variables and the output is not very complex this can be used. Also, it is very sensitive to outliers.

Multiple regression analysis is employed to determine if there exists a statistically significant connection between groups of variables. Its purpose is to identify trends within those data sets.

Multiple regression analysis is nearly identical to simple linear regression, with the key distinction being the number of predictors (the "x" variables) incorporated in the regression.

- Simple regression analysis involves using a single x variable for each dependent "y" variable. For instance: (x1, Y1).
- Multiple regression, on the other hand, incorporates several "x" variables for each independent variable: (x1)1, (x2)1, (x3)1, Y1).

Multicollinearity

Linear Regression is a supervised learning technique for predicting continuous variables. When building a model, multicollinearity can occur, which is when two or more independent variables are correlated. This can undermine the reliability of statistical conclusions, as high correlation among predictors makes it hard to assess their individual contributions. Thus, it is crucial to remove correlated variables when developing a multiple regression model.

Multicollinearity is a frequent issue encountered in econometrics. As outlined in a preceding post, multicollinearity occurs when there are insufficient observations to accurately assess the impacts of two or more highly correlated variables on the dependent variable. We can visually demonstrate the issue of multicollinearity using Venn diagrams. The Venn diagrams below all depict the same regression model.

$$y = x_1 + x_2 + \epsilon$$

Each circle represents the variability of a single variable in the regression model. Specifically, the circle representing y illustrates the variability of the dependent variable y, while the circle for x_1 shows the variability of variable x_1, and the circle for x_2 indicates the variability of variable x_2. The areas where the circles overlap demonstrate the variation that the

variables share. For example, the overlapping region between variable y and variable x_1 indicates the variation in variable y that can be accounted for by variable x_1.

In the first figure, the circles x_1 and x_2 both intersect with the circle y. However, there is no overlap between the circle x_1 and the circle x_2. In this scenario, variable x_1 and variable x_2 are both correlated with variable y, but the two explanatory variables themselves are uncorrelated. Therefore, it is possible to accurately determine the impact of each explanatory variable (x_1 and x_2) on the dependent variable (y).

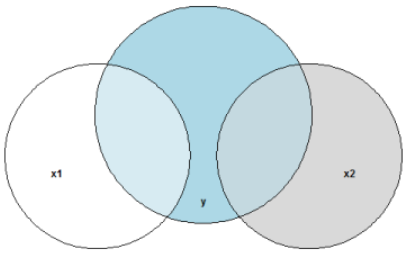


Fig: *No 1 Multicollinearity: Variable x_1 and x_2 are uncorrelated*

In Figure 2 we can see some correlation between the two explanatory variables. That, in Figure 2, some overlap between the circle x_1 and the circle x_2 means that the two variables have some variation in common. It becomes less clear to determine what the effect of one explanatory variable on the dependent variable actually is, i.e. there is some area overlapping all three variables. Although there exists some correlation between variable x_1 and x_2, there is still enough variation left to determine the effect of x_1 and x_2 rather precisely.

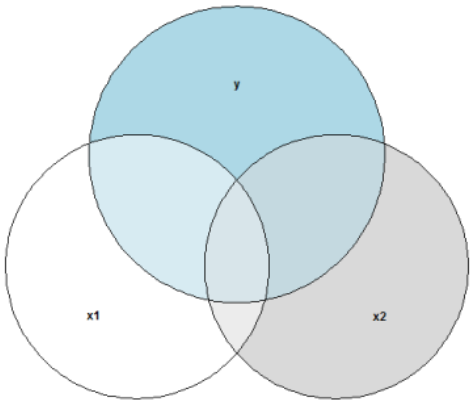


Fig: *No 2 Moderate Multicollinearity: Variable x_1 and x_2 are somewhat correlated*

Moderate multicollinearity is not a significant issue. However, when the correlation between two or more explanatory variables is very high, it becomes increasingly difficult to accurately estimate the individual effect of one explanatory variable on the dependent variable.

Figure 3 illustrates a scenario where the variables x1 and x2 are highly correlated. Consequently, there is progressively less variation that can be attributed solely to one explanatory variable and y. In such cases, we require additional data to accurately determine the effect of a single explanatory variable on the dependent variable. In general, multicollinearity reduces the accuracy of our estimates..

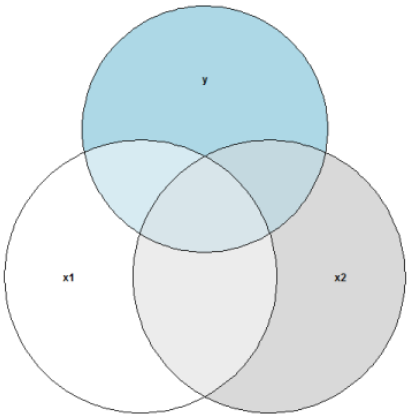


Figure 3: Strong Multicollinearity: Variable x_1 and x_2 are strongly correlated

Multicollinearity does not present issues from a mathematical standpoint as long as there is no perfect multicollinearity. In the context of a Venn diagram, perfect multicollinearity between variables x1 and x2 would imply that the circles representing x1 and x2 are the same, meaning there is complete overlap between them. As a result, one variable can be expressed as a linear function of the other, leaving no variation to be estimated. The Variance Inflation Factor (VIF) is employed to identify multicollinearity within a model, assessing the correlation and degree of correlation among the independent variables in a regression analysis. - A VIF value less than 1 indicates no correlation - A VIF value ranging from 1 to 5 suggests moderate correlation - A VIF value exceeding 5 signals severe correlation. The Variance Inflation Factor (VIF) quantifies the increase in the coefficient of an independent variable caused by collinear relationships with other independent variables. A VIF of 1 signifies that the coefficient of regression is not affected by the other predictors, thus indicating a lack of multicollinearity. As a general guideline, a VIF greater than 5 necessitates further examination, while VIF values above 10 highlight the presence of multicollinearity. Ideally, VIF values should remain below 3..

Regression Code in R

Illustration:

Person	Age	Blood Pressure
1	39	144
2	47	220
3	45	138

4	47	145
5	65	162
6	46	142

```
rm(list=ls())
r=read.csv(file.choose(),header = T)
r
head(r)
tail(r)
t2=lm(r$bp~r$age)
library(lmtest)
coeftest(t2)
summary(t2)
```

Suggested reading

Gupta, A., Sharma, A and Goel, A. (2017). Review of Regression Analysis Models. *International Journal of Engineering Research & Technology (IJERT)*. 2278-0181. Vol. 6 Issue 08.

Uyanik, K.G and Guler.N. A Study on Multiple Linear Regression Analysis. (2013). Social and Behavioral Sciences. 106. 234 – 240.

https://www.wiley.com/en-us/Applied+Regression+Analysis%2C+3rd+Edition-p-9780471170822. (Draper, N.R and Smith.H Applied Regression Analysis)

Regression Analysis for Categorical Data

Vandita Kumari
ICAR-CAZRI, Jodhpur, Rajasthan
Email: vandita.kumari@icar.org.in

1. Introduction

Categorical data is a type of data that represents categories or groups, often qualitative, where values are labels or names rather than numerical measurements. It is used to classify items into distinct groups based on characteristics. The categories in which the data is divided are distinct and with no intermediate values. The data are in the form of labels, not numbers (though numbers can be used as labels). These are called as nominal data eg. gender, color, or brand. When the data is in categories with a meaningful order but no consistent numerical difference between them are called as Ordinal Data. Examples: education levels (high school, bachelor's, master's), survey ratings (poor, fair, good, excellent), or socioeconomic status (low, middle, high)In many real-life applications, the variables of interest are often binary in nature. The term "binary" refers to the two possible values that the response variable can take (e.g., presence/absence, success/failure, on/off), which are typically represented by 0 and 1. By convention, 1 usually indicates the occurrence of the event of interest, while 0 signifies its absence. These two-category response variables are commonly referred to as dichotomous response variables. This situation frequently arises in fields such as agriculture, forestry, environmental science, and epidemiology, where the aim is to explain or predict a binary outcome using specific explanatory variables. For instance, the outcome of a treatment (whether it succeeded or failed), the acceptance status of a candidate for a postgraduate program, or the yes/no and agree/disagree responses obtained from survey questions are all examples of categorical outcomes. In these situations, standard linear regression is not appropriate because categorical outcomes violate assumptions such as normality or the requirement for continuous values. Instead, regression models used depends on the nature of the categorical data (nominal or ordinal) and the number of categories. The frequently used models are logit model, probit model and poisson regression model. The subsequent part addresses these models.

2. Linear probability model:

Let Y represent a binary outcome variable, which is coded as Y = 1 for the desired outcome, referred to as a "success," and Y = 0 for the alternative outcome, labeled a "failure." We denote the probability of the "success" outcome occurring in the population (i.e., Y = 1) using the Greek character.The chance of experiencing a "failure" outcome (i.e. Y = 0) is calculated as 1 - . It is also important to recognize that the average value of Y in the population, known as the "expected value" of Y and represented by E(Y), is simply equal to . Therefore, we have Pr(Y = 1) = E(Y) = and Pr(Y = 0) = 1 .

Let's define a variable, Y, that represents cholesterol levels, where Y = 1 indicates a "high" cholesterol level and Y = 0 signifies a "low" cholesterol level. We may want to explore how Y varies based on explanatory variables. One approach could be to apply the linear probability model, which takes the following form:

$$Y = \alpha + \beta x + \epsilon \, . \qquad (2.1)$$

where, $\epsilon \sim N(0,\sigma^2_\epsilon)$ and ϵ_i and ϵ_j are independent for $i \neq j$. In this context, we represent the likelihood of having "high" cholesterol as a linear function based on the age variable. The model described above indicates a linear association between the dependent variable Y and the independent variable X.

Also, under this assumption that E (ϵ) = 0, it follows that E($Y \mid x$) = $\alpha + \beta x$. The expectation of Y given x may be extended as E ($Y \mid x$) = [{$\pi(x)$} (1)] + [{1- $\pi(x)$} (0)] = $\pi(x)$ Therefore, it is given as $\pi = \alpha + \beta x$ i.e.

$$E(Y \mid x) = \alpha + \beta x \, , \qquad (2.2)$$

When Y has two distinct values, epsilon (ϵ) is likewise dichotomous.

When we considered Y =1, then ϵ = 1- E ($Y \mid x$) =1- ($\alpha + \beta x$) = 1- $\pi(x)$ (which occur with probability $\pi(x)$); and Y= 0 then ϵ = 0 - ($\alpha + \beta x$) = $\pi(x)$ (with probability 1- $\pi(x)$). Since the error is binary, it cannot be even roughly normally distributed. Additionally, the variance of ϵ is not uniform; it varies with x due to its effect on π.

$$V(\epsilon) = E(Y^2) - [E(Y)]^2$$

$$= [1^2 \pi(x) + 0^2\{1-\pi(x)\}] - \{\pi(x)\}^2 = \pi(x) - \{\pi(x)\}^2 = \pi(x)[1-\pi(x)] .$$

Consequently, we observe that the linear probability model has a significant structural flaw when dealing with dichotomous outcomes. In reality, probabilities should be confined to the range of 0 and 1, whereas linear functions can output values across the entire spectrum of real numbers. As a result, in regression analysis: 1. The predicted values generated from this model could exceed the [0,1] range. Since represents a probability, its value must remain within the [0,1] interval. 2. The conventional regression assumption regarding the normality of Y is not upheld, as Y can only take on values of 0 or 1. Hence, the errors should be described by the binomial distribution rather than the normal distribution. Thus, the appropriate approach is to apply a transformation of instead of directly fitting a model for π, we use a transformation of π.

3. Logit model:

Because of structural problem in linear probability model, it is better to study models implying a curvilinear relationship between x and π. The S-shaped curves that obtained (Figure 3.1) are shape for logistic regression curves. A function having this shape is:

$$\pi = \frac{\exp(\mathbf{x}^T\boldsymbol{\beta})}{1+\exp(\mathbf{x}^T\boldsymbol{\beta})} = \frac{1}{1+\exp\{-(\mathbf{x}^T\boldsymbol{\beta})\}} \qquad (3.1)$$

where, $\mathbf{x}^T$ is the vector of k-auxiliary variables linked to each study variable and $\boldsymbol{\beta}$ is the vector of unknown parameters This function is called the logistic regression function.

For this model, the transformation we will apply is the odds of a "successful" outcome, meaning we will model $\pi/(1-\pi)$. The odds are calculated by dividing the likelihood of a "success" by the likelihood of a "failure."

$$\text{odds} = \frac{\Pr(\text{success})}{\Pr(\text{failure})} = \frac{\Pr(\text{success})}{1-\Pr(\text{success})} = \frac{\pi}{1-\pi} . \qquad (3.2)$$

Converting between probabilities and odds is quite straightforward. Keep in mind that odds can range between 0 to ∞. For examples,

1. If $\pi = 0.8$, then the odds are equal to 0.8/(1-0.8) = 0.8/0.2 = 4.

2. If $\pi = 0.5$, then the odds are equal to 0.5/(1-0.5) = 0.5/0.5 = 1. So, if the odds are equal to 1 the chance of "success" is equal to the probability of "failure".
3. If the odds are equal to 0.3 then solving $\pi/(1 - \pi) = 0.3$ gives $\pi = 0.3/1.3 = 0.2308$.

We can view odds as an alternative method for expressing probabilities. It's important to mention that because dividing by zero is prohibited, odds will be undefined when the probability of "failure" is considered.(i.e. $1-\pi$) is 0.

Therefore, we can represent the odds of making response of the above model (3.1) as:

$$\frac{\pi}{1-\pi} = \frac{1}{(1/\pi)-1} = \exp(\mathbf{x}^T\boldsymbol{\beta}) \qquad (3.3)$$

A different and corresponding method to express it is utilizing the logarithm of the odds, referred to as the logit form of the model. By applying the logarithm to both sides of equation (3.3), we arrive at. (3.4)

This transformation of π is reffered as logit transformation. The importance of this transformation lies in that logit(π) is linear in its parameters, it may be continuous, and may range from $-\infty$ to $+\infty$, depending on the range of x. When g(x) = $-\infty$ then π(x) = 0, and when g(x) = ∞ then π(x) =1.

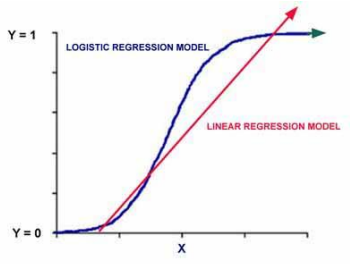


Figure 3.1 Logistic regression model vs linear regression model

4. Logistic Regression

We will strive to measure the correlation between the likelihood of a "success" event and the explanatory factors $X_1, X_2, \ldots, X_k$ based on some sample data. At this point, we presume that

in the there is a relationship between π and a single continuous explanatory variable X and that this relationship can be expressed in the form

$$\text{logit}(\pi) = \log\left[\frac{\pi}{1-\pi}\right] = \beta_0 + \beta_1 X \ . \tag{4.1}$$

This model can be obtained using various statistical software as

$$\log it(\hat{\pi}) = b_0 + b_1 X \ , \tag{4.2}$$

where b_0 and b_1 are the estimated regression coefficients. The estimation process for logistic regression is typically conducted through the statistical technique known as maximum likelihood estimation (MLE). To implement MLE, our goal is to select the β parameters that maximize the likelihood of the observed data. The likelihood function can be expressed as

$$L = P(y_1, y_2, ..., y_N) = \prod_{i=1}^{N} P(y_i) = \prod_{i=1}^{N} \pi^{y_i}(1-\pi^{1-y_i}) \ . \tag{4.3}$$

Now take the logarithm of both sides and substitute in the logistic regression model we get

$$l(\beta) = \sum \left\{ y_i \log \pi_i + (1-y_i)\log\left(1-\pi_i\right)\right\} \ , \tag{4.4}$$

Next, we select values of β that maximize this equation.

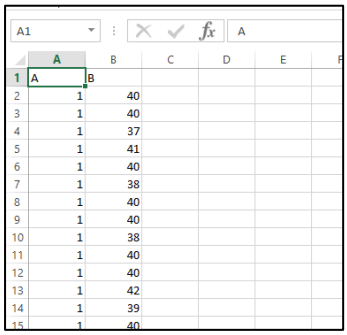
Example 1 - Let's consider the relationship between the gestational age (GAGE) of a human fetus, measured in weeks from conception to birth, and birth weight. The dependent variable in this case is birth weight (BWGHT), which is categorized as 1 for normal weight and 0 for low weight. The dataset comprises information on 24 infants, who have been classified as either low or normal weight at the time of birth, as presented in Table 1.

Table 1: Gestational Ages (in Weeks) of 24 Babies by Birth Weight

Normal Birth Weight (BWGHT = 1)	Low Birth Weight (BWGHT = 0)
40, 40, 37, 41, 40, 38, 40, 40 38, 40, 40, 42, 39, 40, 36, 38, 39	38, 35, 36, 37, 36, 38, 37

Solution

Enter the data in excel sheet and save the data in .csv format



A	B
1	40
1	40
1	37
1	41
1	40
1	38
1	40
1	40
1	38
1	40
1	40
1	42
1	39
1	40

open R and perform the analysis using following commands

>ges<-read.csv(file.choose(),header=T)

rename the columns

>colnames(ges)<-c("Gestational.Ages", "Birth.Weight")

>ges

>str(ges)#structure of the data

to convert integer into factors

>ges$Gestational.Ages<-as.factor(ges$Gestational.Ages)

>str(ges)

#extract the data

>y=ges[,1]; x1=ges[,2]

>ges.data=glm(y ~ x1, data = ges, family = binomial) #fit logistic regression

>summary(ges.data)

The output is obtained as following :

```
Call:
glm(formula = y ~ x1, family = binomial(link = logit), data = ges)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6084  -0.3858   0.2324   0.4402   1.9120
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -48.9085    20.3382  -2.405   0.0162 *
x1            1.3127     0.5409   2.427   0.0152 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Dispersion parameter for binomial family taken to be 1)
Null deviance: 28.975  on 23  degrees of freedom
Residual deviance: 16.298  on 22  degrees of freedom
```

AIC: 20.298
Number of Fisher Scoring iterations: 6

In the previous output, the initial element presented is the call, with R recalling the model we executed, along with the options we selected, and so on. In this instance, we have fitted the model.

$$\text{logit}(\pi) = \beta_0 + \beta_1 \text{ GAGE} \quad (4.5)$$

We then observe the deviance residuals, which indicate how well the model fits. This section of the output displays the distribution of the deviance residuals for the individual cases included in the model. The subsequent part of the output presents the coefficients, their standard errors, the z-statistic, and the corresponding p-values. Both intercept and X_1 are statistically significant. The value of the parameter estimates are b_0 = -48.908 and b_1=1.313, and so our fitted model is

$$\text{logit}(\hat{\pi}) = -48.908 + 1.313 \text{ GAGE}. \quad (4.6)$$

The coefficients (i.e., the 's) in logistic regression are estimated using the maximum likelihood estimation (MLE) method.

Considering our model from example of gestational age and birth weight,

$$\text{logit}(\pi) = \beta_0 + \beta_1 \text{ GAGE} \quad . \quad (4.7)$$

One method to determine if the variable GAGE should be included in the model is to compute the ratio of the estimate to its standard error (which is somewhat similar to the t-statistic used in linear regression).

$$b_1 / s_{b_1} \quad . \quad (4.8)$$

If the null hypothesis that $\beta_1 = 0$ is true, then this statistic has an approximate *standard normal* distribution.Therefore, we can evaluate against values in the normal tables for a specified significance level. Equivalently, we can calculate the Wald statistic, which is the square of this ratio, i.e. $\left(\frac{b_1}{s_{b_1}}\right)^2$. If the null hypothesis that $\beta_1 = 0$ is true, then this statistic has a *chi-squared* distribution with one degree of freedom.

To calculate Wald statistics in R write following code

```
> install.packages("aod") #install package
> library(aod)
> wald.test(b = coef(ges.data), Sigma = vcov(ges.data), Terms = 2)
```

Wald test:

Chi-squared test:
X2 = 5.9, df = 1, P(> X2) = 0.015

In this instance, R computes Wald statistics and presents both the value and the corresponding p-value. For the relationship between gestational age and birth weight, the Wald test statistic for the coefficient tied to the variable GAGE is 5.890, calculated as $[1.313/0.541]^2$. The associated p-value is reported as 0.015, suggesting that the coefficient is significant at the 5% level, though it does not reach significance at the 1% level.

4.1 Interpreting the Model

The logistic regression model can be expressed in three distinct scales: logit, odds, or probability. This allows for interpretation in each of these scales. For instance, look at the relationship between gestational age and birth weight. The coefficient for GAGE in the model can be interpreted accordingly.

$$\text{logit}(\hat{\pi}) = -48.908 + 1.313\,\text{GAGE},$$

A unit change in GAGE raises the log odds of achieving a normal birth weight by 1.313, on average; in other words, each additional week of gestational age boosts the log odds of normal birth weight by 1.313, on average. This model can also be expressed as the odds model by applying the exponent to both sides.i.e.

$$\text{odds} = \frac{\hat{\pi}}{1-\hat{\pi}}$$

$$= e^{-48.908+1.313\,\text{GAGE}} = e^{-48.908} e^{1.313\,\text{GAGE}}.$$

We can also raise the coefficients to a power and understand them as odds ratios. R is capable of performing this calculation.To get the exponentiated coefficients, R code is

```
> exp(coef(ges.data))
 (Intercept)          x1
5.745193e-22  3.716063e+00
```

Therefore, we can understand that a weekly increase in gestational age modifies the odds of achieving normal birth weight by a factor of e^(1.313), which is approximately 3.716. Likewise, we can state that a one-unit rise in GAGE boosts the odds of normal birth weight by [3.716 – 1] × 100%, or 272%.Finally, we can also write the model in the probability scale, i.e.

$$\hat{\pi} = \frac{e^{-48.908+1.313\,\text{GAGE}}}{1+e^{-48.908+1.313\,\text{GAGE}}}.$$

Consequently, we can assess the likelihood of achieving a normal birth weight for any specific gestational age.For the general fitted model equation

$$\text{logit}(\hat{\pi}) = b_0 + b_1 X \quad ,$$

the value of X when $\hat{\pi} = 0.5$ is called the median effective level, i.e. the outcome of interest has 50% chance of occurring. When the odds = 1, i.e. when the logit (log odds) = 0, and this occurs when X = - b0/b1. For the given birth weight data, when gestational age = 48.908/1.313 = 37.25 weeks. At this age, the baby now has a 50% chance of being of normal birth weight.

To apply logistic regression to new data, the following code can be written in R with a decision boundary set at 0.5. If P(y=1|X) > 0.5 then y = 1 otherwise y=0.

R code

```
>newdata = data.frame(x1 =c( 45,35,22))
>fitted.results <- predict(ges.data, newdata, type='response')  # predicts fitted values
>fitted.class <- ifelse(fitted.results > 0.5,1,0)     # to get fitted class of data
```

Output

```
1 2 3
1 0 0
```

Therefore, based on the results, it can be concluded that a gestational age of 45 weeks is associated with a normal birth weight, while the subsequent two data points (35 and 22) fall within the category of low birth weight.

4.2. Interpretation of the constant term

Certain statisticians view the exponent of the constant term as the initial odds. In our case, this represents the odds when GAGE equals 0. However, this interpretation is not particularly relevant in this context, and we consider the constant merely as an extraneous parameter to be included in the model to ensure that the odds are appropriately scaled.

Note: When there are two or more explanatory variables X_1 and X_2, the interpretation of the coefficient β1 as the change in the log odds when X_1 changes by one unit is correct only if X_1 and X_2 are unrelated (i.e. when X_1 changes, X_2 is unaffected).

5. Probit Model

To describe the behavior of a dichotomous dependent variable, it is necessary to select a Cumulative Distribution Function (CDF). One option is the logit model, but in certain applications, the normal CDF has also proven to be effective. The model derived from the

normal CDF is commonly referred to as the Probit model, and it is occasionally called the Normit model.

In Probit model, we use the $\Phi(\cdot)$ function for to model the regression function when the dependent variable is binary, that is,

$$Y = \Phi\left(\mathbf{x}'\beta + \varepsilon\right), \tag{5.1}$$

The conditional expectation of Y given X can be expressed as

$$E(Y|X)=P(Y=1|X)=\Phi(\beta 0+\beta 1X)$$

The equation 7.1 can also be written in the following form

$$\Phi^{-1}\left(Y\right) = \mathbf{x}'\beta + \varepsilon \tag{5.2}$$

In a probit model, the probability is represented by the inverse of the standard normal distribution, which is expressed as a linear combination of the predictors. Here, our link function is $\Phi^{-1}\left(Y\right)$. This function is known as the Probit link. The term "Probit," short for "probability unit," was coined in the 1930s by biologists studying the link between dosage and cure rate.

Also we know that, $\Phi(z)=P(Z\leq z)$, $Z\sim N(0,1)$, therefore in a probit model, the value of $\mathbf{x}'\beta$ is taken to be the z-value of a normal distribution. Higher the values of $\mathbf{x}'\beta$ means that event is more likely to happen. Here, a one unit change in X_i leads to a β_i change in the z-score of Y. In probit analysis, the estimated curve resembles an S-shaped cumulative normal distribution.

To conduct the analysis in R, we will utilize the data from Example 1 and implement the following code.

```
>install.packages("faraway")
>library(faraway)

>ges.probit=glm(y ~ x1, data = ges, family = binomial(link=probit))
```

>summary(ges.probit)

The above code will provide the coefficients with p value. To fit the probit and logit link we will run following code in R

>lines(x, pnorm(-28.7667+0.7725*x), lty=2) #fit probit link

>lines(x,ilogit(-48.9085+1.3127*x)) #fit logit link

While the coefficients obtained from the probit model can vary considerably from those of the logit model, the overall fits are fairly comparable, particularly within the specified data range. In summary, it's important to highlight that the logit link is typically the preferred method for analyzing binary data. Furthermore, one of the benefits of using logit instead of probit is that it involves simpler mathematics, owing to the complexities of the cumulative distribution function (Φ). Additionally, logistic regression offers a more straightforward way to interpret results in terms of odds..

6. Poisson Regression Model

One often encounters situations where the outcome variable is in the form of counts. In statistics, count data in which the observations can take only the non-negative integer values {0, 1, 2, 3, ...}, and where these integers arise from counting rather than ranking. Such data often arises in the of statistics, econometrics, agriculture, biometrical science, environmental science and epidemiology. Poisson regressiis extensively used to model count data. When the response variable represents a count of some relatively rare event, probability model for count data is often Poisson distribution. The response variable represents a count of some relatively rare event, such as defects in a unit of manufactured product, errors or ‘bugs’ in software, or a count of particulate matter or other pollutants in the environment. In such cases, one is interested in modelling the relationship between the observed counts and potentially useful regressors or predictor variables. For example, an engineer could be interested in modelling the relationship between the observed number of defective items(Y) and production condition when the unit was actually manufactured.

The response variable y is a count, such that the observation y = 0, 1, . . . , . Probability model for count data is often Poisson distribution

$$\Pr(Y=y)=\frac{e^{-\mu}\mu^{y}}{y!}, \qquad y=0,1,\ldots$$

where the parameter $\mu > 0$. It is the distribution where, $E(Y) = \mu$ and $Var(Y) = \mu$. That is, mean and variance are equal to the parameter μ.

The Poisson regression model can be written as

$$Y_i = E(Y_i) + \varepsilon_i \ , \qquad i = 1, 2, \ldots, n \ .$$

There are various link functions that are commonly used to Poisson distribution. One of the popular link function for Poisson distribution is the log link

$$g(\mu_i) = \log(\mu_i) = x_i'\beta \ .$$

So, from (1) the relationship between the mean of the response variable and the linear predictor is

$$\mu_i = g^{-1}(x_i'\beta) = e^{x_i'\beta} \ .$$

(6.3)

The log link is particularly attractive for Poisson regression as it ensures that all the predicted values of response will be nonnegative.

Example 2

Let's examine the built-in R dataset ‘mtcar’, which have 32 observations on 11 variables. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

Solution

To perform the analysis in R, we will write the following code

```
> mtcars$gear #count data

 [1] 4 4 4 3 3 3 3 4 4 4 4 3 3 3 3 3 3 4 4 4 3 3 3 3 3 4 5 5 5 5 5 4

#Fit Poisson regression

>poisson_model <- glm(gear ~ mpg + wt, family = poisson(link = "log"), data = mtcars)

> summary(poisson_model)
```

The output is obtained as

```
Call:
glm(formula = gear ~ mpg + wt, family = poisson(link = "log"),
```

```
    data = mtcars)
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.887707   1.236085   1.527    0.127
mpg         -0.005138   0.031361  -0.164    0.870
wt          -0.151230   0.201626  -0.750    0.453

(Dispersion parameter for poisson family taken to be 1)
    Null deviance: 4.4634  on 31  degrees of freedom
Residual deviance: 2.8523  on 29  degrees of freedom
AIC: 110.32
Number of Fisher Scoring iterations: 4

> exp(coef(poisson_model))

(Intercept)         mpg          wt
  6.6042084   0.9948748   0.8596500
```

Suggested Readings

Agresti, A. (1990). Categorical Data Analysis, New York: John Wiley & Sons, Inc.

Hosmer, D.W. and Lemeshow, S. (2000). Applied Logistic Regression, New York: Wiley.

Faraway J. J. (2006). Extending the Linear Model with R, London; Chapman & Hall/ CRC.

McCullagh, P. and Nelder, J. A. (1989). Generalized Linear Models, London: Chapman Hall.

Misra, A.K., Om Prakash, and Ramasubramanian, V. (2004). Forewarning powdery mildew caused by Oidium mangiferae in mango (Mangifera indica) using logistic regression models. *Indian Journal of Agricultural Sciences*. 74(2): 84-87.

Econometric Analysis using Panel Data

Ranjit Kumar Paul
ICAR- Indian Agricultural Statistics Research Institute, New Delhi-110012
ranjitstat@gmail.com

Introduction

Various types of data are commonly used for empirical analysis, such as time series, cross-sectional, and panel data.Time series data consists of observations on a single phenomenon collected over multiple time periods. For instance, this could include GDP figures recorded over several quarters or years. In this type of data, both the numerical values and their sequence are important.On the other hand, cross-sectional data involves gathering values for one or more variables from different sample units or entities at a specific point in time. An example of this would be the crime rates for all 50 U.S. states during a particular year.Panel data sets integrate both time series and cross-sectional data, which increases the number of observations available. For example, if we have data spanning 10 years across 10 countries, this results in 100 observations. While there may not be sufficient data to create a model using only time series or cross-sectional data, the panel format provides enough data for effective estimation.Panel data can be categorized as either balanced or unbalanced. A balanced panel contains observations for each unit across all time periods, while an unbalanced panel has missing data. Panel data helps address unobserved heterogeneity in cross-sectional datasets, which arises from differences in characteristics among survey respondents.

Let us examine a dataset concerning egg production and pricing across 50 districts in India for the years 1990 and 1991. For each year, the information on egg production and prices constitutes a cross-sectional sample. For each district, there are two observations over time regarding egg production and prices. Therefore, in total, we have panel observations on egg production and their respective prices. Panel data can also be referred to by other names, such as pooled data (which is a combination of time series and cross-sectional observations), a mix of time series and cross-sectional data, micro panel data, and longitudinal data (which refers to the examination of a variable or group over time). The regression models derived from such panel data are referred to as panel data regression models (Baltagi, 2001).

Benefits of panel data.(Gujarati, 2004):

1. Panel data, which includes individuals, companies, or states over time, often shows heterogeneity among these units. Estimation methods can account for this by using individual-specific variables.

2. By combining time series with cross-sectional data, panel data offers more informative insights, greater variability, reduced collinearity, increased degrees of freedom, and enhanced efficiency.
3. It allows for a better analysis of change dynamics, making it effective for studying unemployment spells, job turnover, and labor mobility through repeated observations.

4. Panel data can identify and quantify effects that might go unnoticed in pure cross-sectional or time series data. For example, assessing the impact of minimum wage laws on employment and earnings is more comprehensive with successive wage increases.
5. It supports more complex behavioral models, such as economies of scale and technological change, better than pure cross-sectional or time series data.
6. By providing data across many units, panel data can minimize bias from aggregating individuals or firms into broad categories.

Panel Data: An illustrative example

The analysis presented was conducted using data sourced from a renowned study on investment theory suggested by Y. Grunfeld.

Grunfeld was interested in finding out how real gross investment (Y) depends on the real value of the firm (X_2) and real capital stock (X_3). Data on four companies, General electric (GE), General Motor (GM), U.S. Steel (US), and Westinghouse. Data for each company on the preceding three variables are obtained for period 1935-54. Therefore, there are four cross-sectional units and 20 time periods. Now, we have 80 observations. A prior, Y is expected to be positively related to X_2 and X_3.

By pooling or combining all 80 observations, the Grunfeld investment function can be expressed as:

$$Y_{it} = \beta_1 + \beta_2 X_{2it} + \beta_3 X_{3it} + \upsilon_{it} \qquad (1)$$
$$i = 1, 2, 3, 4$$
$$t = 1, 2, \ldots, 20$$

where *i* stands for the i^{th} cross-sectional unit and *t* for the t^{th} time period and it is assumed that the X's are nonstochastic and that the error term follows classical assumptions, namely, $E(\upsilon_{it}) \sim N(0, \sigma^2)$.

Estimation of panel data regression models (Hsiao, 2003; Gujarati, 2004**):**

1. The fixed effects approach.

The estimation of (1) relies on the assumptions we establish regarding the intercept, the slope coefficients, and the error term. There are a few options:
1. Assume that both the intercept and slope coefficients remain constant over time and different contexts, with the error term reflecting variations across time and individuals.
2. The slope coefficients stay consistent while the intercept differs among individuals.
3. The slope coefficients remain fixed, but the intercept changes for both individuals and time.
4. All coefficients, including the intercept and slope coefficients, vary between individuals.
5. Both the intercept and slope coefficients change across individuals and over time.

(i) All coefficients constant across time and individuals

The most straightforward, and perhaps overly simplistic, method is to ignore the spatial and temporal aspects of the pooled data and simply perform the typical OLS regression. In other words, combine the 20 observations for each firm consecutively, resulting in a total of 80 observations for each variable in the model.

The OLS results are as follows

$$\hat{Y} = -63.3041 + 0.1101X_2 + 0.3034X_3$$
$$\text{se} = (29.6124) \quad (0.0137) \quad (0.0493)$$
$$t = (2.1376) \quad (8.0188) \quad (6.1545) \qquad (2)$$
$$R^2 = 0.7565 \qquad \text{Durbin-Watson} = 0.2187$$
$$n = 80 \quad df = 77$$

All coefficients are statistically significant and high, but a low Durbin-Watson statistic suggests potential autocorrelation in the data. The model assumes identical intercepts for GE, GM, US, and Westinghouse, along with uniform slope coefficients for the two X variables across these firms. These restrictive assumptions may lead to a misrepresentation of the relationship between Y and the X variables in the pooled regression.

ii) The slope coefficients are constant but the intercept varies over individuals:

The Fixed Effects or Least-Squares Dummy Variables (LSDV) Regression Model

One approach to account for the uniqueness of each company or each cross-sectional unit is to allow the intercept to differ for each firm while still maintaining that the slope coefficients remain constant across all firms.We write the model as:

$$Y_{it} = \beta_{1i} + \beta_2 X_{2it} + \beta_3 X_{3it} + \upsilon_{it} \qquad (3)$$

The variation in the intercept could be attributed to the approach or beliefs of the management.

The model (3) is referred to as the fixed effects regression model (FEM). The label "fixed effects" arises from the fact that, while the intercept may differ between individuals, the intercept for each individual remains constant over time; in other words, it is invariant to time.

This can be accomplished using the dummy variable method.Therefore we write the model as

$$Y_{it} = \alpha_1 + \alpha_2 D_{2i} + \alpha_3 D_{3i} + \alpha_4 D_{4i} + \beta_2 X_{2it} + \beta_3 X_{3it} + \upsilon_{it} \qquad (4)$$

where $D_{2i} = 1$ if the observation from GM, 0 otherwise; $D_{3i} = 1$ if the observation from US, 0 otherwise; and $D_{4i} = 1$ if the observation belongs to WEST, 0 otherwise. Here α_1 representing the intercept of GE and $\alpha_2, \alpha_3, \text{and} \alpha_4$, the differential intercept coefficients, tell by how much the intercepts of GM, US, and WEST differ from the intercept of GEAs we are utilizing dummy variables to calculate the fixed effects, this model is often referred to as the least-squares dummy variable (LSDV) model.

The results is as follows:

$$\hat{Y}_{it} = -245.7924 + 161.5722 D_{2i} + 339.6328 D_{3i} + 186.5666 D_{4i} + 0.1079 X_{2i} + 0.3461 X_{3i}$$

se = (35.8112) (46.4563) (23.9863) (31.5068) (0.0175) (0.0266)

t = (6.8635) (3.4779) (14.1594) (5.9214) (6.1653) (12.9821)

$R^2 = 0.9345$ d =1.1076 df = 74 (5)

All the estimated coefficients are significantly important on their own, and the intercept values for the four companies show statistical differences. The variation in intercepts could be attributed to distinct characteristics of each company, like variations in management approach or the skill level of management.Judged by the statistical significance of the estimated coefficients, and the fact that the R^2 value has increased substantially we can conclude that (5) is better than (2). The Durbin-Watson *d* value is much higher; suggesting that model (2) was miss-specified.

We can also conduct a formal examination of the two models. Regarding (5), model (2) is considered a restricted model because it enforces a shared intercept across all companies. Consequently, we can apply the restricted F test.Using the formula we get

$$F = \frac{(R^2_{UR} - R^2_R)/3}{(1 - R^2_{UR})/74} = 66.9980 \qquad (6)$$

where the restricted value is from (2) and the unrestricted is from (5).

The F value of 66.998 is evidently significant, indicating that the restricted regression (2) appears to be invalid.

The Time Effect

Just as we utilized dummy variables to capture individual effects, we can also incorporate time effects, indicating that the Grunfeld investment function varies over time. To address this situation, we introduce time dummies for each year. From the results of the regression, we conclude that none of the specific time dummies were statistically significant individually. We have previously established that the individual company effects were statistically significant, while the individual year effects were not.

(iii) Slope coefficients constant but the intercept varies over individual as well as time

To explore this option, we can integrate (5) with the time effect model in the following manner:

$$Y_{it} = \alpha_1 + \alpha_2 D_{GMi} + \alpha_3 D_{USi} + \alpha_4 D_{WESTi} + \lambda_0 + \lambda_1 DUM35 + ... + \lambda_{19} DUM53 + ... + \beta_2 X_{2i} + \beta_3 X_{3i} + \upsilon_{it} \qquad (7)$$

When conducting this regression, we observe that the company dummies and the coefficients of the X variables are all statistically significant on their own, but none of the time dummies reach significance. Essentially, we revert to (5).

(iv) All coefficients vary across individuals

In this scenario, we consider that the intercepts and slope coefficients vary for every individual or cross-sectional unit. This implies that the investment functions for GE, GM, US, and WEST are distinct from one another. We can readily adapt our LSDV model to address this circumstance. In this case, we multiply each of the company dummy variables by each of the X variables.That is, we estimate the following model:

$$Y_{it} = \alpha_1 + \alpha_2 D_{2i} + \alpha_3 D_{3i} + \alpha_4 D_{4i} + \beta_2 X_{2it} + \beta_{3it} + \gamma_1 (D_{2i} X_{2it}) + \gamma_2 (D_{2i} X_{3it}) + \gamma_3 (D_{3i} X_{2it}) + ... + \upsilon_{it} \quad (8)$$

The γ's are the differential slope coefficients, just as $\alpha_2, \alpha_3, \text{and} \alpha_4$ are the differential intercepts. If one or more of the γ coefficients are statistically significant, it will tell us that one or more slope coefficients are different from the base group. If all the coefficients for the differential intercept and differential slope are statistically significant, we can infer that the investment functions vary among the four companies.

Estimation of panel data regression models:

The random effects approach

The fixed effects or LSDV model can consume a lot of degrees of freedom when dealing with multiple cross-sectional units. If the dummy variables indicate a lack of understanding regarding the (true) model, why not convey this uncertainty via the disturbance term? This is exactly the method proposed by supporters of the so-called error components model (ECM) or random effects model (REM).

To start with (3):

$$Y_{it} = \beta_{1i} + \beta_2 X_{2it} + \beta_3 X_{3it} + \upsilon_{it} \quad (9)$$

Instead of treating β_{1i} as fixed, we assume that it is a random variable with a mean value of β_1. And the intercept value for an individual company can be expressed as

$$\beta_{1i} = \beta_1 + \varepsilon_i \qquad i = 1, 2, \ldots, N \quad (10)$$

where ε_i is a random error term with a mean value of zero and variance σ_ε^2.

The sample consists of four firms selected from a significantly larger group of such companies, and they share a common mean value for the intercept, with the unique variations in each company's intercept values represented in the error term.

Substituting (10) into (9), we get:

$$Y_{it} = \beta_1 + \beta_2 X_{2it} + \beta_3 X_{3it} + \varepsilon_i + \upsilon_{it}$$

$$Y_{it} = \beta_1 + \beta_2 X_{2it} + \beta_3 X_{3it} + \omega_{it} \quad (11)$$

where $$\omega_{it} = \varepsilon_i + \upsilon_{it} \quad (12)$$

The composite error term is made up of two parts: the individual-specific error component and the error component that combines both time series and cross-section elements.The usual assumptions made by ECM are that

$$\begin{aligned} &\varepsilon_i \sim N(0, \sigma_\varepsilon^2) \\ &\upsilon_{it} \sim N(0, \sigma_\upsilon^2) \\ &E(\varepsilon_i \upsilon_{it}) = 0 \qquad E(\varepsilon_i \varepsilon_j) = 0 \qquad (i \neq j) \\ &E(\upsilon_{it} \upsilon_{is}) = E(\upsilon_{it} \upsilon_{jt}) = E(\upsilon_{it} \upsilon_{js}) = 0 \qquad (i \neq j; t \neq s) \end{aligned} \quad (13)$$

In other words, the separate error components do not exhibit correlation with one another and are also not autocorrelated across both cross-sectional and time series units.

As a result it follows that

$$E(\omega_{it}) = 0 \quad (14)$$

$$var(\omega_{it}) = \sigma_\varepsilon^2 + \sigma_\upsilon^2 \quad (15)$$

As (15) shows, the error term ω_{it} is homoscedastic. However, it can be shown that ω_{it} and ω_{is} are correlated; The error terms for a specific cross-sectional unit at two distinct moments in time are correlated. The correlation coefficient is as follows:

$$corr(\omega_{it}, \omega_{is}) = \frac{\sigma_\varepsilon^2}{\sigma_\varepsilon^2 + \sigma_\upsilon^2} \quad (16)$$

If the correlation structure is not considered, estimating (11) using OLS will lead to inefficient estimators. In this case, the best approach is to use the Generalized Least Squares (GLS) method.

Fixed Effects (LSDV) versus Random Effects Model

Panel data models analyze individual-specific effects, time effects, or both, classified as fixed or random effects. A fixed effect model checks if intercepts differ across groups or time, while a random effect model looks at variations in error variances. One-way models use one set of dummy variables (e.g., firm), and two-way models use two sets (e.g., firm and year). If the error component and independent variables are uncorrelated, the error correction model (ECM) may be suitable; if correlated, the fixed effects model (FEM) is preferable. The choice between FEM and ECM can be made by:

When T (the number of time series observations) is large and N (the number of cross-sectional units) is small, the parameter estimates from the Fixed Effects Model (FEM) and the Error Component Model (ECM) are likely to be similar, with the choice of method favoring computational convenience, typically making FEM preferable.In contrast, when N is large and T is small, the estimates can differ significantly. ECM treats individual effects as random, while FEM considers them fixed. If the individual error component is correlated with regressors, ECM estimates can be biased, whereas FEM estimates remain unbiased. Thus, under these conditions, if ECM's assumptions hold, its estimators tend to be more efficient than those of FEM.
The Hausman test, established in 1978, helps determine whether to use Fixed Effects Models (FEM) or Random Effects Models (ECM). Its null hypothesis posits no significant difference between the two estimators, and the test statistic follows an asymptotic chi-square distribution. If the null hypothesis is rejected, it indicates that ECM may be inappropriate, suggesting that FEM is more suitable, with statistical inferences dependent on the error component in the sample.

Conclusion

Panel data combines inter-individual differences and intra-individual dynamics, offering benefits over cross-sectional or time-series data. It captures the complexities of human behavior more effectively, allowing for more precise estimation of model parameters. Additionally, panel data provides greater degrees of freedom and sample variability and reduces omitted variable bias, helping to reveal dynamic relationships.

R code

```
library(plm)
#the fixed effects model (within),
#the pooling model (pooling),
#the first-difference model (fd),
#the between model (between),
#the error components model (random).
grun.fe <- plm(I ~ F + C, data = mydata, effect="twoways", model = "within")
grun.re <- plm(I ~ F + C, data = mydata,effect="twoways", model = "random")
grun.ols <- lm(I ~ F + C, data = mydata)
fixed<-fixef(grun.fe, effect="time")
```

```
summary(fixed)
## F test for fixed effects vs ols model
pFtest(grun.fe, grun.ols)
## Breusch-Pagan test for testing the significance of effect i.e. cross section effect and time effect
plmtest(grun.fe, effect="individual", type="bp")
#least square dummy variable model
lsdv.model <- lm(I ~ F + C+(-1)+ as.factor(YEAR)+as.factor(FIRM), data=mydata, na.action=na.omit)
summary(lsdv.model)
#Hausman test: if p value is less than 0.05 then fixed effect model is better
phtest(grun.fe, grun.re)
summary(grun.re)
```

References

- Baltagi, B.H. (2001), *Econometric Analysis of Panel Data*, Second edition, New York: Wiley.
- Gujarati, D.N., (2004). *Basic Econometrics*, Fourth Edition, Tata Mcgraw-Hill Edition.
- Hausman, J.A. (1978), "Specification Tests in Econometrics", *Econometrica*, 46. 1251-71.
- Hsiao, C. (2003), *Analysis of Panel Data*, 2nd edition, Cambridge: Cambridge University Press (Econometric Society monograph no. 34).

Nonlinear Growth Models

Mrinmoy Ray
AKMU, ICAR-Indian Agricultural Research Institute, New Delhi-110012
Email: mrinmoy4848@gmail.com

1. Introduction

Growth is described as "a permanent increase in size and volume that results from differentiation and distribution in plants or animals." A model serves as a schematic depiction of a system's concept, an imitation, or a collection of equations that illustrates a system's behavior. A model is additionally defined as "a representation of an object, system, or concept in a form different from the entity itself." Its main goal is usually to assist in explaining, understanding, or enhancing a system's performance.

TYPES OF MODELS

Models can be categorized into various groups or types based on their intended purpose. Some of these include:

a. Statistical models: These models illustrate the connections between variables. Statistical methods are used to assess relationships within a system through these models. For example, regression models and time series models fall into this category.

b. Mechanistic models: These models not only clarify the relationships between variables but also provide insight into the underlying mechanisms (explaining how dependent variables are influenced). The foundation of these models is based on physical selection.

c. Deterministic models: These models predict the exact value of a dependent variable. They feature well-defined coefficients as well.

d. Stochastic models: Each output in these models includes a probability element. For any given set of inputs, different outputs are presented alongside their associated probabilities. These models characterize the state of the dependent variable at a specific rate.

e. Dynamic models: These models treat time as a variable. Both the dependent and independent variables maintain constant values over a specified time frame.

f. Static models: Time is not treated as a variable in these models. The values of dependent and independent variables remain consistent over time.

g. Simulation models: Typically, these are computer models that mathematically represent real-world systems. The main objective of crop simulation models is to estimate agricultural yield based on weather and soil conditions and crop management practices. These models utilize one or more sets of differential equations to evaluate rate and state variables over time, usually from planting to harvest maturity or final harvest.

Statistical Modelling

A core issue in statistics involves creating models from a sample of observations and drawing conclusions based on those models. Vast quantities of data related to crop yield/productivity, import and export of different agricultural products, and other areas are being gathered continuously over time across nearly all fields of agriculture, including animal sciences and fisheries. A characteristic of data is subsequent observations are interrelated. Each observation of this observed data series, Y_t, is considered as a realization of a stochastic process $\{Y_t\}$, which is a family of random variables $\{Y_t, t \in T\}$, where $T = \{0, \pm1, \pm2, \ldots\}$, and applying standard time-series approach to develop an ideal model will adequately represent the set of realizations and also their statistical relationships in a satisfactory manner. Predicting time-series data is essential for decision-makers and planners. In recent decades, a novel area called "Nonlinear time-series modelling" has developed. There are fundamentally two strategies available: parametric and nonparametric. Clearly, the former should be utilized if we are confident about the functional structure in a specific context; otherwise, the latter can be applied.

Parametric and Nonparametric Approaches

In recent decades, regression analysis has become increasingly popular as a method for statistical modeling and data analysis. It illustrates how a response variable relates to one or more predictor variables. The primary aim is to represent the average of the response variable based on the predictor variables. The general structure of the regression model is as follows:

$$Y = m(X) + \varepsilon$$

Where Y represents the response variable, $m(X) = E(Y|X)$ is the mean response or regression function and ε is the error. The function m(X) in regression is typically not known, and the goal is to find an appropriate estimate for m(X) by utilizing a set of observed data.

In linear regression, it is presumed that the average of the response variable Y is a linear function of the predictor variable(s) X, expressed in the form

$$E(Y|X) = X\beta$$

i.e. $m(X)$ is linear in parameters. The parameter vector is typically determined using the Method of least squares. In nonlinear regression, it is assumed that the mean of the response variable is a nonlinear function of the predictor variable (s) X of the form

$$E(Y|X)=m(X,\beta)$$

i.e. $m(X)$ is nonlinear in parameters. Typically, there will not be a closed-form solution for the estimates, and iterative methods will be necessary for parameter estimation.

A parametric regression model (whether linear or nonlinear) operates under the assumption that the structure of m is known except for some unknown parameters, and the characteristics of the regression function depend entirely on these parameters. It can often be challenging to determine the most suitable functional form merely by examining the data. There may be instances where an appropriate parametric form does not exist to represent the regression function. In such situations, the nonparametric regression method becomes valuable as it does not impose stringent assumptions regarding the shape of the regression function. A nonparametric regression model primarily assumes that m belongs to an infinitely large set of functions. One downside of this earlier approach is that it typically depends on certain assumptions concerning the smoothness of the function being estimated, which might not hold true in practice. Consequently, the data being analyzed could be subjected to oversmoothing.

Linear Model

A mathematical model is defined as an equation or a collection of equations that depict the behavior of a system. Such models can be categorized as 'linear' or 'nonlinear.' A linear model is one in which all variables are expressed in a linear manner.

Nonlinear Models

Any form of statistical analysis that earnestly takes into account concepts from an established body of knowledge is likely to lead to a 'Nonlinear model.' Such models are essential for grasping the intricate relationships between different variables. A 'nonlinear model' is defined as one where at least one of the parameters exhibits a non-linear behavior. More precisely, in a 'nonlinear model,' at least one derivative concerning a parameter must involve that specific parameter.

- Examples of a nonlinear model are:

$$Y(t) = \exp(at+bt^2) \quad (1a)$$

$$Y(t) = at + \exp(-bt) \quad (1b)$$

Note. Certain authors refer to the phrase 'intrinsically nonlinear' to describe a nonlinear model that can be converted into a linear model through a specific transformation.For example, the model given by Eq. (1a) is 'intrinsically nonlinear' in view of the transformation $X(t) = \log_e Y(t)$.

a. Malthus Model:

Thomas R. Malthus, an English scholar, introduced a mathematical framework for population growth in 1798. Despite its straightforwardness, this model has become the basis for most subsequent biological population modeling. His work, "An Essay on the Principle of Population," features a commendable examination of the limitations inherent in mathematical modeling and should be essential reading for anyone serious about the topic. Malthus noted that, in the absence of environmental or societal limitations, human populations seemed to double every twenty-five years, irrespective of their initial size. In other words, he suggested that populations grew at a constant ratio over specific time frames, and that this ratio remained unchanged by the size of the population when there were no restrictions. According to Malthus, if a population of 100 individuals increased to 135 individuals over a period of about five years, then a population of 1000 individuals would rise to 1350 individuals in the same amount of time. The model proposed by Malthus is an illustration of a one-variable, one-parameter model. The element we are interested in observing is called a variable. These typically progress over time. Parameters are values known to the modeller prior to the construction of the model. They

are often constants, although a parameter can vary over time. In the Malthusian model, the variable is population, while the parameter is the growth rate of the population.

If N(t) represents the population size or biomass at time t, and r indicates the intrinsic growth rate, then the change in population size over time is expressed by

$dN/dt = rN$

Therefore, $N(t) = N_o \exp(rt)$

Note: The Malthus model can be utilized to illustrate the growth of basic organisms that start to expand through the binary division of cells.

Drawback: $N(t) \to \infty$ as $t \to \infty$, which will not happen in reality.

Malthus anticipated that if population growth were left unregulated, it would soon exceed the capacity of the environment, leading to overpopulation and associated social issues.s.

b. Monomolecular Model:

The monomolecular model operates on the premise that the carrying capacity is one, indicating that the highest possible level of disease is also one. Therefore, the severity or incidence of the disease is expressed as a fraction. Diseased plant tissue can only have a value ranging from zero (healthy) to one (fully diseased). Additionally, it assumes that the absolute rate of change is directly proportional to the amount of healthy tissue, represented as (1-y).

It outlines a development process where the growth rate at any given moment is considered to be proportional to the resources that are still available to be acquired.

$dN/dt = r(K-N)$,

where K is the carrying capacity.

or $N(t) = K - (K-N_o) \exp(-rt)$

Drawback: No point of inflexion.

c. Logistic Model:

The logistic model was created by Belgian mathematician Pierre Verhulst in 1838, who proposed that the growth rate of a population might be constrained, meaning it could be influenced by population density. As the population size, N, increases, the growth rate diminishes, reaching zero when N equals K. The parameter K represents the maximum potential for population growth and is referred to as the carrying capacity. It is often understood as the quantity of resources available, measured by the number of organisms those resources can sustain. If the population surpasses K, the growth rate turns negative and the population starts to decline.

The differential equation represents this model:

$$dN/dt = rN\,(1-N/K) \qquad (1)$$

Therefore, $N(t) = K/[1+(K/N_o-1)\exp(-rt)]$. The graph of N(t) versus t is elongated S-shaped and the curve is symmetrical about its point of inflexion.

d. Gompertz Model

This is yet another model displaying sigmoid behavior that has proven to be quite beneficial in biological research. Benjamin Gompertz introduced the Gompertz curve to assess human mortality (Gompertz, B. "On the Nature of the Function Expressive of the Law of Human Mortality, and on a New Mode of Determining the Value of Life Contingencies." Phil. Trans. Roy. Soc. London 123, 513-585, 1832). An early account of applying this equation to growth processes was provided by Charles Winsor (1932). However, in contrast to the logistic model, this one does not feature a symmetric inflection point.

This model's differential equation is

$$dN/dt = rN \log_e (K/N) \qquad (2)$$

or $N(t) = K \exp[\log_e (N_o / K) \exp(-rt)]$

e. Richards Model:

The Richards curve, often referred to as the generalized logistic function, is a widely used growth model capable of accommodating various S-shaped growth patterns. Versions with either 4 or 5 parameters are frequently utilized. The logistic curve exhibits symmetry around

its inflection point. In 1959, Richards added an extra parameter to address the issue of asymmetrical growth patterns.

This model is given by

$$N(t) = K N_o / [N_o + (K^m - N_o^m) \cdot \exp(-rt)]^{1/m} \quad (4)$$

However, unlike the earlier models, this model has four parameters.

Drawback. Number of parameters are more.

MIXED-INFLUENCE MODEL:

This is a mixture of 'Monomolecular' and 'Logistic' Models. It is given by

dN/dt = r (K-N) +s N (1-N/K),

Fitting of Nonlinear Models

The models described earlier have been established in a deterministic manner. This is clearly not realistic, so we substitute these deterministic models with statistical models by incorporating an error term on the right side and making suitable assumptions. This results in a 'Nonlinear statistical model.' The 'Method of least squares' can be employed to estimate parameters in nonlinear regression, just as it is used in linear regression. However, minimizing the residual sum of squares leads to normal equations that contain nonlinear parameters. Since precise solutions for nonlinear equations are unattainable, iterative methods are utilized to derive approximate analytical solutions..

- Four main methods are:
  i) Linearization (or Taylor Series) method
  ii) Steepest Descent method
  iii) Levenberg-Marquardt's method
  iv) Do not use Derivatives method

Draper and Smith examine the details of these techniques, along with their advantages and disadvantages (1998). Neither the Linearization nor the Steepest descent methods are flawless. The Levenberg-Marquardt method is the most commonly utilized technique for calculating

nonlinear least squares estimates. This approach strikes a balance between the two other methods, effectively integrating their best aspects while steering clear of their major shortcomings. It's advantageous because it nearly always converges and maintains a consistent pace at the latter stages of the iterative process.

Choice of Initial Values

Every nonlinear estimation method necessitates initial parameter values, and choosing appropriate starting values is essential. Nonetheless, there is no universally accepted approach for acquiring preliminary estimates. Utilizing prior information is the most straightforward way to generate initial guesses. Estimates obtained from prior experiments, established values for comparable systems, and values derived from theoretical insights all contribute to forming optimal first guesses.

Other techniques include:

(i) Linearization:

By disregarding the error term, evaluate the model's structure to determine if it can be transformed into a linear format through some type of transformation. In these instances, linear regression may be employed to derive initial values.

(ii) Solving a system of equations:

If there are p parameters, replace p sets of observations in the model while ignoring the error. If feasible, resolve these equations for the parameters. Values with widely separated xi tend to yield the best results.

R Code

Monomolecular growth model

```
 z=read.csv(file.choose(), header=TRUE)
head(z)
kk=data.frame(z)
grz1=nls(y~k-(k-y0)*exp(-r*t),data=kk,  start=list(k=1 ,y0=0.03,r=0.1))
summary(grz1)
 fitted=kk$y-resid(grz1)
kkk=data.frame(fitted)
MSE.nn <- sum((kk$y- kkk)^2)/nrow(kkk)
```

```
plot_colors <- c("blue","red")
plot(kk$y,type="o", col=plot_colors[1], ylim=c(0,1),axes=FALSE, ann=FALSE)
axis(1, at=1:20, lab=c(0:19))
axis(2, las=1, at=0.2*0:5)
box()
lines(fitted,type="o", pch=22, lty=2,col=plot_colors[2])
title(main="Actual vs predicted",col.main="red", font.main=4)
title(xlab= "Time", col.lab=rgb(0,0.5,0))
title(ylab= "Growth", col.lab=rgb(0,0.5,0))
legend("topleft",c("actual", "predicted"),cex=0.8, col=plot_colors, pch=21:22, lty=1:2);
zz=resid(grz1)
predicted= 0.99651-(0.99651-0.08844)*exp(-0.26727*20)
Gompertz model
z=read.csv(file.choose(), header=TRUE)
 head(z)
 kk=data.frame(z)
gr1=nls(y~k*exp(log(y0/k)* exp(-r*t)),data=kk,  start=list(k=50,y0=11.72,r=0.1))
summary(gr1)
fitted=kk$y-resid(gr1)
kkk=data.frame(fitted)
MSE.nn <- sum((kk$y- kkk)^2)/nrow(kkk)
plot_colors <- c("blue","red")
plot(kk$y,type="o", col=plot_colors[1], ylim=c(0,35),axes=FALSE, ann=FALSE)
axis(1, at=1:38, lab=c(0:37))
axis(2, las=1, at=5*0:8)
box()
lines(fitted,type="o", pch=22, lty=2,col=plot_colors[2])
title(main="Actual vs predicted",col.main="red", font.main=4)
title(xlab= "Time", col.lab=rgb(0,0.5,0))
title(ylab= "Growth", col.lab=rgb(0,0.5,0))
legend("topleft",c("actual", "predicted"),cex=0.8, col=plot_colors, pch=21:22, lty=1:2);
logistic model
z=read.csv(file.choose(), header=TRUE)
 head(z)
 kk=data.frame(z)
gr2=nls(y~k/(1+(k/y0-1)* exp(-r*t)), data=kk,  start=list(k=50,y0=11.72,r=0.1))
summary(gr2)
fitted=kk$y-resid(gr2)
kkk=data.frame(fitted)
MSE.nn <- sum((kk$y- kkk)^2)/nrow(kkk)
plot_colors <- c("blue","red")
plot(kk$y,type="o", col=plot_colors[1], ylim=c(0,35),axes=FALSE, ann=FALSE)
axis(1, at=1:38, lab=c(0:37))
axis(2, las=1, at=5*0:8)
box()
lines(fitted,type="o", pch=22, lty=2,col=plot_colors[2])
title(main="Actual vs predicted",col.main="red", font.main=4)
```

```
title(xlab= "Time", col.lab=rgb(0,0.5,0))
title(ylab= "Growth", col.lab=rgb(0,0.5,0))
legend("topleft",c("actual", "predicted"),cex=0.8, col=plot_colors, pch=21:22, lty=1:2);
```

Suggested Readings

Banks, H. T. (1994). Growth and diffusion phenomena: Mathematical frameworks and applications. Springer.

Barro, R. J., & Sala-i-Martin, X. (2004). Economic growth (2nd ed.). MIT Press.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian data analysis (3rd ed.). CRC Press.

Pinheiro, J. C., & Bates, D. M. (2000). Mixed-effects models in S and S-PLUS. Springer.

Ratkowsky, D. A. (1990). Handbook of nonlinear regression models. Marcel Dekker.

Seber, G. A. F., & Wild, C. J. (2003). Nonlinear regression. Wiley.

Regularization Techniques in Regression

Bishal Gurung[1], KN Singh[2], Achal Lama[2], Gopal Saha[2] and Biwash Gurung[3]
[1]North_Eastern Hill University, Shillong, India
[2]ICAR-Indian Agricultural Statistics Research Institute, New Delhi, India
[3]School of Agricultural Sciences, G D Goenka University, Sohna, India
Email: vsalrayan@gmail.com

1. Introduction

The multiple regression model is frequently employed to forecast future outcomes or to examine the connections between a response variable and several predictor variables. Think about a linear regression model that includes p predictors $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_p}$,

$$\mathbf{y} = \mathbf{X\beta} + \boldsymbol{\varepsilon} \qquad 1.1$$

where $\mathbf{X}$ is known n×p model matrix with rank $\rho(\mathbf{X}) = p$, $\mathbf{y}$ is a n×1 response vector and $\boldsymbol{\varepsilon}$ is a n×1 vector of errors with $E(\boldsymbol{\varepsilon})=\mathbf{0}$ and $\text{cov}(\boldsymbol{\varepsilon})=\sigma^2\mathbf{I}$. The ordinary least squares (OLS) estimator $\hat{\boldsymbol{\beta}}_{LS} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ is most commonly used for estimating the parameters ($\boldsymbol{\beta}$). This estimate is derived from minimizing the residual sum of squares, and the predicted responses are provided by $\hat{\mathbf{y}} = \hat{\beta}_0 + \hat{\beta}_1\mathbf{x}_1 + \cdots + \hat{\beta}_p\mathbf{x}_p$.

In regression analysis, we typically presume that included regressor variables are important. The focus is on ensuring the model's functional form is correct and underlying assumptions are met. Nevertheless, in reality, analysts usually possess a selection of potential regressors that presumably contain significant factors, but they must identify the best subset for the model. This process of identifying the most relevant variables is known as variable selection.

Build a regression model with a subset of available regressors balancing two competing goals: (1) including as many relevant regressors as possible to capture the underlying relationships and improve predictions, while (2) keeping the model parsimonious by including as few regressors as necessary to avoid complexity and potential overfitting.

Beside this the criteria for evaluating the quality of a model are as follows:

1. 1. The model needs to effectively predict or anticipate future data points.
2. 2. A less complex model is generally favored as it offers clearer insights into how the response variable relates to the predictors. Keeping things simple is essential when

handling numerous predictors, as it is preferable to gather the maximum information about the response variable with the minimum number of predictors.

The two primary goals of a linear regression model are: (1) achieving high prediction accuracy, where predicted values closely match true values, and (2) minimizing model complexity.

Although Ordinary Least Squares (OLS) is a commonly employed estimation method, it frequently underperforms in prediction accuracy and model interpretation. Moreover, with big data analysis involving high-dimensional data (e.g., microarray data) with numerous independent variables, OLS becomes inadequate, highlighting the need for alternative regression techniques better suited for handling such complex data.

To tackle the drawbacks of OLS regression, two common approaches are utilized: subset selection and shrinkage methods. Subset selection results in models that are easy to interpret, but it may be unstable because of its discrete characteristics, where variables are either included or left out, and slight alterations in the data can result in significantly different models, which can ultimately undermine prediction accuracy. Among shrinkage methods, ridge regression is widely used, which minimizes the residual sum of squares while constraining the L_2-norm of the coefficients.

Ridge regression which is a continuous shrinkage method, improves prediction performance by bias-variance tradeoff, but it can't produce parsimonious models since it doesn't set coefficients to zero. The LASSO (Least Absolute Shrinkage and Selection Operator) overcomes the limitations of traditional regression methods by incorporating the advantages of ridge regression and subset selection. By applying an L_1 penalty to the regression coefficients, LASSO achieves both continuous shrinkage and automatic variable selection. This dual approach improves prediction accuracy and enhances the interpretability of the model.

Tibshirani (1996) and Fu (1998) found lasso and ridge regression have comparable prediction performance, with neither uniformly outperforming the other. Nonetheless, the capability of lasso to conduct variable selection increases its attractiveness in contemporary data analysis. Lasso, despite its effectiveness, has drawbacks such as limiting selection to n variables when $p > n$ and struggling with grouped variables (highly correlated predictors), often picking only

one. In cases of multicollinearity (n > p), ridge regression may perform better than lasso in prediction.

To tackle this issue, Zou and Hastie (2005) introduced a novel approach called Elastic Net. It performs at least as effectively as the lasso when the lasso is at its best, and it can resolve the lasso's limitations mentioned earlier.

3. Regularization

Regularization is a technique that overcomes overfitting by retaining all features or regressors but shrinking their corresponding parameter values (coefficients) by introducing a penalty term, which adds bias. This helps to reduce the model's complexity and improve its generalizability.

The principles of regularization can be articulated as follows:

1. ***Stabilization.*** The adjusted regularized problem seeks to achieve a singular solution and lower sensitivity in comparison to the initial ill-posed problem.
2. ***Meaningful solution.*** While simple regularization methods can stabilize numerical computations and yield reasonable solutions in many cases, numerous problems require more specialized techniques that leverage additional information or prior knowledge about the solution's physical significance to achieve better results.

A general penalty-type regularization can be viewed as a optimization challenge which can be formulated as following,

$$\min_x L(F(x),y)+\lambda R(x) \qquad 2.1$$

where L represents a loss function with non-negative values that assesses the deviation between the data y and the model F(x). R is a penalty function that is also non-negatively valued and indicates the desired constraints on the solution x. The scalar $\lambda > 0$ is the factor that manages the bias-variance tradeoff. In the literature, this scalar is referred to as the penalty parameter, regularization parameter, or smoothing parameter.

3. Ridge regression

In regard to model 1.1, following a transformation of location and scale, it is possible to assume that the response variable is centered and the predictor variables are standardized.Then,

$$\sum_{i=1}^{n} y_i = 0, \quad \sum_{i=1}^{n} x_{ij} = 0 \quad \text{and} \sum_{i=1}^{n} x_{ij}^2 = 1, \quad \text{for } j=1,2,...,p$$

Due to the high variability of the estimated regression coefficients, regularization methods are applied to manage this variance. Consequently, instead of solely minimizing the sum of squared residuals like in ordinary least squares (OLS), ridge regression introduces an additional constraint known as the penalized residual sum of squares.

$$L(\lambda_2, \boldsymbol{\beta}) = |\mathbf{y} - \mathbf{X}\boldsymbol{\beta}|^2 + \lambda_2 |\boldsymbol{\beta}|^2 \qquad 3.1$$
$$= \sum_{i=1}^{n} (y_i - \mathbf{x}_i'\boldsymbol{\beta}) + \lambda_2 \sum_{j=1}^{p} \beta_j^2$$

Differentiating equation 3.1 concerning the parameters results in the normal equation $(\mathbf{X}'\mathbf{X} + \lambda_2 \mathbf{I})\hat{\boldsymbol{\beta}}_R = \mathbf{X}'\mathbf{y}$ and the ridge estimator is given by $\hat{\boldsymbol{\beta}}_R = (\mathbf{X}'\mathbf{X} + \lambda_2 \mathbf{I})^{-1} \mathbf{X}'\mathbf{y}$, where $\lambda_2 \geq 0$

Advantages of Ridge regression-

1. By utilizing a regularization technique, it reduces the variability of the estimated regression coefficients, leading to more stable estimators.e.
2. It effectively addresses the issue of multicollinearity by incorporating a shrinkage or biasing parameter λ_2.

Disadvantages of Ridge regression-

1. As ridge regression shrinks the regression coefficient towards zero by minimizing the residual sum of squares subject to the condition $|\boldsymbol{\beta}|^2 \leq t$ i.e. L_2-norm, However, it does not eliminate any coefficients by setting them to zero, making it impossible for ridge regression to generate a simplified or parsimonious model.

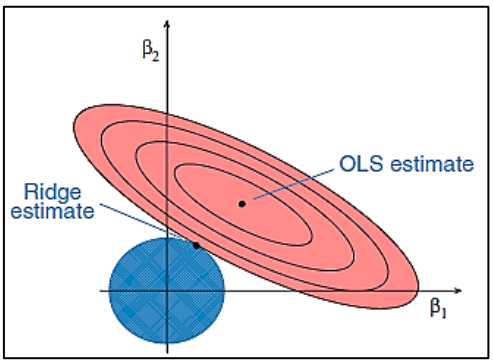


Figure 1. Two dimensional contour plot for ridge penalty

2. Ridge regression cannot perform variable selection.

4. **Least absolute shrinkage and selection operator (LASSO):**

To address the constraints of Ridge regression, Robert Tibshirani introduced a novel regularization method known as the least absolute shrinkage and selection operator in 1995. The Lasso merges the benefits of shrinkage from ridge regression with the capability for variable selection. Taking into account model 1.1, after applying a location and scale transformation, it is possible to assume that the response variable is centered and the predictor variables are standardized.

In order to achieve the variable selection benefits while also enjoying the benefits of ridge regression, lasso regression employs the L1-norm for penalization rather than the L2-norm used in ridge regression. Consequently, in lasso regression, the residual sum of squares with a penalty is expressed as follows:

Now differentiating the equation 4.1 with respect to $\boldsymbol{\beta}$ and equating to 0 gives the lasso estimates ($\boldsymbol{\beta}_L$) which is as follows:

$$\hat{\boldsymbol{\beta}}_L = (\mathbf{X}'\mathbf{X})^{-1}\left(\mathbf{X}'\mathbf{y} - \frac{\lambda_1}{\mathbf{w}}\right) \qquad 4.2$$

where $\mathbf{w}$ is the vector having all element as 1 or -1 denoting the sign of $\boldsymbol{\beta}$ and $\lambda_1 \geq 0$.

Advantages of Lasso regression-

1. By utilizing regularization techniques, it reduces the variance of the estimated regression coefficients, leading to more stable estimators.
2. Additionally, it can conduct variable selection by shrinking the estimated values of less significant regression coefficients to exactly zero.

For clarity in illustration, let's examine a model with just two regressor variables in Figure 2.

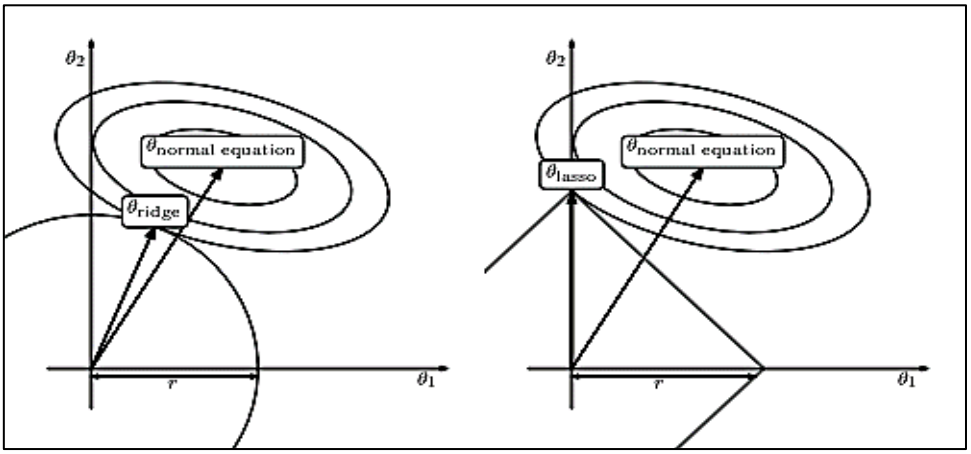


Fig.2. Contour plots in two dimensions showcasing the ridge penalty (left) and the lasso penalty (right).

From figure 2, it is quite clear that the contour of normal equation of OLS i.e. $(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y}$ It is impossible to intersect the constraint area, which is circular in the context of ridge regression, through the axis because, in ridge regression, the penalty term is based on the L2 norm.No estimated regression coefficients are assigned a value of zero. However, with lasso regression, the constraint region takes on a diamond shape because it uses the L1-norm as a penalty, which increases the likelihood that the normal equation's contour intersects the lasso constraint region at one of the axes. As a result, it can cause some of the estimated coefficients to be exactly equal to zero.

Disadvantages of Lasso regression-

1. In high-dimensional scenarios (p>n), lasso can select a maximum of n variables before it reaches its limit, which restricts its ability to choose variables.

2. When dealing with highly correlated variables (grouped), lasso typically picks just one variable from each group while disregarding the rest.

3. In situations where predictors are highly correlated (even with n>p), ridge regression often provides better prediction performance than lasso.

Why there is a need of another regularization technique?

When handling high-dimensional data with many regressor variables, pairwise correlations are common, often rendering lasso regression ineffective. For instance, in microarray data with thousands of predictor genes and limited samples, genes within the same biological pathway are typically highly correlated, forming groups. An effective gene selection method should discard irrelevant genes while automatically including entire groups once one gene is selected, enabling "grouped selection."

In situations where the number of predictors (p) exceeds the number of samples (n), lasso regression is inadequate, as it can only select up to n variables out of p (Efron et al., 2004) and fails to capture grouping information. Furthermore, its predictive performance is often lacking. This challenge inspired Hui Zou and Trevor Hastie to develop a new regularization technique called elastic net.

Illustrations

4.1. Numerical example 1

There is many forms of cancer which human contract throughout their lifetime and the prostate cancer is a prevalent form in males. The dataset we are using in this example is obtained from http://www.arxiv.org/abs/q-bio.QM/0401043 having data related to study of this prostate cancer. The predictors consist of eight clinical measurements: log of cancer volume (lcavol), log of prostate weight (lweight), age, logarithm of the amount of benign prostatic hyperplasia (lbph), presence of seminal vesicle invasion (svi), log of capsular penetration (lcp), Gleason score (gleason), and the percentage of Gleason score 4 or 5 (pgg45). The response variable is the logarithm of prostate-specific antigen (lpsa).

The purpose of this illustration is to evaluate the effectiveness of different regression models, including OLS, ridge regression, and lasso, regarding their prediction accuracy and ability to select variables.

The dataset has been split into two segments: a training set comprising 67 observations and a test set containing 30 observations. Parameters for model fitting and tuning are determined using tenfold cross-validation on the training set. Then the mean squared error is computed on the test data for all the methods i.e. OLS, ridge regression, lasso.

Table1. Comparison between OLS, ridge, and Lasso

METHOD	PARAMETERS	Test mean-squared error	Variable Selected
OLS		0.586	All
Ridge regression	λ =1	0.566	All
Lasso	s=0.39	0.499	(1,2,4,5,8)

5. References

Breiman, L. (1996). Heuristics of instability and stabilization in model selection. *Annals of Statistics.* **24**, 2350–2383.

Craig, A.P. *et al.* (2014). Application of elastic net and infrared spectroscopy in the discrimination between defective and non-defective roasted coffees. *The New England journal of medicine.* **128**, 393-400.

Dettling, M. and B¨uhlmann, P. (2004). Finding predictive gene groups from microarray data. *Journal of Multivariate Analysis.* **90**,106–131.

Diaz-Uriarte, R. (2003). A simple method for finding molecular signatures from gene expression data. *Technical Report*. Spanish National Cancer Center. (Available from http://www.arxiv.org/abs/q-bio.QM/0401043.)

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004).Least angle regression. *Annals of Statistics*. **32** (2), 409–499.

Greenshtein. E. (2006). Best subset selection, persistence in high-dimensional statistical learning and optimization under l1 constraint. *Annals of Statistics*. **34**(5), 2367–2386.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: biased estimation for non-orthogonal problems. *Techonomics.* **12**, 55–67.

Hoerl, A. E. and Kennard, R. W. (1976). Ridge regression: iterative estimation of the biasing parameter. *Communications in Statistics: Theory and Methods*. **5**, 77–88.

http://support.sas.com/documentation/cdl/en/statug/66859/HTML/default/viewer.htm#statug_sashelp_sect012.htm

Montgomery, D. C., Peck, E. A. and Vining, G. G. (2001). *Introduction to linear regression analysis*, 3rd edition, Wiley, New York.

Palejev, D. *et al.* (2011). An application of the elastic net for an endophenotype analysis. *The New England journal of medicine.* **41**(1), 120-124.

Segal, M., Dahlquist, K. and Conklin, B. (2003). Regression approach for microarray data analysis. *Journal of Computational Biology.* **10**, 961–980.

Tibshirani, R. (1996) Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society*, Series B. **58**, 267–288.

Van de Geer, S.A. (2008). High-dimensional generalized linear models and the lasso. *Annals of Statistics*. **36**(2), 614–645.

Yuan, M., Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*, Series B. **68,** 49–67.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, Series B. **67**, 301–320.

Data Classification and Data Reduction Techniques

Susheel Kumar Sarkar
ICAR-IASRI, Library Avenue, New Delhi - 110 012
Email: susheel.sarkar@icar.org.in

1. Principal Component Analysis

The aim of principal component analysis is to obtain a limited number of linear combinations (principal components) from a set of variables that preserve as much information as possible from the original variables. Frequently, a small set of principal components can substitute for the original variables in applications such as plotting, regression, and clustering. Principal component analysis can also be interpreted as a method to eliminate multicollinearity in the dataset. In this approach, the original variables are transformed into a new set of uncorrelated random variables. These new variables are linear combinations of the original ones, generated in descending order of significance, so that the first principal component captures as much variation as possible from the initial data. Let $x_1, x_2, x_3, ..., x_p$ are variables under study, then first principal component may be defined as

$$z_1 = a_{11}x_1 + a_{12}x_2 + ... + a_{1p}x_p$$

such that variance of z_1 is as large as possible subject to the condition that

$$a_{11}^2 + a_{12}^2 + ... + a_{1p}^2 = 1$$

This restriction is implemented due to the possibility that if it is overlooked,then $Var(z_1)$ can be increased simply by multiplying any $a_{1j}'s$ by a constant factor. The second principal component is defined as

$$z_2 = a_{21}x_1 + a_{22}x_2 + ... + a_{2p}x_p$$

such that $Var(z_2)$ is as large as possible next to $Var(z_1)$ subject to the constraint that

$$a_{21}^2 + a_{22}^2 + ... + a_{2p}^2 = 1 \text{ and } Cov(z_1, z_2) = 0 \text{ and so on.}$$

It is quite probable that a small number of principal components represent the majority of the variability found in the original dataset. If this is the case, these principal components can take the place of the initial p variables in further analyses, thus simplifying the problem's dimensionality. Conducting a principal components analysis frequently uncovers relationships that may not have been previously identified, allowing for insights that would not typically emerge. Nevertheless, Principal Components Analysis is primarily a tool for achieving broader goals rather than an end in itself, as it often functions as a step in larger research efforts by lowering the problem's dimensionality and facilitating interpretation. It serves as a mathematical approach that does not necessitate the user to define any statistical model or assumptions regarding the distribution of the original variables. Additionally, it's important to note that principal components are synthetic variables and, in many cases, it is challenging to ascribe physical meaning to them. Furthermore, since Principal Components Analysis converts the original set of correlated variables into a new set of uncorrelated variables, it is crucial to

highlight that if the original variables are already uncorrelated, performing Principal Components Analysis is redundant. It is essential to remember that the principal components are influenced by the measurement scale used. A typical method to address this issue is to apply standardized variables with unit variances.

Example 1.1: Let us consider the following data on average minimum temperature (x_1), average relative humidity at 8 hrs. (x_2), average relative humidity at 14 hrs. (x_3) and total rainfall in cm. (x_4) pertaining to Raipur district from 1970 to 1986 for kharif season from 21st May to 7th Oct.

	x_1	x_2	x_3	x_4
	25.0	86	66	186.49
	24.9	84	66	124.34
	25.4	77	55	98.79
	24.4	82	62	118.88
	22.9	79	53	71.88
	7.7	86	60	111.96
	25.1	82	58	99.74
	24.9	83	63	115.20
	24.9	82	63	100.16
	24.9	78	56	62.38
	24.3	85	67	154.40
	24.6	79	61	112.71
	24.3	81	58	79.63
	24.6	81	61	125.59
	24.1	85	64	99.87
	24.5	84	63	143.56
	24.0	81	61	114.97
Mean	23.56	82.06	61.00	112.97
S.D.	4.13	2.75	3.97	30.06

with the variance co-variance matrix.

$$\sum = \begin{bmatrix} 17.02 & -4.12 & 1.54 & 5.14 \\ & 7.56 & 8.50 & 54.82 \\ & & 15.75 & 92.95 \\ & & & 903.87 \end{bmatrix}$$

Determine the eigenvalues and eigenvectors of the matrix mentioned above. Order the eigenvalues from highest to lowest. The eigenvalues in descending order along with their corresponding eigenvectors are:

$\lambda_2 = 18.375 \quad \mathbf{a_2} = (0.955, -0.296, 0.011, 0.012)$

$\lambda_3 = 7.87 \quad \mathbf{a_3} = (0.141, 0.485, 0.855, -0.119)$

$\lambda_4 = 1.056 \quad \mathbf{a_4} = (0.260, 0.820, -0.509, 0.001)$

The principal components for this data will be

$$z_1 = 0.006x_1 + 0.061x_2 + 0.103x_3 + 0.993x_4$$
$$z_2 = 0.955x_1 - 0.296x_2 + 0.011x_3 + 0.012x_4$$
$$z_3 = 0.141x_1 + 0.485x_2 + 0.855x_3 - 0.119x_4$$
$$z_4 = 0.26x_1 + 0.82x_2 - 0.509x_3 + 0.001x_4$$

The variance of principal components will be given by eigenvalues i.e.
$Var(z_1) = 916.902,\ Var(z_2) = 18.375,\ Var(z_3) = 7.87,\ Var(z_4) = 1.056$

The total variation explained by principal components given as follows
$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 916.902 + 18.375 + 7.87 + 1.056 = 944.20$

Consequently, it is evident that the overall variation accounted for by the principal components is identical to that attributed to the original variables. Additionally, both mathematical and empirical evidence can demonstrate that the principal components are uncorrelated.

The proportion of total variation for by the principal components is

$$\frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4} = \frac{916.902}{944.203} = 0.97$$

The first two components contribute for a proportion

$$\frac{\lambda_1 + \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4} = \frac{935.277}{944.203} = 0.99$$

of the total variance.

Hence, in subsequent analysis, the first two principal components z_1 and z_2 could replace four variables without losing the information about the total variation in the system. The scores for the principal components can be derived by substituting the values of x_i's in the equations of z_i's. For above data, the first two principal components for first observation i.e. for year 1970 can be worked out as

$z_1 = 0.006 \times 25.0 + 0.061 \times 86 + 0.103 \times 66 + 0.993 \times 186.49 = 197.380$
$z_2 = 0.955 \times 25.0 - 0.296 \times 86 + 0.011 \times 66 + 0.012 \times 186.49 = 1.383$

Similarly for the year 1971
$z_1 = 0.006 \times 24.9 + 0.061 \times 84 + 0.103 \times 66 + 0.993 \times 124.34 = 135.54$
$z_2 = 0.955 \times 24.9 - 0.296 \times 84 + 0.011 \times 66 + 0.012 \times 124.34 = 1.134$

Therefore, the entire dataset containing four variables can be transformed into a new dataset with two principal components.

Example 1.2: Consider the same data as given in Example 1. The variance-covariance matrix was given as

$$\boldsymbol{\Sigma} = \begin{bmatrix} 2.879368 & 10.01 & -1.80905 \\ 10.01 & 199.7884 & -5.64 \\ -1.80905 & -5.64 & 3.627658 \end{bmatrix}$$

Now determine the eigenvalues and eigenvectors of the matrix mentioned above. Organize the eigenvalues from largest to smallest. The eigenvalues listed in decreasing order along with their corresponding eigenvectors are.

$\lambda_2 = 4.532 \quad \mathbf{a_2} = (-0.5737,\ 0.0530,\ \ 0.8173)$

$\lambda_3 = 1.301 \quad \mathbf{a_3} = (0.8175,\ -0.0249,\ 0.5754)$

The principal components for this data are

$$z_1 = 0.0508x_1 + 0.9983x_2 - 0.0291x_3$$
$$z_2 = -0.5737x_1 + 0.0530x_2 + 0.8173x_3$$
$$z_3 = 0.8175x_1 - 0.0249x_2 + 0.5754x_3$$

The variance of principal components will be eigenvalues i.e.

$Var(z_1) = 200.462,\ \ Var(z_2) = 4.532,\ \ Var(z_3) = 1.301$

The total variation explained by principal components is

$\lambda_1 + \lambda_2 + \lambda_3 = 200.462 + 4.532 + 1.301 = 206.295$

It can therefore be observed that the overall variation accounted for by principal components is equivalent to that accounted for by the original variables. Additionally, it can be demonstrated both mathematically and empirically that the principal components are uncorrelated.

The proportion of total variation accounted for by the principal components is

$$\frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} = \frac{200.462}{206.295} = 0.9717 \text{ of the total variance.}$$

Continuing, the first two components account for a proportion

$$\frac{\lambda_1 + \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} = \frac{204.994}{206.295} = 0.9937 \text{ of the total variance.}$$

Hence, in further analysis, the first two principal components z_1 and z_2 could replace four variables without losing information about the total variation in the system. The scores of principal components can be derived by substituting the values of x_i's in the equations of z_i's. For above data, the first two principal components of first observation i.e. of first individual is

$z_1 = 0.0508 \times 3.7 + 0.9983 \times 48.5 - 0.0291 \times 9.3$

$z_2 = -0.5737 \times 3.7 + 0.0530 \times 48.5 + 0.8173 \times 9.3$

Scores for principal components can likewise be calculated for other individuals. As a result, the complete dataset with three variables can be converted into a new dataset that contains two principal components.

R code:

```
set.seed(555)
ind <- sample(2,nrow(iris),replace=T,prob = c(0.8,0.2))
training <- iris[ind==1,]
testing <- iris[ind==2,]
library(psych)
pairs.panels(iris[1:4],gap=0,bg=c("red", "green", "blue")[iris$Species], pch=21)
pc <- prcomp(training[,-5], center = T, scale. = T)
attributes(pc)
pc$center
pc$scale
print(pc)
summary(pc)
pairs.panels(pc$x,gap=0,bg=c("red", "green", "blue")[iris$Species], pch=21)
```

2. Discriminant Analysis

The term discriminant analysis encompasses various analytical methods, including classificatory discriminant analysis (which classifies observations into two or more predefined groups based on one or more quantitative variables), Canonical discriminant analysis (a dimension reduction method related to principal components and canonical correlation), and Stepwise discriminant analysis (a variable selection approach aimed at identifying a subset of quantitative variables that most effectively distinguishes between the classes).

In the case of classificatory discriminant analysis, Fisher's Discriminant function is typically employed. It will be explained in the following sections.

Fisher's idea is to convert the multivariate observations **x** to univariate observations *y* so that *y*'s derived from the populations π_1 and π_2 are separated as much as possible. Fisher's approach presumes that populations are normal and also population covariance matrices are equal because a pooled estimate of common covariance matrix is used.

A specific linear combination of the x-values yields the following results: $y_{11}, y_{12}, ..., y_{1n_1}$ for the observations from the first population and the values $y_{21}, y_{22}, ..., y_{2n_2}$ for the observations from the second population. The separation of these two sets of univariate *y*'s is assessed in terms of the differences between $\bar{y}_1$ and $\bar{y}_2$ expressed in standard deviation units. That is,

$$\text{separartion} = \frac{|\bar{y}_1 - \bar{y}_2|}{s_y}, \text{ where } s_y^2 = \frac{\sum_{j=1}^{n_1}(y_{1j} - \bar{y}_1)^2 + \sum_{j=1}^{n_2}(y_{2j} - \bar{y}_2)^2}{n_1 + n_2 - 2}$$

is the pooled estimate of the variance. The objective is to select the linear combination of the **x** to achieve maximum separation of the sample means $\bar{y}_1$ and $\bar{y}_2$.

Result: The linear combination $y = \hat{\mathbf{l}}'\mathbf{x} = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}_{pooled}^{-1}\mathbf{x}$ maximizes the ratio

$$\frac{(\text{Squared distance between sample mean of y})}{(\text{Sample variance of y})} = \frac{(\bar{y}_1 - \bar{y}_2)^2}{s_y^2}$$

$$= \frac{(\hat{\mathbf{l}}\bar{\mathbf{x}}_1 - \hat{\mathbf{l}}'\bar{\mathbf{x}}_2)^2}{\hat{\mathbf{l}}'S_{pooled}\hat{\mathbf{l}}}$$

$$= \frac{(\hat{\mathbf{l}}'\mathbf{d})^2}{\hat{\mathbf{l}}'S_{pooled}\hat{\mathbf{l}}'}$$

over all possible coefficient vectors $\hat{\mathbf{l}}'$ where $\mathbf{d} = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$. The maximum of the above ratio is $\mathbf{D}^2 = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}_{pooled}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$, the Mahalanobis distance.

Fisher's solution to the separation problem can also be used to classify new observations. An allocation rule is as follows.

Allocate $\mathbf{x}_0$ to π_1 if $y_0 = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}_{pooled}^{-1}\mathbf{x}_0 \geq \hat{\mathbf{m}} = \frac{1}{2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}_{pooled}^{-1}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2)$

and to π_2 if $y_0 < \hat{\mathbf{m}}$

If we assume the populations π_1 and π_2 are multivariate normal with a common covariance matrix, the a test of $\mathbf{H}_0 : \mu_1 = \mu_2$ versus $\mathbf{H}_1 : \mu_1 \neq \mu_2$ is accomplished by referring

$$\frac{(n_1 + n_2 - p - 1)}{(n_1 + n_2 - 2)p}\left(\frac{n_1 n_2}{n_1 + n_2}\right)\mathbf{D}^2$$

to an F-distribution with $\nu_1 = p$ and $\nu_2 = n_1 + n_2 - p - 1$ degrees of freedom. If $\mathbf{H}_0$ is rejected, we can conclude the separation between the two populations is significant.

Example 2.1: {Example 11.3 in Johnson and Wichern, 2002}. To construct a procedure for detecting potential hemophilia 'A' carriers, blood samples were analyzed for two groups of women and measurements on two variables, $x_1 = \log_{10}(AHF\ activity)$ and

$x_2 = \log_{10}(AHF-like\ antigens)$ recorded. The first group of $n_1 = 30$ women were selected from a population who do not carry hemophilia gene (normal group). The second group of $n_2 = 22$ women were selected from known hemophilia 'A' carriers (obligatory group). The mean vectors and sample covariance matrix are given as

$$\bar{\mathbf{x}}_1 = \begin{bmatrix} -0.0065 \\ -0.0390 \end{bmatrix}, \quad \bar{\mathbf{x}}_2 = \begin{bmatrix} -0.2483 \\ 0.0262 \end{bmatrix} \text{ and } \mathbf{S}_{pooled}^{-1} = \begin{bmatrix} 131.158 & -90.423 \\ -90.423 & 108.147 \end{bmatrix}$$

Now the linear discriminant function is

$$y_0 = \hat{\mathbf{l}}'\mathbf{x}_0 = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}_{pooled}^{-1}\mathbf{x}_0$$

$$= \begin{bmatrix} 0.2418 & -0.0652 \end{bmatrix} \begin{bmatrix} 131.158 & -90.423 \\ -90.423 & 108.147 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= 37.61\, x_1 - 28.92\, x_2$$

Moreover

$$\bar{y}_1 = \hat{\mathbf{l}}'\mathbf{x}_1 = \begin{bmatrix} 37.61 & -28.92 \end{bmatrix} \begin{bmatrix} -0.0065 \\ -0.0390 \end{bmatrix} = 0.88$$

$$\bar{y}_2 = \hat{\mathbf{l}}'\mathbf{x}_2 = \begin{bmatrix} 37.61 & -28.92 \end{bmatrix} \begin{bmatrix} -0.2483 \\ -0.0262 \end{bmatrix} = -10.10$$

And the mid-point between these means is

$$\hat{\mathbf{m}} = \frac{1}{2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}_{pooled}^{-1}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2) = \frac{1}{2}(\bar{y}_1 + \bar{y}_2) = -4.61$$

Now to classify a women who may be a hemophilia 'A' carrier with $x_1 = -0.210$ and $x_2 = -0.044$.

We calculate: $y_0 = \hat{\mathbf{l}}'\mathbf{x}_0 = 37.61\, x_1 - 28.92\, x_2 = -6.62$. Since $y_0 < \hat{\mathbf{m}}$ we classify the women in π_2 population, *i.e.*, to obligatory carrier group.

Example 2.2: Consider a problem of discrimination between two species of flea beetles, namely Chaetocnema concinna (Π_1) and Chaetocnema heikertingeri(Π_2) based on various physical measurements. Data on the maximal width of the aedeagus in the forepart (X1) in microns and the front angle of the aedeagus(X2) in units of 7.5 degrees were subsequently analysed. Construct the linear discriminant function for each species and use it to assign a new x_o'=[153 ,16] to one of the populations Π_t ,i=1,2.And identify the misclassified species.

R code

```
data("iris")
iris
library(psych)
pairs.panels(iris[1:4],gap=0,bg=c("red", "green", "blue")[iris$Species], pch=21)
set.seed(555)
ind <- sample(2,nrow(iris),replace=T,prob = c(0.6,0.4))
training <- iris[ind==1,]
testing <- iris[ind==2,]
library(MASS)
linear <- lda(Species~.,training)
linear
attributes(linear)
 linear$prior
p <- predict(linear, training)
p
ldahist(data=p$x[,1], g=training$Species)
ldahist(data=p$x[,2], g=training$Species)
# Enable the r-universe repo
#options(repos = c(
#  fawda123 = 'https://fawda123.r-universe.dev',
 # CRAN = 'https://cloud.r-project.org'))
# Install ggord
#install.packages('ggord')
library(ggord)
ggord(linear,training$Species, ylim=c(-10,10))
library(klaR)
partimat(Species~., data=training, method="lda")
p1 <- predict(linear, training)$class
tab <- table(Predicted=p1, Actual=training$Species)
tab
sum(diag(tab)/sum(tab))
p2 <- predict(linear, testing)$class
p2
tab2 <- table(Predicted=p2, Actual=testing$Species)
tab2
sum(diag(tab2)/sum(tab2))
```

3. Factor Analysis

The primary aim of factor analysis is, if possible, to explain the covariance relationships between numerous variables using a limited number of underlying, yet unobservable, random quantities known as factors. A common point of misunderstanding in factor analysis is the term

"factor." It sometimes denotes a hypothetical, unobservable variable, as seen in the term common factor. In this context, factor analysis should be differentiated from component analysis, as a component represents an observable linear combination. Factor is also used in the context of matrix factorization, meaning one matrix is considered a factor of another if the product of the first matrix and its transpose results in the second matrix. Thus, factor analysis encompasses all methods of data analysis employing matrix factors, which include component analysis and common factor analysis. A common factor is an unobservable hypothetical variable that influences the variance of at least two observed variables. The term "factor" without qualification typically refers to a common factor. A unique factor, on the other hand, is an unobservable hypothetical variable that affects the variance of only one observed variable.

To understand the role of Factor Analysis, consider the following examples

Example 3.1: What fundamental attitudes influence people's responses to the questions on a political survey? Analyzing the relationships between the survey items shows that there is considerable overlap among different categories of questions—items related to taxes tend to be correlated with one another, while those concerning military topics also show correlation, and so forth. By using factor analysis, you can explore the number of underlying factors, often allowing you to discern the conceptual meaning of those factors. Furthermore, you can calculate factor scores for each participant, which can later be used in further analyses. For instance, you may create a logistic regression model to forecast voting behaviour based on these factor scores.

Example 3.2: A producer of fabricated parts seeks to determine the key factors that contribute to a successful salesperson. The producer possesses data as outlined in the following table. They are curious to know if these seven variables can be condensed into two or three factors for a more insightful understanding of the issue.

Data Matrix for Factor Analysis of seven variables (14 sales people)

Sales Person	**Height** (x_1)	**Weight** (x_2)	**Education** (x_3)	**Age** (x_4)	**No. of Children** (x_5)	**Size of Household** (x_6)	**IQ** (x_7)
1	67	155	12	27	0	2	102
2	69	175	11	35	3	6	92
3	71	170	14	32	1	3	111
4	70	160	16	25	0	1	115
5	72	180	12	30	2	4	108
6	69	170	11	41	3	5	90
7	74	195	13	36	1	2	114
8	68	160	16	32	1	3	118
9	70	175	12	45	4	6	121
10	71	180	13	24	0	2	92
11	66	145	10	39	2	4	100

12	75	210	16	26	0	1	109
13	70	160	12	31	0	3	102
14	71	175	13	43	3	5	112

Is it possible to condense the seven variables into three main factors? One might intuitively propose that three key factors exist: maturity indicated by age, number of children, and household size; physical stature represented by height and weight; and intellect or training denoted by education and IQ.

The SAS program has been utilized to analyze the data from the sales team. This program takes input data in its original form and automatically converts it into standardized scores. Below are the three factors obtained from the sales team data through principal component analysis using the SAS program:

Three-factor results with seven variables

Sales People Characteristics

Variable	**Factor I**	**Factor II**	**Factor III**	**Communality**
Height	0.59038	0.72170	-0.30331	0.96140 (sumsq I,II and III)
Weight	0.45256	0.75932	-0.44273	0.97738
Education	0.80252	0.18513	0.42631	0.86006
Age	-0.86689	0.41116	0.18733	0.95564
No. of Children	-0.84930	0.49247	0.05883	0.96730
Size of Household	-0.92582	0.30007	-0.01953	0.94756
IQ	0.28761	0.46696	0.80524	0.94918
Sum of squares	3.61007	1.85136	1.15709	
Variance summarized	0.51572	0.26448	0.16530	Average=0.94550

Factor Loadings

The coefficients in the factor equations are called "factor loadings". They appear above in each factor column, corresponding to each variable. The equations are:

$$\mathbf{F}_1 = 0.59038x_1 + 0.45256x_2 + 0.80252x_3 - 0.86689x_4 - 0.84930x_5 - 0.92582x_6 + 0.28761x_7$$

$$\mathbf{F}_2 = 0.72170x_1 + 0.75932x_2 + 0.18513x_3 + 0.41116x_4 + 0.49247x_5 + 0.30007x_6 + 0.46696x_7$$

$$\mathbf{F}_3 = -0.30331x_1 - 0.44273x_2 + 0.80252x_3 + 0.18733x_4 + 0.58830x_5 - 0.01953x_6 + 0.80524x_7$$

The factor loadings depict the relative importance of each variable with respect to a particular factor. In all the three equations, education (x_3) and IQ (x_7) have got positive loading factor indicating that they are variables of importance in determining the success of salesperson.

Variance summarized

Factor analysis uses the principle of minimizing variance in the original set of variables. Each factor plays a role in this minimization. In our case, Factor I represents 51.6% of the overall variance. Factor II accounts for 26.4%, while Factor III contributes 16.5%. Altogether, these three factors clarify nearly 95% of the variance.

Communality

In a perfect scenario, the derived factors would account for 100% of the variance in all original variables. "Communality" reflects the proportion of variance in the original variables that is explained by the factor combinations in the solution. Therefore, communality is calculated for each original variable. The communality of each variable can be viewed as indicating how well it is depicted by the factor system. In our case, the communality exceeds 85% for each variable. Hence, the three factors appear to encapsulate the underlying dimensions represented by these variables.

There is also a method known as varimax rotation that can be utilized after obtaining the initial results. This can be applied if the analyst finds it necessary. We do not plan to focus on this further, and those interested in exploring this aspect can use the SAS program for varimax rotation.

R code:

```
library(psych)
data
#fm is a factoring method
#data accepted ia a raw data matrix filled with numerical values
#data can be correlation or covariance matrix, result should be same

pa <- fa(r=data, nfactor=3
        rotate="varimax"
        fm="pa"
        residuals=TRUE)
ml <- fa(r=data, nfactor=3
        rotate="varimax"
        fm="ml"
        residuals=TRUE)
```

4. Cluster Analysis

The primary goal of cluster analysis is to identify “natural” or “actual” groupings, if they exist, among a collection of individuals (or objects, points, units, etc.). This collection may represent an entire population or a sample drawn from a larger population. In more formal terms, cluster analysis seeks to assign a collection of individuals into mutually exclusive and exhaustive

groups so that those within the same group exhibit similarities, while those in different groups show differences. These groups are referred to as partitions or dissections. Additionally, cluster analysis can be employed for data summarization instead of solely identifying natural or actual groupings. Clustering or grouping is different from classification methods because classification involves a predetermined number of groups, aiming to place new observations into one of those. Cluster analysis is considered a more fundamental technique since it does not operate on assumptions about the number or structure of groups. Grouping is carried out based on similarity or distance (dissimilarity). Various distance criteria are utilized for this purpose.

Euclidean distance: This is likely the most frequently selected type of distance. It represents the geometric distance in a multidimensional space and is calculated as:

$$d(\mathbf{x},\mathbf{y}) = \left[\sum_{i=1}^{p}(x_i - y_i)^2\right]^{1/2} = \sqrt{(\mathbf{x}-\mathbf{y})'(\mathbf{x}-\mathbf{y})}$$

Where, $\mathbf{x},\mathbf{y}$ are the p-dimensional vectors of observations. It's important to note that Euclidean (and squared Euclidean) distances are typically calculated using raw data rather than standardized data. This approach has certain benefits, such as ensuring that the distance between any two objects remains unaffected by the introduction of new objects into the analysis, particularly if those new objects are outliers. However, the computed distances can be significantly influenced by variations in scale across the dimensions used for the calculations. For instance, if one dimension represents a length measured in centimeters and is then converted to millimeters by multiplying the values by 10, the resulting Euclidean or squared Euclidean distances derived from multiple dimensions can be heavily impacted, potentially leading to biased outcomes due to the dimensions with a larger scale. As a general rule, it's advisable to standardize the dimensions so that they possess comparable scales.

Squared Euclidean distance: This approach is implemented to assign increasingly higher significance to items that are located farther away. This distance is the square of the Euclidean distance.

Statistical distance: The statistical distance between the two p-dimensional vectors $\mathbf{x}$ and $\mathbf{y}$ is $d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})'\mathbf{s}^{-1}(\mathbf{x}-\mathbf{y})}$, where $\mathbf{s}$ is the sample variance-covariance matrix.

A greater variety of distance measures can be found in the literature. For more information, one can refer to Romesburg (1984).

R code:

```
data("ïris")
iris
plot(iris)
```

```
irissacled <- scale(iris[,-5])
fitk <- kmeans(irissacled,3)
fitk
str(fitk)
plot(iris, col=fitk$cluster)
#hierarchial clustering....
d <- dist(irissacled)
fitH <- hclust(d,"ward.D2")
plot(fitH)
rect.hclust(fitH, k=3, border="red")
```

References

Chatfield, C and Collins, A.J. (1980). *Introduction to Multivariante Analysis*. Chapman and Hall, New York.

Gabriel, K. R. (1971, The biplot graphic display of matrices with application to principal component analysis. *Biometrika* **58**:453-467.

Johnson, R.A. and Wichern, D.W. (1988). *Applied Multivariate Statistical Analysis.* Prentice hall, New Jersey.

Kshirsagar, A.M. (1972). *Multivariate Analysis*. Marcell Dekker, Inc. New York.

Rao, C.R. (1989). *Linear Statistical Inference and its applications*. Wiley Eastern Ltd., New Delhi.

Rawlings, J.O. (1988). *Applied Regression Analysis*: A Research Tool. Wadsworth and Brooks/Cole Advanced Books and Software.

Romesburg, H.C. (1984). *Cluster Analysis for Researchers*. Lifetime Learning Publications, California.

Principal Components Analysis

Arpan Bhowmik[1], Bijay Chanda[2] and Dibyendu Deb[1]
[1]ICAR-Indian Agricultural Research Institute, Gogamukh, Assam
[2]ICAR-Central Agroforestry research Institute, Jhanshi, Uttar Pradesh
Email: arpan.stat@gmail.com

Multivariate data consist of observations on several different variables for a number of individuals or subjects. Data of this type arise in all the branches of science, ranging from psychology to biology, and methods of analyzing multivariate data constitute an increasingly important area of statistics. Indeed, the vast majority of data in forestry is multivariate and proper handling of such data is highly essential. Principal component analysis (PCA) and factor analysis (FA) are multivariate methods used on a single collection of variables to identify which groups of variables within that collection create consistent subsets that are largely independent from each other. The specifics of PCA and FA are elaborated upon below.

Principal Components Analysis

Often, the variables being analyzed are strongly correlated, meaning they essentially convey the same information. To explore the relationships among a collection of p correlated variables, it can be beneficial to convert the initial set of variables into a new set of uncorrelated variables known as principal components. These new variables are formed as linear combinations of the original variables and are established in decreasing order of significance, so that, for instance, the first principal component captures as much of the variation in the original dataset as possible.

Let $x_1, x_2, x_3, \ldots, x_p$ are variables under study, then first principal component may be defined as

$$z_1 = a_{11} x_1 + a_{12} x_2 + \ldots\ldots + a_{1p} x_p$$

such that variance of z_1 is as large as possible subject to the condition that

$$a_{11}^2 + a_{12}^2 + \ldots\ldots + a_{1p}^2 = 1$$

This limitation is established because if it isn't enforced, $Var(z_1)$ can be increased merely by multiplying any a_{1j} s by a constant factor.

The second principal component is defined as

$$z_2 = a_{21} x_1 + a_{22} x_2 + \ldots\ldots + a_{2p} x_p$$

such that $Var(z_2)$ is as large as possible next to $Var(z_1)$ subject to the constraint that

$a_{21}^2 + a_{22}^2 + ... + a_{2p}^2 = 1$ and $cov(z_1, z_2) = 0$ and so on.

It is highly probable that the initial few principal components capture most of the variability present in the original dataset. If that is the case, these few principal components can substitute for the original p variables in further analyses, thereby decreasing the effective dimensionality of the issue. A principal component analysis often uncovers relationships that were not previously recognized, allowing for interpretations that typically may not arise. However, Principal Component Analysis should be viewed more as a tool to achieve larger objectives rather than an end goal on its own, as it often serves as a step within a more extensive investigation by simplifying the problem's dimensionality and facilitating interpretation. This method is mathematical in nature and does not require the user to define a statistical model or make assumptions about the distribution of the original variables. It is also important to note that principal components are constructed variables, and often, it is challenging to attribute physical meaning to them. Furthermore, since Principal Component Analysis converts the initial set of correlated variables into a new set of uncorrelated variables, it is important to emphasize that if the original variables are uncorrelated, conducting a principal component analysis would be unnecessary.

Computation of principal components:

Let us take a look at the following information regarding the average minimum temperature (x_1), average relative humidity at 8 a.m. (x_2), average relative humidity at 2 p.m. (x_3), and total rainfall in centimeters (x_4) for the Raipur district from 1970 to 1986, specifically during the kharif season, which spans from May 21 to October 7.

X_1	x_2	x_3	x_4
25.0	86	66	186.49
24.9	84	66	124.34
25.4	77	55	98.79
24.4	82	62	118.88
22.9	79	53	71.88

	7.7	86	60	111.96
	25.1	82	58	99.74
	24.9	83	63	115.20
	24.9	82	63	100.16
	24.9	78	56	62.38
	24.3	85	67	154.40
	24.6	79	61	112.71
	24.3	81	58	79.63
	24.6	81	61	125.59
	24.1	85	64	99.87
	24.5	84	63	143.56
	24.0	81	61	114.97
Mean	23.56	82.06	61.00	112.97
S.D.	4.13	2.75	3.97	30.06

with the variance co-variance matrix.

$$\Sigma = \begin{bmatrix} 17.02 & -4.12 & 1.54 & 5.14 \\ & 7.56 & 8.50 & 54.82 \\ & & 15.75 & 92.95 \\ & & & 903.87 \end{bmatrix}$$

Determine the eigenvalues and eigenvectors of the given matrix. Organize the eigenvalues in descending order. The eigenvalues, listed in descending order along with their corresponding eigenvectors, are

$\lambda_1 = 916.902 \quad a_1 = (0.006, \quad 0.061, \quad 0.103, \quad 0.993)$

$\lambda_2 = 18.375 \quad a_2 = (0.955, \quad -0.296, \quad 0.011, \quad 0.012)$

λ_3 = 7.87 a_3 = (0.141, 0.485, 0.855, -0.119)

λ_4 = 1.056 a_4 = (0.260, 0.820, -0.509, 0.001)

The principal components of the data will be

$z_1 = 0.006\ x_1 + 0.061\ x_2 + 0.103\ x_3 + 0.993\ x_4$

$z_2 = 0.955\ x_1 - 0.296\ x_2 + 0.011\ x_3 + 0.012\ x_4$

$z_3 = 0.141\ x_1 + 0.485\ x_2 + 0.855\ x_3 - 0.119\ x_4$

$z_4 = 0.26\ x_1 + 0.82\ x_2 - 0.509\ x_3 + 0.001\ x_4$

The variance of principal components given by eigen values i.e.

$Var(z_1) = 916.902,\ Var(z_2) = 18.375,\ Var(z_3) = 7.87,\ Var(z_4) = 1.056$

The total variation is explained by original variables

$= Var(x_1) + Var(x_2) + Var(x_3) + Var(x_4)$

$= 17.02 + 7.56 + 15.75 + 903.87 = 944.20$

The total variation was explained by principal components

$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 916.902 + 18.375 + 7.87 + 1.056 = 944.20$

The overall variation explained by the principal components is the same as that explained by the original variables. Both mathematical and empirical evidence show that the principal components are uncorrelated. The first principal component accounts for a specific proportion of the total variation, which is...

$$\frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4} = \frac{916.902}{944.203} = .97$$

Continuing, the first two components account for a proportion

$$\frac{\lambda_1 + \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4} = \frac{935.277}{944.203} = .99$$

$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4$ 944.203

of the total variance.

Hence, in further analysis, the first or first two principal components z_1 and z_2 could replace four variables without losing information about the total variation in the system. The scores of principal components can be obtained by substituting the values of x_i s in the equations of z_i s. For the above data, the first two principal components for the first observation i.e. for the year 1970, can be worked out as

$z_1 = 0.006 \times 25.0 + 0.061 \times 86 + 0.103 \times 66 + 0.993 \times 186.49 = 197.380$

$z_2 = 0.955 \times 25.0 - 0.296 \times 86 + 0.011 \times 66 + 0.012 \times 186.49 = 1.383$

Similarly for the year 1971

$z_1 = 0.006 \times 24.9 + 0.061 \times 84 + 0.103 \times 66 + 0.993 \times 124.34 = 135.54$

$z_2 = 0.955 \times 24.9 - 0.296 \times 84 + 0.011 \times 66 + 0.012 \times 124.34 = 1.134$

Thus the whole data with four variables can be converted to a new data set with two principal components.

Note:It is essential to understand that principal components are affected by the measurement scale used. For example, if X1 is recorded in degrees Fahrenheit (°F) rather than degrees Celsius (°C), and X4 is noted in millimeters (mm) instead of centimeters (cm), the principal components calculated from the data will change when converted back to the original variables. In rare situations, the results might remain the same. To tackle this challenge, a typical solution is to employ standardized variables that have unit variances, which involves using a correlation matrix instead of a dispersion matrix. However, it's important to recognize that the principal components obtained from the original variables differ from those derived from the correlation matrix and might explain varying proportions of variance in the dataset. Additionally, one set of principal components is not a straightforward function of the other. When variables are standardized, they typically contribute more equally to the principal components generated from the correlation matrix. Therefore, it is advisable to standardize variables when they are measured on scales with significantly different ranges or when the units of measurement are not compatible. Often, the population dispersion or correlation matrix is not available, and in these cases, the sample dispersion matrix or correlation matrix can be used instead.

Applications of principal components:

- The primary application of principal component analysis (PCA) is data reduction. It effectively determines the dimensionality of the dataset. When the first few components capture the majority of the variation in the original data, their scores can be used for further analysis instead of the original variables.

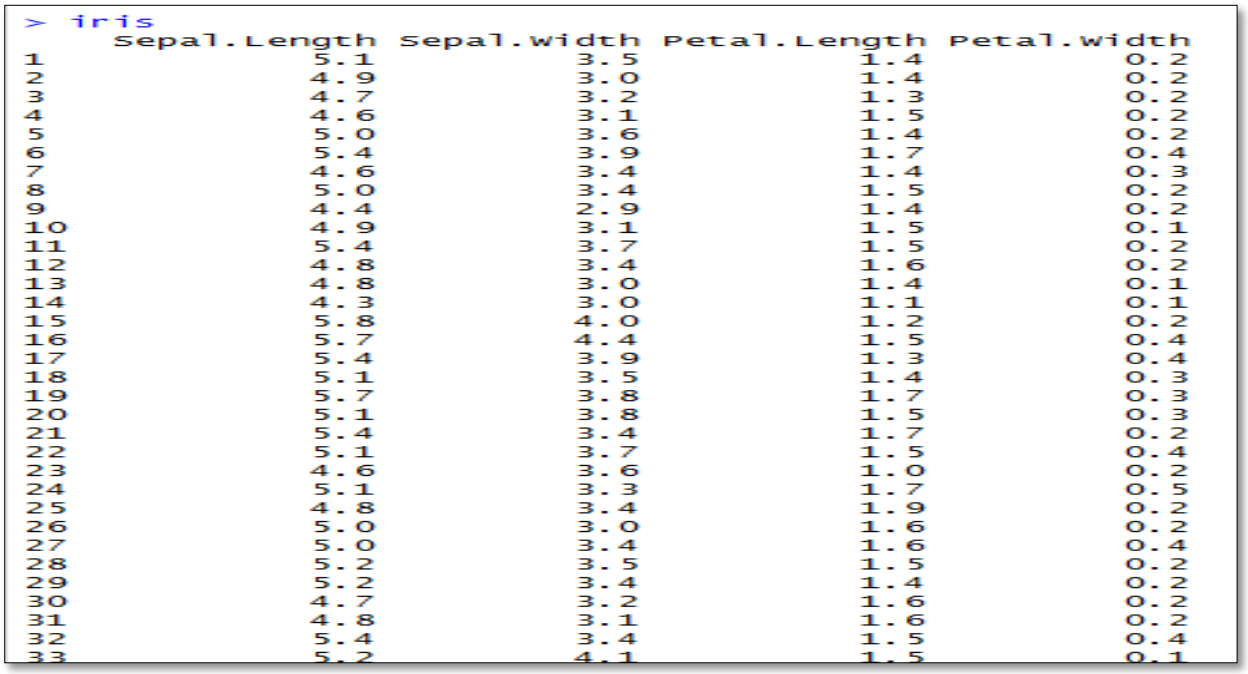
- Visualizing data becomes challenging with more than three variables. By using PCA, it is often possible to explain most of the variability in the data using just the first two components, allowing us to plot the scores of these two components for each individual. Consequently, PCA facilitates the representation of data in two dimensions, making it easier to identify outliers or clusters among individuals. The technique frequently uncovers relationships among variables that may go unnoticed by other methods.

- Reducing dimensionality can also benefit analyses when the number of variables exceeds the number of observations, such as in discriminant and regression analysis. In these scenarios, PCA assists by minimizing the dimensionality of the data.

- Multiple regression can pose risks when independent variables are highly correlated. PCA serves as an effective technique to address this issue. Regression analysis can then be performed using principal components as predictors instead of the original variables, a method referred to as principal component regression.

Analysis using R

the screenshot of iris data is as follow:



```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width
1            5.1         3.5          1.4         0.2
2            4.9         3.0          1.4         0.2
3            4.7         3.2          1.3         0.2
4            4.6         3.1          1.5         0.2
5            5.0         3.6          1.4         0.2
6            5.4         3.9          1.7         0.4
7            4.6         3.4          1.4         0.3
8            5.0         3.4          1.5         0.2
9            4.4         2.9          1.4         0.2
10           4.9         3.1          1.5         0.1
11           5.4         3.7          1.5         0.2
12           4.8         3.4          1.6         0.2
13           4.8         3.0          1.4         0.1
14           4.3         3.0          1.1         0.1
15           5.8         4.0          1.2         0.2
16           5.7         4.4          1.5         0.4
17           5.4         3.9          1.3         0.4
18           5.1         3.5          1.4         0.3
19           5.7         3.8          1.7         0.3
20           5.1         3.8          1.5         0.3
21           5.4         3.4          1.7         0.2
22           5.1         3.7          1.5         0.4
23           4.6         3.6          1.0         0.2
24           5.1         3.3          1.7         0.5
25           4.8         3.4          1.9         0.2
26           5.0         3.0          1.6         0.2
27           5.0         3.4          1.6         0.4
28           5.2         3.5          1.5         0.2
29           5.2         3.4          1.4         0.2
30           4.7         3.2          1.6         0.2
31           4.8         3.1          1.6         0.2
32           5.4         3.4          1.5         0.4
33           5.2         4.1          1.5         0.1
```

For performing PCA analysis following code is required
Iris #data

iris2<-iris[,1:4] #considering only quantitative variables

iris2

iris.pca<-prcomp(iris2, center = TRUE, scale = TRUE)

The output is follows:

```
> iris.pca$sdev
[1] 1.7083611 0.9560494 0.3830886 0.1439265
> iris.pca$rotation
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
> iris.pca$center
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
> iris.pca$scale
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.8280661    0.4358663    1.7652982    0.7622377
> iris.pca.var<-(iris.pca$sdev)^2
> iris.pca.var
[1] 2.91849782 0.91403047 0.14675688 0.02071484
```

```
> variation.PC=iris.pca.var/sum(iris.pca.var)
> variation.PC
[1] 0.729624454 0.228507618 0.036689219 0.005178709
```

Suggested Readings

- Chatfield, C. and Collins, A.J. (1990). Introduction to multivariate analysis. *Chapman and Hall publications.*
- Johnson, R.A. and Wichern, D.W. (1996). Applied multivariate statistical analysis. *Prentice-Hall of India Private Limited*.

Factor Analysis

Arpan Bhowmik[1], Bijay Chanda[2] and Dibyendu Deb[1]
[1]ICAR-Indian Agricultural Research Institute, Gogamukh, Assam
[2]ICAR-Central Agroforestry research Institute, Jhanshi, Uttar Pradesh
Email: arpan.stat@gmail.com

Factor analysis emerged from psychology to clarify concepts such as intelligence and attitudes. Its primary goal is to understand the relationships among numerous variables by relating them to a limited number of unobservable random quantities known as factors.In the factor model, which operates under the assumption of linear relationships, each observed variable is portrayed as a linear combination of a few unobservable common factors and one specific latent factor. The common factors explain the covariances between the observed variables, while specific factors contribute only to the variances of their respective responses.The rationale behind the factor model is the belief that variables can be categorized based on their correlations—those within a category show high correlation with each other but have low correlations with variables in other categories. This suggests that each group of variables may represent a single underlying factor that accounts for the observed correlations. For instance, a person's grades in various subjects may be influenced by general aptitudes (common factors) and individual differences (specific factors), with the objective being to deduce the unobservable aptitudes based on the observable grades in different subjects.

The Factor Model

Imagine that we collect data on p variables for n subjects (x_{ij} , where i=1,2,…p; j=1,2,…n). The factor analysis approach presumes the existence of m latent factors (with m<p), and every observed variable is modeled as a linear combination of these factors along with unique error terms, such that

$$x_{ij} = a_{i1} f_{1j} + a_{i2} f_{2j} + \ldots\ldots\ldots\ldots + a_{im} f_{mj} + a_{i0} y_{ij} \qquad j = 1,2,\ldots,p$$

where $a_{i1}, a_{i2}, \ldots\ldots, a_{im}$ are *factor loadings* of i-th variable to m common hypothetical factors of j-th respondent ($f_{1j}, f_{2j}, \ldots.., f_{mj}$) and a_{i0} is given to factor specific to i-th variable pertaining to j-th respondent (y_{ij}).

The j-th variable's variance that is attributed to the m common factors is referred to as the j-th communality, while the variance attributed to the specific factors is known as the uniqueness or specific variance.

Factor analysis involves:

- Deciding number of common factors (m)
- Estimating factor loadings (a_{ik})
- Calculating factor scores (f_{kj})

Methods of Estimation

Factor analysis is conducted in two stages; the first stage involves applying certain restrictions to obtain an initial solution, followed by a final solution achieved through the rotation of this initial result. In the literature, the two most prevalent methods for estimating parameters are the principal component method (along with its related principal factor method) and the maximum likelihood method. The outcomes from either approach can be rotated to facilitate the interpretation of factors, which results in factor loadings that are either close to one or close to zero. The Varimax Rotation method is the most widely used for orthogonal rotation, while oblique rotations may be employed in certain specific cases. It is generally advisable to explore multiple solution methods. If the factor model fits the issue being analyzed, the solutions should demonstrate consistency with each other. The methods for estimation and rotation necessitate iterative calculations that must be performed using a computer. Factor analysis will not be useful when variables are uncorrelated; in such cases, specific factors become predominant, while the primary objective of factor analysis is to identify a few significant common factors.

Theoretical guidance for the number of factors is provided by the rank of the population variance-covariance matrix. Nevertheless, in practice, the number of common factors retained in the model is often increased until a satisfactory proportion of the total variance is accounted for. A common convention seen in various software programs is to set the number of factors (m) equal to the number of eigenvalues exceeding one (as seen in programs like SAS and SPSS). Similar to principal component analysis, the principal factor method for factor analysis is influenced by the units of measurement. Changes in units will alter the solution; however, in this method, the estimated factor loadings for any specific factor remain unchanged as the number of factors increases. Conversely, with the maximum likelihood method, the solution remains unaffected by changes in the units of measurement, but it will change if the number of common factors is modified.

Example: In a study focused on consumer preferences, a random group of customers was surveyed to evaluate various features of a new product. Their responses, recorded on a 5-point semantic differential scale, were compiled, and the correlation matrix of the attributes was created, as shown below:

Attribute		**Correlation matrix**				
		1	**2**	**3**	**4**	**5**
Taste	**1**	1	.02	.96	.42	.01
Good buy for money	**2**	.02	1	.13	.71	.85
Flavor	**3**	.96	.13	1	.5	.11
Suitable for snack	**4**	.42	.71	.50	1	.79
Provides energy	**5**	.01	.85	.11	.79	1

The correlation matrix indicates that variables 1 and 3, as well as variables 2 and 5, create distinct groups. Variable 4 is more closely associated with the (2,5) group compared to the (1,3) group. Based on the findings, it can be anticipated that the noticeable linear associations among the variables might be accounted for by a maximum of two or three underlying factors.

Initial Factor Method: Principal Components

Prior Communality Estimates: ONE
Eigenvalues of the Correlation Matrix: Total = 5 Average = 1

	1	2	3	4	5
Eigenvalue	2.8531	1.8063	0.2045	0.1024	0.0337
Difference	1.0468	1.6018	0.1021	0.0687	

Proportion	0.5706	0.3613	0.0409	0.0205	0.0067
Cumulative	0.5706	0.9319	0.9728	0.9933	1.0000

	FACTOR1	FACTOR2
TASTE	0.55986	0.81610
MONEY	0.77726	-0.52420
FLAVOR	0.64534	0.74795
SNACK	0.93911	-0.10492
ENERGY	0.79821	-0.54323

Variance explained by each factor

FACTOR1	FACTOR2
2.853090	1.806332

Final Communality Estimates: Total = 4.659423

TASTE	MONEY	FLAVOR	SNACK	ENERGY
0.979461	0.878920	0.975883	0.892928	0.932231

Residual Correlations with Uniqueness on the Diagonal

	TASTE	MONEY	FLAVOR	SNACK	ENERGY
TASTE	0.02054	0.01264	-0.01170	-0.02015	0.00644
MONEY	0.01264	0.12108	0.02048	-0.07493	-0.05518
FLAVOR	-0.01170	0.02048	0.02412	-0.02757	0.00119
SNACK	-0.02015	-0.07493	-0.02757	0.10707	-.01660
ENERGY	0.00644	-0.05518	0.00119	-0.01660	0.06777

Rotation Method: Varimax

Rotated Factor Pattern

	FACTOR1	FACTOR2
TASTE	0.01970	0.98948
MONEY	0.93744	-0.01123
FLAVOR	0.12856	0.97947
SNACK	0.84244	0.42805
ENERGY	0.96539	-0.01563

Variance explained by each factor

FACTOR1	FACTOR2
2.537396	2.122027

Final Communality Estimates: Total = 4.659423

TASTE	MONEY	FLAVOR	SNACK	ENERGY

0.979461 0.878920 0.975883 0.892928 0.932231

Initial Factor Method: Maximum Likelihood
Eigenvalues of the Weighted Reduced Correlation Matrix:
Total = 84.5563187
Average = 16.9112637

	1	2	3	4	5
Eigenvalue	59.7487	24.8076	0.1532	-0.0025	-0.1507
Difference	34.9411	24.6544	0.1557	0.1482	
Proportion	0.7066	0.2934	0.0018	-0.0000	-0.0018
Cumulative	0.7066	1.0000	1.0018	1.0018	1.0000

Factor Pattern

	FACTOR1	FACTOR2
TASTE	0.97601	-0.13867
MONEY	0.14984	0.86043
FLAVOR	0.97908	-0.03180
SNACK	0.53501	0.73855
ENERGY	0.14567	0.96257

Variance explained by each factor

	FACTOR1	FACTOR2
Weighted	59.748704	24.807616
Unweighted	2.241100	2.232582

	TASTE	MONEY	FLAVOR	SNACK	ENERGY
Communality	0.971832	0.762795	0.959603	0.831686	0.947767

Rotation Method: Varimax
Rotated Factor Pattern

	FACTOR1	FACTOR2
TASTE	0.02698	0.98545
MONEY	0.87337	0.00342
FLAVOR	0.13285	0.97054
SNACK	0.81781	0.40357
ENERGY	0.97337	-0.01782

Variance explained by each factor

	FACTOR1	FACTOR2
Weighted	25.790361	58.765959
Unweighted	2.397426	2.076256

TASTE MONEY FLAVOR SNACK ENERGY

The two-factor model, with the factor loadings displayed above, demonstrates a strong fit to the data, as the first two factors account for 93.2% of the total standardized sample variance.i.e., $\left(\frac{\lambda_1 + \lambda_2}{p}\right) x100$, where p is the number of variables. The results also indicate

that there is no obvious separation between the factor loadings for the two factors prior to rotation, but this distinction becomes evident after rotation.

SPSS Commands

File → **New** → **Data** → **Define Variables** → **Statistics** → **Data reduction** → **Factor** → **Variables** (mark variables and click '▶' to enter variables in the box) → **Descriptives** → Statistics ('✔' univariate descriptives , '✔' initial solution) → Corrlation matrix ('✔' coefficients) → **Extraction** → Method (principal components or maximum likelihood) → Extract (Number of factors as 2) → Display (Unrotated factor solution) → Continue → **Rotation** → Method (varimax) → Display (loading plots) → Continue → **Scores** → Display factor score coefficient matrix → Continue → OK.

R codes

```
library(psych)
# Example Data: Suppose we have 5 variables
set.seed(123)  # For reproducibility
data <- data.frame(
  Var1 = rnorm(20, mean = 50, sd = 10),
  Var2 = rnorm(20, mean = 55, sd = 12),
  Var3 = rnorm(20, mean = 60, sd = 8),
  Var4 = rnorm(20, mean = 65, sd = 9),
  Var5 = rnorm(20, mean = 70, sd = 11)
)
# Step 1: Check correlation matrix
cor(data)
# Step 2: Determine number of factors using Scree Plot
fa.parallel(data, fa = "fa")  # Suggests number of factors to retain
# Step 3: Perform Factor Analysis (let's assume 2 factors)
result <- fa(r = cor(data), nfactors = 2, rotate = "varimax", fm = "ml")
# Step 4: View Factor Loadings
print(result$loadings)
#  Step 5: Get detailed output
print(result)
#  Step 6: Get Factor Scores (if you want)
factor_scores <- factor.scores(data, result)
head(factor_scores$scores)
```

Suggested Readings

Chatfield, C. and Collins, A.J. (1990). Introduction to multivariate analysis. *Chapman and Hall publications*.

Johnson, R.A. and Wichern, D.W. (1996). Applied multivariate statistical analysis. *Prentice-Hall of India Private Limited*.

Non-Parametric Tests

Ponnaganti Navyasree, Nobin Chandra Paul, K. Ravi Kumar, Prabhat Kumar and Santosha Rathod

ICAR-National Institute of Abiotic Stress Management, Baramati, Pune-413115

Email: navyaponnaganti97@gmail.com

1. Introduction

A non-parametric test is a type of hypothesis test that does not depend on particular assumptions about the shape of the population distribution or the values of population parameters.This contrasts with parametric tests, which assume certain conditions such as normality and known population parameters. Non-parametric tests, often referred to as distribution-free tests, rely on fewer and less stringent assumptions. Nevertheless, this flexibility frequently results in reduced statistical power when compared to parametric tests. **Parametric tests** are generally more powerful and allow for testing a broader range of alternative hypotheses. They are more appropriate when data are approximately normally distributed and meet other assumptions like homogeneity of variances. However, there are scenarios where data clearly do not meet these assumptions, such as:

- When the outcome is an ordinal variable or involves ranks
- When the data contains significant outliers
- When the distribution of the dependent variable is clearly non-normal
- When assumptions such as: Normality, and Homogeneity of variance are violated

In such cases, non-parametric tests are more suitable for testing the significance of the dataset.

Table 1: Comparison Chart: Parametric vs Non-Parametric Tests

Basis of Comparison	Parametric Test	Non-Parametric Test
Information about population	Completely known	Not known / Unavailable
Basis of test statistic	Normal probability distribution	Non-normal / Arbitrary distribution
Measurement level	Metric scale (interval or ratio)	Any scale (nominal, ordinal, non-normal metric)
Measure of central tendency	Mean	Median
Assumptions	Stringent and specific	Less rigid assumptions

2. Why we go for non-parametric tests

We opt for non-parametric tests when:

a. **Data does not meet parametric assumptions:**

- The distribution is skewed or non-normal.
- Variance is not homogeneous across groups**.**

b. Data are ordinal or categorical:

- When the measurement scale is nominal (*e.g.*, categories) or ordinal (*e.g.*, ranks), parametric tests are not appropriate.

c. Small sample sizes:

- With very small samples, normality tests are unreliable, and non-parametric tests offer a safer alternative.

d. Presence of outliers:

- Non-parametric tests are more robust to outliers, as they often use ranks rather than raw values.

e. Flexibility and general applicability:

- These tests can be used across different types of data without strict distributional requirements.

Advantages of Non-Parametric Tests

- Simple and easy to apply.
- No assumptions are made regarding the underlying population distribution.
- Applicable to any type of data: nominal, ordinal, or non-normally distributed interval/ratio data.
- Fewer assumptions make them more flexible in real-world scenarios.

Disadvantages of Non-Parametric Tests

- Less powerful than parametric tests when parametric assumptions are met.
- No well-developed non-parametric methods for complex models (*e.g.*, testing interactions in ANOVA).
- May discard some information (*e.g.*, ignoring actual values when only ranks are used).
- May require a larger sample size to achieve the same level of power as a parametric test.

Table 2: Frequently used non-parametric tests and their parametric alternatives.

Non-Parametric Test	**Purpose**	**Parametric alternative**

Sign Test	Compare the median/mean of one sample to known value	One sample t-test
Wilcoxon Signed Rank Test (one sample)	Compare one sample to a hypothetical median/mean	One sample t-test
Wilcoxon Signed Rank Test (paired)	Compare two related samples	Paired t-test
Mann–Whitney U Test	Compare two independent samples	Independent (unpaired) t-test
Run Test	Test for randomness in data sequence	-
Kruskal-Wallis Test	Compare >2 independent groups	One-way ANOVA
Friedman Test	Compare >2 related/matched groups	Repeated Measures ANOVA
Spearman's Rank Correlation Test Kendall's Tau Test	Measure linear correlation	Pearson's correlation coefficient

3. Different non-parametric tests and their applications

3.1. Sign Test

The Sign Test is a simple non-parametric test used to evaluate whether the median of a sample differs from a hypothesized value. It does not assume normal distribution and only considers the direction (sign) of the difference from the hypothesized median.

Assumptions:

- Data are independent.
- Measurement scale is at least ordinal.

Example:

Suppose 10 days of sulphur oxide emissions (in tons) from an industrial plant are recorded as: **18, 25, 20, 9, 13, 14, 10, 19, 26, 30**

We want to test whether the median daily emission is less than 23.5 tons.

- Null Hypothesis (H_0): Median = 23.5
- Alternative Hypothesis (H_1): Median < 23.5

This test is appropriate when the exact distribution of the data is unknown, making it ideal for pollution studies and small environmental datasets.

3.2. Wilcoxon Signed Rank Test (One Sample)

The one-sample Wilcoxon Signed Rank Test is a non-parametric alternative to the one-sample t-test. It tests whether the median of a single sample differs significantly from a specified value and considers both magnitude and direction of the differences.

Assumptions:

- Data are paired or one sample from a symmetric distribution.
- Measurement scale is at least interval.

Example:

Marks of 10 students are recorded as: 71, 79, 40, 70, 82, 72, 60, 76, 69, 75 We test whether the median score is 67.

This test is useful when evaluating performance or yield where data may not be normally distributed.

3.3. Wilcoxon Signed Rank Test (Paired Samples)

This test compares two related samples or repeated measurements on a single sample to determine whether their mean ranks differ in their population. It is a non-parametric equivalent of the paired t-test.

Assumptions:

- Paired observations are randomly and independently selected.
- The distribution of differences is symmetric.

Example: Consider cholesterol levels before and after drug administration in 10 patients. In agricultural research, similar analysis can compare soil quality before and after treatment or yield before and after applying a bio-fertilizer.

This test helps detect significant changes when applying treatments over time.

3.4. Mann–Whitney U Test

The Mann–Whitney U Test, which is also referred to as the Wilcoxon rank-sum test, is a non-parametric method used to assess whether two independent groups originate from the same distribution. It serves as an alternative to the unpaired t-test.

Assumptions:

- Independent observations.
- The response variable is at least ordinal.

Example: Bacteria counts from two different culture techniques:

- Culture 1: 27, 31, 26, 25
- Culture 2: 32, 29, 35, 28

This test helps determine whether the two methods result in significantly different bacterial growth, relevant in agriculture for testing different inoculation techniques.

3.5. Kruskal-Wallis H Test

The Kruskal-Wallis test is the non-parametric equivalent of the one-way ANOVA. It is used to compare more than two independent groups and determine whether their distributions differ.

Assumptions:

- Independent observations.
- Ordinal or continuous response variable.

Example 1: Comparing crop yields from three different fertilizer treatments:

- Treatment A: 25, 30, 28
- Treatment B: 35, 40, 38
- Treatment C: 45, 48, 50

This test is ideal when data do not meet ANOVA assumptions and is widely used in agronomic trials.

Example 2: Yields from three irrigation systems (e.g., Irrigation A: 420, 430, 410, 425, 415 kg/ha; Irrigation B: 450, 460, 445, 455, 440 kg/ha; Irrigation C: 480, 490, 475, 485, 470 kg/ha) can be tested for differences in medians. The test ranks all observations and compares rank sums across groups. The parametric alternative, one-way ANOVA, assumes normality and equal variances, which may not be held for heterogeneous agricultural data.

3.6. Friedman Test

The Friedman Test is a non-parametric alternative to repeated measures ANOVA. It evaluates differences in treatments across multiple test attempts or blocks.

Assumptions:

- Data are ordinal or continuous.
- Each block (e.g., field or plot) is exposed to all treatments.

Example: For example, nitrogen levels in five plots at three time points (e.g., Time 1: 120, 125, 118, 130, 122 mg/kg; Time 2: 115, 120, 112, 125, 118 mg/kg; Time 3: 110, 115, 108, 120, 112 mg/kg) can be tested for differences. The test ranks observations within each subject across

time points and evaluates rank consistency. Its parametric alternative, repeated-measures ANOVA, assumes normality and sphericity, which may be violated in agricultural experiments.

3.7. Spearman's Rank Correlation Coefficient

The Spearman's Rank Correlation measures the monotonic relationship between two variables. It is the non-parametric counterpart of Pearson's correlation.

Assumptions:

- Variables are ordinal or continuous.
- The relationship is monotonic.

Example 1: Examining the correlation between the amount of fertilizer used and crop yield. If yield increases consistently (but not linearly), Spearman's rank correlation is appropriate.

Example 2: Soil moisture levels (e.g., 20, 22, 18, 25, 23, 21, 19, 24, 20, 22%) and corresponding yields (e.g., 450, 460, 440, 480, 470, 455, 445, 475, 450, 460 kg/ha) can be tested for a monotonic association. The test computes the correlation between ranks of the two variables. The parametric alternative, Pearson's correlation, assumes a linear relationship and normality, which may not apply to non-linear or ordinal agricultural data.

3.8. Kendall's Tau

Kendall's Tau test assesses the strength of a monotonic relationship between two variables, similar to Spearman's test, but uses a different approach based on concordant and discordant pairs. It is useful in agriculture for correlating environmental factors with outcomes, such as rainfall and crop yield.

For example, rainfall amounts (*e.g.*, 50, 55, 45, 60, 58, 52, 48, 62, 50, 54 mm) and yields (e.g., 450, 460, 440, 480, 470, 455, 445, 475, 450, 460 kg/ha) can be tested. The test counts pairs where the variables move in the same or opposite directions. Like Spearman's, its parametric alternative is Pearson's correlation, which is less robust to non-linear relationships or outliers common in agricultural data

3.9. Run Test (Wald-Wolfowitz Runs Test)

The Run Test for randomness is a nonparametric test used to assess whether a sequence of binary outcomes, such as high or low values relative to a threshold, occurs randomly, making it valuable for analyzing time-series data in agricultural research. It is useful in detecting trends or cycles in time series data.

For example, we consider a sequence of daily rainfall measurements (in mm) over 20 days in a farming region to determine if the pattern of high and low rainfall days is random, which is

critical for understanding irrigation needs or crop growth patterns. The data, consisting of rainfall values (12.5, 8.2, 15.3, 7.9, 18.1, 6.4, 13.7, 9.8, 16.2, 5.5, 14.0, 10.1, 17.4, 6.8, 19.0, 8.5, 11.3, 15.9, 7.2, 12.8 mm), are classified as above (1) or below (0) the median rainfall (approximately 12.65 mm), resulting in a binary sequence (0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1). The Run Test evaluates the null hypothesis that the sequence is random against the alternative that it exhibits a non-random pattern, such as clustering or excessive alternation, at a 5% significance level. By counting the number of runs (consecutive sequences of the same value), the test determines if the observed number of runs deviates significantly from the expected number under randomness. There is no direct parametric alternative, as the Run Test specifically addresses randomness in binary sequences, making it uniquely suited for detecting temporal patterns in agricultural data like rainfall.

4. Conclusions

Non-parametric tests are invaluable tools in agricultural research, particularly when data do not conform to the assumptions of parametric tests. They offer robust and flexible methods for hypothesis testing, ranking, correlation analysis, and treatment comparison, especially when working with small samples, ordinal data, or outliers. By selecting appropriate non-parametric tests, researchers can derive valid insights from a wide variety of datasets.

5. Non-parametric statistical tests using R

##... Implementation of Non-Parametric Tests Using R …##

```
# 1. Sign Test
# Daily sulphur oxide emissions
data <- c(18, 25, 20, 9, 13, 14, 10, 19, 26, 30)
median_hyp <- 23.5
# Count signs
signs <- sign(data - median_hyp)
n_positive <- sum(signs > 0)
n_negative <- sum(signs < 0)

# Use binomial test as sign test
stats::binom.test(n_negative, n_negative + n_positive, p = 0.5, alternative = "greater")
stats::binom.test(n_positive, n_negative + n_positive, p = 0.5, alternative = "greater")

# 2. Wilcoxon Signed Rank Test (One-sample)
# Example: Student marks, H0: median = 67 vs HA: median != 67
marks <- c(71, 79, 40, 70, 82, 72, 60, 76, 69, 75)
wilcox_one_sample <- wilcox.test(marks, mu = 67, alternative = "two.sided", exact = TRUE)
print("Wilcoxon Signed Rank Test (One-sample) for Student Marks:")
print(wilcox_one_sample)
# Interpretation: If p-value < 0.05, reject H0 (median != 67)
```

```
# 3. Wilcoxon Signed Rank Test (Paired)
# Example 1: Cholesterol levels before and after drug administration
# Using agriculture data: Soil pH before and after fertilizer application
soil_pH <- data.frame(
  Initial = c(6.10, 6.40, 6.70, 6.20, 6.10, 6.30, 6.80, 6.50, 6.70, 6.40),
  Final = c(7.92, 7.28, 6.88, 6.53, 6.78, 6.87, 6.99, 7.36, 7.57, 6.51)
)
wilcox_paired <- wilcox.test(soil_pH$Initial, soil_pH$Final, paired = TRUE, alternative = "two.sided")
print("Wilcoxon Signed Rank Test (Paired) for Soil pH:")
print(wilcox_paired)
# Interpretation: If p-value < 0.05, reject H0 (difference in medians != 0)

# Example 2
# Wilcoxon Signed Rank Test (Paired Samples)
initial <- c(240, 237, 264, 233, 236, 234, 265, 241, 261, 259)
final <- c(228, 222, 262, 224, 240, 237, 264, 219, 251, 254)
wilcox.test(initial, final, paired = TRUE, alternative = "two.sided")

# 4. Mann-Whitney U Test
# Example: Bacteria counts for two cultures
# Using agriculture data: Crop yields (kg/ha) from two different fertilizers
# Crop yields (kg/ha) using two fertilizers
fertilizer_1 <- c(450, 480, 430, 460, 470, 455, 440, 465)  # Fertilizer A
fertilizer_2 <- c(500, 490, 510, 485, 495, 505, 520, 488)  # Fertilizer B

mann_whitney <- wilcox.test(fertilizer_1, fertilizer_2, alternative = "two.sided", exact = TRUE)
print("Mann-Whitney U Test for Crop Yields:")
print(mann_whitney)
# Interpretation: If p-value < 0.05, reject H0 (medians of two groups differ)

# Example 2:
# Mann-Whitney U Test (Independent Samples)
culture1 <- c(27, 31, 26, 25, 33)
culture2 <- c(32, 29, 35, 28, 37)
wilcox.test(culture1, culture2, alternative = "two.sided")

# 5. Wald-Wolfowitz Runs Test
library(randtests)  # For runs test

# Example 1: Run Test for Randomness in Rainfall Data
# Daily rainfall data (in mm) for 20 days
rainfall <- c(5.2, 7.4, 3.1, 6.0, 4.8, 2.9, 3.5, 5.9, 7.1, 4.2,
              3.8, 6.5, 2.7, 4.9, 5.3, 3.3, 6.8, 4.6, 3.2, 5.7)

# Perform runs test on numerical data
```

```
runs.test(rainfall)
# Interpretation: If p-value < 0.05, reject H0 (sequence is not random)

# Example 2:
# Data input: Daily rainfall values (mm) for 20 days
  rainfall <- c(12.5, 8.2, 15.3, 7.9, 18.1, 6.4, 13.7, 9.8, 16.2, 5.5,
                14.0, 10.1, 17.4, 6.8, 19.0, 8.5, 11.3, 15.9, 7.2, 12.8)

  # Classify rainfall as above (1) or below (0) the median
  median_rainfall <- median(rainfall)
  binary_sequence <- ifelse(rainfall > median_rainfall, 1, 0)

  # Perform Wald-Wolfowitz Runs Test
  run_test_result <- runs.test(binary_sequence, threshold = 0.5, alternative = "two.sided")
  # Output results
  cat("Run Test for Randomness in Rainfall Data:\n")
  cat("Daily Rainfall (mm):", sprintf("%.1f", rainfall), "\n")
  cat("Median Rainfall (mm):", sprintf("%.1f", median_rainfall), "\n")
  cat("Binary Sequence (1 = above median, 0 = below median):", binary_sequence, "\n")
  cat("Number of runs:", run_test_result$runs, "\n")
  cat("P-value:", run_test_result$p.value, "\n")
  cat("Conclusion: ",
      if (run_test_result$p.value < 0.05) {
        "Reject H0: Evidence suggests the rainfall sequence is not random at 5% significance
level."
      } else {
        "Fail to reject H0: No evidence that the rainfall sequence is not random at 5%
significance level."
      }, "\n")

  # 6. Kruskal-Wallis H-Test
# Example: Comparing crop yields across three different irrigation methods
irrigation_A <- c(420, 430, 410, 425, 415)  # Yields (kg/ha)
irrigation_B <- c(450, 460, 445, 455, 440)
irrigation_C <- c(480, 490, 475, 485, 470)
yields_all <- c(irrigation_A, irrigation_B, irrigation_C)
groups <- factor(rep(c("A", "B", "C"), each = 5))
kruskal_wallis <- stats::kruskal.test(yields_all ~ groups)
print("Kruskal-Wallis Test for Irrigation Methods:")
print(kruskal_wallis)
# Interpretation: If p-value < 0.05, reject H0 (medians differ across groups)

# 7. Friedman Test
# Example: Comparing soil nutrient levels across three time points (repeated measures)
# Data: Nitrogen levels (mg/kg) in soil at three time points for 5 plots
nitrogen <- data.frame(
  Plot = 1:5,
```

```
  Time1 = c(120, 125, 118, 130, 122),
  Time2 = c(115, 120, 112, 125, 118),
  Time3 = c(110, 115, 108, 120, 112)
)
nitrogen_matrix <- as.matrix(nitrogen[, 2:4])  # Convert to matrix for friedman.test
friedman_result <- stats::friedman.test(nitrogen_matrix)
print("Friedman Test for Soil Nitrogen Levels:")
print(friedman_result)
# Interpretation: If p-value < 0.05, reject H0 (medians differ across time points)

# 8. Spearman's Rank Correlation Test
# Example: Correlation between soil moisture and crop yield
soil_moisture <- c(20, 22, 18, 25, 23, 21, 19, 24, 20, 22)  # % moisture
crop_yield <- c(450, 460, 440, 480, 470, 455, 445, 475, 450, 460)  # kg/ha
spearman_result <- cor.test(soil_moisture, crop_yield, method = "spearman", exact = FALSE)
print("Spearman's Rank Correlation Test for Soil Moisture and Crop Yield:")
print(spearman_result)
# Interpretation: If p-value < 0.05, reject H0 (no monotonic relationship)

# 9. Kendall's Tau Test
# Example: Correlation between rainfall and crop yield
rainfall <- c(50, 55, 45, 60, 58, 52, 48, 62, 50, 54)  # mm
crop_yield <- c(450, 460, 440, 480, 470, 455, 445, 475, 450, 460)  # kg/ha
kendall_result <- cor.test(rainfall, crop_yield, method = "kendall", exact = FALSE)
print("Kendall's Tau Test for Rainfall and Crop Yield:")
print(kendall_result)
# Interpretation: If p-value < 0.05, reject H0 (no monotonic relationship)
```

References

Conover, W. J. (1999). *Practical nonparametric statistics* (Vol. 350): john wiley & sons.

Dixon, W. J., & Mood, A. M. (1946). The statistical sign test. *Journal of the American Statistical Association, 41*(236), 557-566.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association, 32*(200), 675-701.

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association, 47*(260), 583-621.

Lehmann, E. L. (1975). Statistical methods based on ranks. *Nonparametrics. San Francisco, CA, Holden-Day, 2*.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 50-60.

Wilcoxon, F. (1992). Individual comparisons by ranking methods *Breakthroughs in statistics: Methodology and distribution* (pp. 196-202): Springer.

Fuzzy Set and Fuzzy Time Series Forecasting

Amit Saha[1], Santosha Rathod[2] and Nobin Chandra Paul[2]
[1]Directorate General of Commercial Intelligence and Statistics, Government of India, Kolkata
[2]ICAR - National Institute of Abiotic Stress Management, Baramati
Email: amit.saha20@gov.in

1. Introduction:

Time series forecasting is a widely recognized technique for making predictions across various domains. Its appeal lies in the ease with which time series data can be analyzed to produce forecast values. Moreover, many real-world datasets are fundamentally time series.In conventional time series forecasting, data is typically considered as fixed values. However, there are situations where the data might be incomplete or imprecise, such as with river water levels or temperature measurements. Fuzzy techniques are particularly useful in these scenarios as they can manage the inherent uncertainty in the data.Fuzzy data is frequently found in fields such as artificial intelligence, quality control, biology, psychometrics, agriculture, social economics, and image recognition. Utilizing a fuzzy time series model can enhance the effectiveness of data use and improve the accuracy of forecasts.

2. Situations where fuzzy techniques are useful?

1. Fuzzy techniques are utilized in situations where-
2. 1. Decisions made by individuals are at play.
3. 2. The data lacks precision.
4. 3. The assumed distributions do not hold true.
5. 4. There is insufficient historical data, making existing time series models unusable.

3. Fuzzy Set:

3.1 Fuzzy set theory:

In 1965, Zadeh introduced the concept of a fuzzy set, defining it as “a collection of objects that have varying degrees of membership.” He explored the different characteristics of fuzzy sets. Various operations on fuzzy sets, such as union, intersection, and complement, were examined and found to differ from the standard operations of traditional sets. In a conventional set, an element is either a member or not a member of that set. Consequently, our answers regarding the presence or absence of a specific element in a set are always definitive, either Yes or No. For instance, consider a set of overweight individuals; each person is classified as either overweight or not, with no ambiguity in between. This limitation highlights a significant disadvantage of traditional sets. In reality, it is often challenging to categorize certain concepts solely based on their presence or absence in a particular set. For example, a person may be deemed overweight if their weight is 100 kg or above, while they are considered not overweight if their weight is below 70 kg. However, this provides no information about individuals whose weights fall within that range. Thus, there exists a gray area where one cannot easily determine

a clear yes or no. Conversely, fuzzy sets utilize a membership function to assign different degrees of membership to elements. This approach allows for a gradual transition from yes to no rather than a stark change. Typically, the degree of membership ranges from 0 to 1, with an element's affiliation to a specific set becoming stronger as its degree of membership increases.

Definition: A fuzzy set is characterized as a collection of ordered pairs, where each individual pair (fuzzy singleton) consists of an element and its degree of membership in the fuzzy set. In mathematical terms, a fuzzy set can be expressed as $F = \left[\left(x_{i,}\mu_F(x_{i,})\right)\right]$

Where, $i = 1,2,3 \ldots n$, x_i is the i^{th} element which is a member of F and $\mu_F(x_{i,})$ represents the degree of membership of x_i in F. Fuzzy set can also be written as: $F = I_1/x_1 + I_2/x_2 + \ldots I_n/x_n$ (Only for discrete and finite fuzzy set), where, I_1, I_2,…, I_n are the intervals of the defined fuzzy set and the above operations are not algebric operations.

3.2 ***Linguistic variable***: Linguistic variables are quantitative measures used to depict fuzzy numbers. Instead of numerical values, linguistic variables take the form of words or phrases.

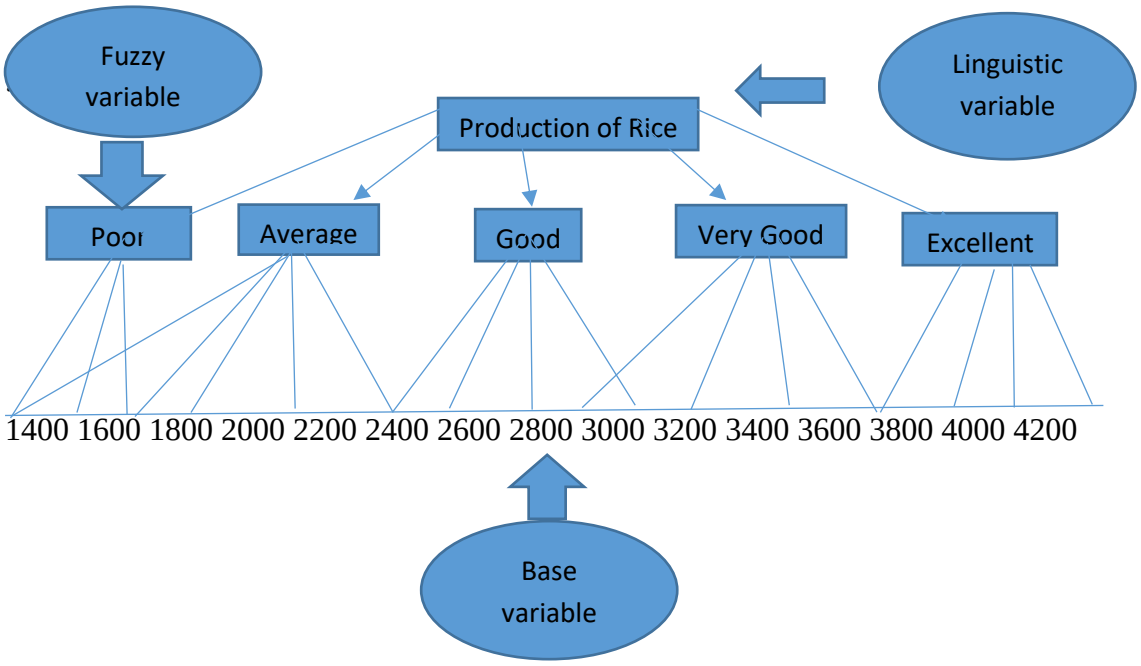


Figure 1: Illustration of linguistic variables and fuzzy

For example, if we consider the height of a person, we can represent it using linguistic expressions such as "very much tall," "very tall," "tall," "more than medium tall," "medium tall," "short," "very short," and "very much short." Each of these linguistic variables is associated with different types of membership functions. The range of these linguistic variables

is determined by the chosen membership function. Figure 2 illustrates a visual representation of fuzzy variables and linguistic variables.

3.3 *Membership Function:*

Zadeh introduced the concept of membership functions in his 1965 paper "Fuzzy Sets," published in Information and Control. Membership functions reflect the uncertainty of data, indicating the degree to which elements belong to a fuzzy set. Numerous types are documented, with the choice depending on the concept being illustrated and its context. These functions can be represented graphically, showing various shapes and properties. Here are the different types of membership functions:s:

1. Triangular membership functions.
2. Trapezoidal membership functions.
3. S membership functions.

1. Triangular membership functions are characterized by three parameters and are represented as piecewise linear functions. In their research, Saha et al. (2019) applied triangular membership functions in space-time series modeling and prediction.Mathematically, it can be written as:-

$$w_F = \begin{cases} \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & x > c \end{cases} \tag{2}$$

2. A trapezoidal membership function is characterized by four parameters. This function can be mathematically expressed as follows:

$$w_F = \begin{cases} \frac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \frac{d-x}{d-c}, & c \le x \le d \\ 0, & x > d \end{cases} \tag{3}$$

3. The S membership function is characterized by two parameters and can be represented in the following way:

$$w_F(x; a, b) = \frac{1}{1 + e^{-b(x-a)}} \tag{4}$$

The midpoint an d slope value are denoted by a and b respectively and b are required to be positive value. Another important thing of sigmoidal membership function is that it never goes to 0 or 1.

Various types of membership functions include G and L open shoulder functions, bell-shaped functions, and Z functions, among others. A graphical illustration is included below to clarify the concept of membership functions.

Example (Fig. 2): Consider that an individual is classified as tall if their height is 180 cm or greater. This means that if Ratan's height is 180 cm, he qualifies as a tall person; otherwise, he does not. If Ratan's height is 178 cm, he does not meet the criteria to be considered tall according to the crisp set, even though his height is quite close to 180 cm. In this case, a person can either be tall or not tall.her tall or not tall.

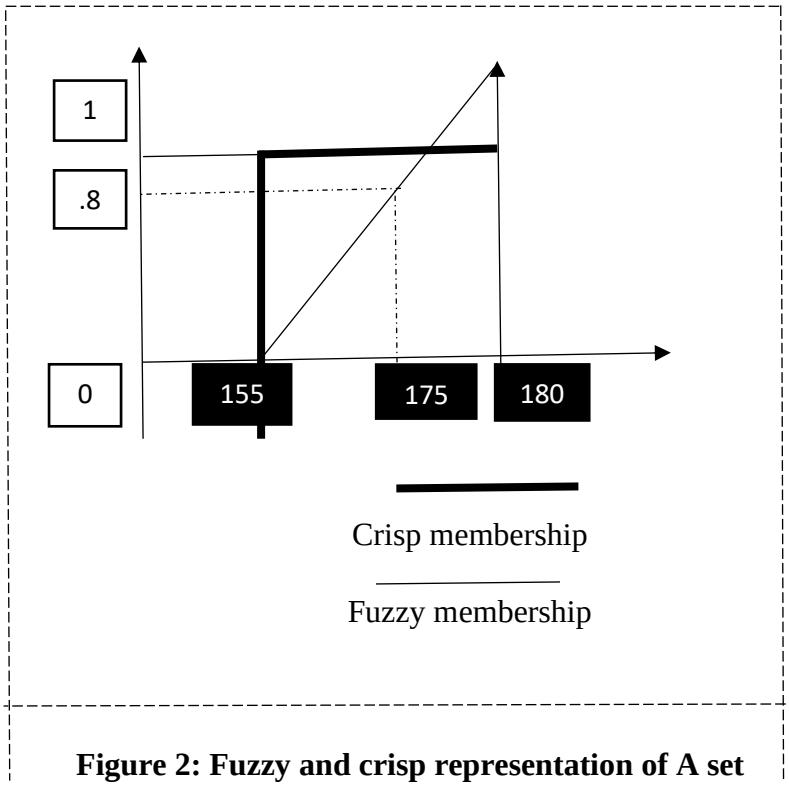


Figure 2: Fuzzy and crisp representation of A set

There isn't a strict boundary between tall and not tall. Defining someone as tall solely based on being 180 cm or taller is too rigid. Instead, a fuzzy set approach captures the concept of "in between" using a membership function. For example, if Ratan is 175 cm tall, his degree of tallness can be quantified with a membership grade of 0.8.

3.4 Fuzzy relation:

The connection between events is defined by relation. A fuzzy relation differs from a crisp relation. Below, examples of both crisp and fuzzy relations are provided.les.

$$A = \{X_1, X_2, X_3\}$$

$$B = \{Y_1, Y_2, Y_3\}$$

Where, A and B are two universal set. X denotes the size of the flower and Y represents the market price.

X_1= Small , X_2= Medium, X_3=Large and Y_1=Low, Y_2=Medium, Y_3=High.

The crisp relation $X \rightarrow Y$ is defined as follows (Table 1):

Table 1: The crisp relation between X and Y

	Low (Y_1)	**Medium (Y_2)**	**High (Y_3)**
Small (X_1)	1	0	0
Medium (X_2)	0	1	0
Large (X_3)	0	0	1

The above table represents the crisp relationship between A and B. Here, 1 represents an association and 0 represents null assosciation. In linguistic term, one can write the above table as:

I. If the size of flower is small then the market price will be low.

II. If the size of flower is medium then the market price will be medium.

III. If the size of flower is large then the market price will be high.

The fuzzy relation $X \rightarrow Y$ is defined as follows (Table 2):

Table 2: The fuzzy relation between X and Y

	Low (Y_1)	**Medium (Y_2)**	**High (Y_3)**
Small (X_1)	1	0.3	0
Medium (X_2)	0.3	1	0.3
Large (X_3)	0	0.3	1

The above table represents the fuzzy relationship between A and B. Here, the values represent the membership grade. In linguistic term, one may write the above table as-

- When the flower is small, it will have a low market price with a membership grade of 1 and a medium market price with a membership grade of 0.3.
- For flowers classified as medium in size, the market price will be medium with a membership grade of 1, low with a membership grade of 0.3, and high with a membership grade of 0.3.
- If the flower is large, it will have a high market price with a membership grade of 1 and a medium price with a membership grade of 0.3.

Definition: A relation R which is a mapping from the cartesian space $X \times Y$ to the closed interval [0,1]- This relation is called as fuzzy relation, where X and Y are two crisp set.

Membership function which represents the strength of mapping is denoted by:-

$$F_A(x,y) = \min\left(F_A(x), F_B(y)\right) \quad (5)$$

Where, A and B are two fuzzy sets.

Fuzzy cartesian product:

$$A = 0.5/x_1 + 0.6/x_2 + 0.2/x_3$$

$$B = 0.3/y_1 + 0.9/y_2$$

$$A \times B = R = \begin{bmatrix} 0.3 & 0.5 \\ 0.3 & 0.6 \\ 0.2 & 0.2 \end{bmatrix}$$

The above matrix is formed by taking the minimum value of each pair of elements.

3.5 *Max-Min Composition operator*:

There are numerous methods to merge the relations. The union or intersection operator can be employed to join the relation. The max-min composition operation is a type of operator used for merging relations.

Let, two fuzzy relation A and B; then max-min composition between A and B is defined as:

$$R = AoB = \left[(x,z), max\left\{min\left(F_A(x,y), F_B(y,z)\right)\right\}\right] \quad \text{for all} \quad x \leftarrow X \;,\; y \leftarrow Y \;,\; z \leftarrow Z \quad (6)$$

A is defined on $X \times Y$.

B is defined on $Y \times Z$.

Example:

Let, $A = \begin{bmatrix} 0.7 & 0.6 \\ 0.2 & 0.5 \\ 0.3 & 0.6 \end{bmatrix}$ and $B = \begin{bmatrix} 0.8 & 0.4 \\ 0.4 & 0.1 \end{bmatrix}$

$$R = AoB = \begin{bmatrix} 0.7 & 0.4 \\ 0.4 & 0.2 \\ 0.4 & 0.3 \end{bmatrix}$$

1st term of AoB,

$$r_{11} = max[min(a_{11}, b_{11}), min(a_{12}, b_{21})]$$

$$= max[min(0.7,0.8), min(0.6,0.4)]$$

$$= max[0.7,0.4]$$

$$= 0.7$$

Last term of AoB,

$$r_{32} = max[min(a_{31}, b_{12}), min(a_{32}, b_{22})]$$
$$= max[min(0.3,0.4), min(0.6,0.1)]$$
$$= max[0.3, 0.1]$$
$$= 0.3$$

3.6 Operations on fuzzy sets:

The fundamental operations on fuzzy sets include union, intersection, and complement. These operations are also foundational to classical set theory. Although the names of these operations are identical in both classical and fuzzy sets, the methods used to perform them differ.

Union: Let, two fuzzy sets are A and B and their membership functions are $m_A(x)$ and $m_B(x)$ respectively. Then, the union of A and B is the maximum value between the $m_A(x)$ and $m_B(x)$. We denote the union as C, then the membership function of C is represented as:

$m_C(x)=m_{A \cup B}(x) = max[m_A(x), m_B(x)]$
(7)

Intersection: The intersection of A and B is minimum value between the $m_A(x)$ and $m_B(x)$. If, the intersection is denoted by C, then the membership function of C is given as:

$$m_C(x)=m_{A \cap B}(x) = min[m_A(x), m_B(x)] \quad (8)$$

Complement: Let, A is a fuzzy set and its membership function is represented as $m_A(x)$. Then, the complement of A is denoted as $\bar{A}$ has a membership function is,

$$m_{\bar{A}}(x) = 1 - m_A(x) \quad (9)$$

4. Fuzzy time series:

Fuzzy time series refers to time series data that incorporates fuzzy elements grounded in fuzzy set theory. Song and Chissom (1993) were the first to introduce fuzzy time series models using the max-min composition operation. They created a detailed procedure for generating forecasts and evaluated the proposed model to confirm its robustness.

4.1 Definition: Let, fuzzy sets $f_i(t)$ are defined on $Y(t)$ and $F(t)$ is the collection of $f_i(t)$. Then, $F(t)$ is known as a fuzzy time series on $Y(t)$.

4.2 Fuzzification: It involves transforming precise data into more ambiguous data.

4.3 Fuzzy logical relationship: If a fuzzy set A_1 is caused only by A_2, then fuzzy logical relationship is denoted by $A_2 \rightarrow A_1$.

4.4 Fuzzy logical relationship group: If some fuzzy logical relationship are $A_2 \rightarrow A_1, A_2 \rightarrow A_3, A_2 \rightarrow A_2$; then fuzzy logical relationship group is denoted by $A_2 \rightarrow A_1, A_2, A_3$.

4.5 Defuzzification: Defuzzification refers to the process of transforming fuzzy data into a precise format. In fact, defuzzification serves as the opposite of the fuzzification process. Numerous techniques for the fuzzification process are available in literature. Below are descriptions of several defuzzification methods:

***4.5.1 Centroid Method*:** The centroid method is a type of weighted average that calculates the centroid value of different sets. This approach is also referred to as the center of area method. The mathematical expression for this technique is-

$$Y = \frac{\sum_{i=1}^{n} \omega_F(x_i) x_i}{\sum_{i=1}^{n} \omega_F(x_i)}$$

(10)

Where, Y is the crisp output.

$\omega_F(x_i)$ is the fuzzy output value of the membership function of x_i

x_i is the value of the element on the $x\ axis_i$ in the fuzzy set F.

4.5.2 Maximum membership method: This technique provides a precise output that corresponds to the value associated with the highest degree of membership. It can be articulated as: $\omega_F(x_\$) \geq \omega_F(x)$ for all $x \in F$;

Where, $x_\$$ is the value associated with the maximum value of membership.

4.5.3 Average maximum membership method: It resembles the maximum membership method, but the key distinction is that it can encompass additional points beyond just the maximum point. It may also incorporate points that fall within a certain range.

5. General Steps involved in Fuzzy Time Series Forecasting Model

Many fuzzy time series forecasting models adhere to a set of steps during the forecasting process.

Step-1: Fixing the universe of discourse which is defined as- $U = [U_{min} - U_1, U_{max} - U_2]$, where U_{min} and U_{max} are minimum and maximum value of the data and U_1 and U_2 are two

any two positive values which are selected by the modeler properly. Establish the appropriate universe of discourse to encompass complete time series data.

Step 2: Segment the universe of discourse or establish the intervals.

Step 3: Create fuzzy sets within the universe of discourse.

Step 4: Fuzzify the data based on the intervals and the corresponding fuzzy sets defined in steps 2 and 3.

Step 5: Establish the fuzzy logical relationship (FLR).

Step 6: Formulate the fuzzy logical relational groups.

Step 7: Predict the time series data.

Step 8: Defuzzify the projected fuzzified results.

Suggested Readings

- Saha, A., Singh, K., Ray, M., Rathod, S., & Dhyani, M. (2022). Fuzzy rule–based weighted space–time autoregressive moving average models for temperature forecasting. *Theoretical and Applied Climatology*, *150*, Article 04230. https://doi.org/10.1007/s00704-022-04230-1
- Saha, A., Singh, K. N., Ray, M., Kumar, S., & Rathod, S. (2019). A new approach for spatio-temporal modelling and forecasting based on fuzzy techniques in conjunction with K-means clustering. *Journal of the Indian Society of Agricultural Statistics, 73*(2), 111–120.
- Song, Q., & Chissom, B. S. (1993a). Forecasting enrollments with fuzzy time series—Part I. *Fuzzy Sets and Systems*, *54*(1), 1–10.
- Song, Q., & Chissom, B. S. (1993b). Fuzzy time series and its models. *Fuzzy Sets and Systems*, *54*(3), 269–277.
- Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353.

Quantile Regression: An Overview

Mrinmoy Ray
AKMU, ICAR-Indian Agricultural Research Institute, New Delhi-110012
Email: mrinmoy4848@gmail.com

1. Introduction:

Regression analysis is a statistical method widely used in agriculture to analyze the relationship between multiple quantitative variables and predict one variable based on the others. For instance, in agriculture, regression analysis can be employed to forecast crop yield by examining its correlation with meteorological factors like temperature, rainfall, and relative humidity. This technique helps uncover relationships between different variables and allows the creation of equations to summarize or characterize a set of facts. Quantile regression is a statistical technique that goes beyond traditional regression analysis by providing a more comprehensive understanding of the relationship between variables. While ordinary least squares regression focuses on estimating the conditional mean of the dependent variable, quantile regression enables the estimation of different conditional quantiles. This flexibility allows for a more nuanced examination of the relationship between variables across the entire distribution of the dependent variable. Quantile regression is particularly useful when dealing with data that exhibits heteroscedasticity, outliers, or non-normality. It offers a robust approach to modeling, as it is less sensitive to extreme observations and provides a more complete picture of how various predictors affect different portions of the response variable's distribution. The technique has gained significant attention and popularity in various fields, including economics, finance, social sciences, and healthcare, where researchers and practitioners are often interested in understanding not only the average effects but also the effects at different points of the distribution. In this chapter, we will delve into the principles and application of quantile regression. We will explore how it differs from ordinary least squares regression, discuss its advantages and disadvantages, and provide practical guidance on how to interpret and utilize the results. By understanding and harnessing the power of quantile regression, researchers and analysts can gain valuable insights into the relationships between variables, uncover hidden patterns, and make more informed decisions in a wide range of disciplines. Quantile regression offers several advantages over conventional regression methods, such as ordinary least squares (OLS), that focus on estimating the conditional mean of the dependent variable. These advantages include:

i) **Capturing heterogeneity:** Conventional regression assumes a homogeneous relationship between predictors and the response variable across the entire distribution. In contrast, quantile regression allows for heterogeneity by estimating the conditional quantiles at different points along the distribution. This provides a more complete understanding of how the predictors affect the response variable at various quantiles, accommodating diverse patterns and capturing important variations that may be missed by average effects.

ii) **Robustness to outliers:** OLS regression is highly sensitive to outliers, as it aims to minimize the squared differences between the observed and predicted values. Quantile regression, on the other hand, is more robust to outliers because it estimates the conditional quantiles by minimizing the absolute differences. This robustness enables the analysis to be less influenced by extreme observations, making it particularly suitable for data with heavy tails or outliers.

iii) **Handling skewed or non-normal distributions:** Conventional regression assumes that the errors follow a symmetric and normally distributed pattern. However, many real-world datasets exhibit skewed or non-normal distributions. Quantile regression does not make any assumptions about the distribution of errors, allowing it to handle data with non-normality effectively. It provides a flexible modeling approach that accommodates various distributional shapes.

iv) **Comprehensive inference:** Quantile regression offers a richer set of inference tools compared to conventional regression. By estimating multiple conditional quantiles, it provides a more comprehensive view of the relationship between predictors and the response variable across the entire distribution. This allows researchers to gain insights into the conditional effects at different quantiles, explore variations in the relationship, and detect potential heterogeneity.

v) **Policy evaluation and risk analysis:** Quantile regression is valuable for policy evaluation and risk analysis. By estimating different quantiles, it enables the assessment of the effects of interventions or policy changes at various points of the distribution. This information is particularly useful when examining the impact on different segments of the population, identifying vulnerable groups, or evaluating the effectiveness of policies under different scenarios.

Overall, the advantages of quantile regression make it a powerful tool for analyzing data with heterogeneous relationships, outliers, and non-normal distributions. Its flexibility and robustness enhance our understanding of the data, provide more nuanced insights, and enable better decision-making in a wide range of fields.

2. Methodology

Linear Regression

We consider a simple linear model with one predictor variable and a linear regression function. As an example, consider the following model:

$$\boldsymbol{y_i = \beta_0 + \beta_1 x_i + \varepsilon_i}$$

Where $\boldsymbol{y_i}$ is the value of the response variable in the ith trial $\boldsymbol{\beta_0}$ and $\boldsymbol{\beta_1}$ are parameters, $\boldsymbol{x_i}$ is the value of the predictor variable in the ith trial, $\boldsymbol{\varepsilon_i}$ is a random error term with mean zero and variance σ^2 and $\boldsymbol{\varepsilon_i}$ and $\boldsymbol{\varepsilon_j}$ are uncorrelated so that their covariance is zero. A simple and linear regression model is referred to as a regression model. It is "simple" in the sense that there is only one predictor variable, and "linear" in the sense that all parameters appear in a linear relationship with the predictor variables. The parameters $\boldsymbol{\beta_0}$ and $\boldsymbol{\beta_1}$ in regression model are called regression coefficients, $\boldsymbol{\beta_1}$ is the slope of the regression line. It indicates the change in the mean of the probability distribution of y per unit increase in x. The parameter β0 is the y intercept of the regression line. When the scope of the model includes x = 0, $\boldsymbol{\beta_0}$ gives the mean of the probability distribution of y at x = 0. When the scope of the model does not cover x = 0, $\boldsymbol{\beta_0}$ does not have any particular meaning as a separate term in the regression model. It is simple to extend this model to include more than one predictor variable. If the linear model includes more than one predictor variable, it is referred to as a multiple linear regression model. For example, if we have p predictor variables, we can formulate a multiple linear regression model as

$$\boldsymbol{y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} + \varepsilon_i}$$

If the model contains an intercept term, it is referred to as a "with intercept" model; otherwise, it is referred to as a "no intercept" model. If our situation necessitates the absence of an intercept in the model, we should employ a "no intercept" model. If, on the other hand, we get some

response even after setting the predictor variables to zero, we should use a "with intercept" model.

Estimation of Parameters

In the above models the variables y and x are known, these are observed. The only unknown quantities are the parameters β's. The main concern in regression analysis is how precisely we can estimate these parameters. Once these parameters are estimated, we have a model that we can use for further analysis. The least squares method is commonly used to estimate these parameters. For each observation (x_i, y_i),the method of least squares considers the error of each observation, i.e., for a simple model $\epsilon_i = \boldsymbol{y_i} - \boldsymbol{\beta_0}\text{-}\boldsymbol{\beta_1 x_i}$. The method of least squares requires the sum of the n squared errors. This criterion is denoted by *Q*:

$$Q = \sum_{i=1}^{n} (\boldsymbol{y_i} - \boldsymbol{\beta_0} - \boldsymbol{\beta_1 x_i})^2$$

According to the method of least squares, the estimators of $\boldsymbol{\beta_0}$ and $\boldsymbol{\beta_1}$ are those values 0 ˆ β and 1 ˆ β , respectively, that minimize the criterion *Q* for the given observations.

Departures of regression model:

Generally following departures may happen with a linear regression model.

(i) The linearity of regression function.

(ii) The normal distribution of error terms.

(iii) The constancy of error variance.

(iv) The independency of error terms.

(v) Presence of one or a few outlier observations.

(vi) Presence of multicollinearity.

The linearity of regression function.

The Teraesvirta Neural Network Test can be used to determine whether a linear regression function is appropriate for the data being analysed. If the assumption of linearity is not satisfied, modelling that data exceeds the capabilities of a conventional linear regression model. In such

a case, a non-linear regression model could be used. However, these models make some assumptions that are not always possible to identify when dealing with real-world data. As a result, in the field of regression analysis, machine learning approaches have gained a lot of traction in modelling non-linear dynamics.

Normality of Errors

Small departures from normality do not create any serious problems. Major departures, on the other hand, should be of concern. There are many tests available for testing normality of errors.

i) (Visual Method) Create a histogram

ii) (Visual Method) Create a Q-Q plot.

iii) (Formal Statistical Test) Perform a Shapiro-Wilk Test.

iv) (Formal Statistics Test) Perform a Kolmogorov-Smirnov Test.

Nonparametric regression could be employed when this assumption violated.

The constancy of error variance

One of the fundamental assumptions of linear regression is that the residuals have equal variance at each level of the predictor variable. This is referred to as homoscedasticity. When this assumption is broken, we say the residuals have heteroscedasticity. When this happens, the regression results become untrustworthy. A Breusch-Pagan test can be used to formally test for heteroscedasticity. One solution is to use weighted least squares (WLS) regression, which assigns weights to the observations so that those with small error variances are given more weight because they contain more information than those with larger error variances.

The independency of error terms

The residuals should be uncorrelated, which is one of the fundamental assumptions of linear regression. The Durbin Watson test is widely used to detect correlated errors. When there is a certain degree of correlation between the residuals in a regression model, one solution is to use the generalised least squares (GLS) technique to estimate the unknown parameters in a linear regression model. Ordinary least squares and weighted least squares can be statistically inefficient or even provide misleading results in these cases.

Presence of one or a few outlier observations

Outliers are observations for which the inputs are reasonable but the response is abnormally large or small in comparison to other cases with similar inputs. Extreme cases are those in which the input value is, in some cases, significantly different from the rest of the data. These are also known as high leverage points. In either case, these observations may have a significant impact on the least squares estimates. Detecting these observations is therefore critical. There are many tests available for testing outliers.

i) Grubbs's test

ii) Dixon's test

iii) Rosner's test

There are two solutions to over this problem

a) Remove the extreme observations and employ linear regression

b) **Quantile regression**: In contrast to regular linear regression, which uses the least squares method to calculate the conditional mean of the target across different feature values, quantile regression estimates the conditional median of the target. Quantile regression is a linear regression extension that is used when the linear regression conditions are not met (i.e., linearity, homoscedasticity, presence of extreme observations independence, or normality).

Presence of multicollinearity:

The occurrence of significant correlations between two or more independent variables in a multiple regression analysis is known as multicollinearity. When researchers or analysts seek to determine the effectiveness of each independent variable in predicting or explaining the dependent variable within a statistical model, multicollinearity can result in distorted or misleading outcomes. Generally, multicollinearity can cause broader confidence intervals and less dependable probabilities regarding the influence of independent variables in a model.

There are various techniques available for identifying multicollinearity. The most widely used approach is the VIF (Variable Inflation Factors). VIF assesses the strength of the correlation among independent variables. It is derived by regressing one variable against all other variables, or the VIF score of an independent variable indicates how well that variable is

accounted for by the other independent variables.The R^2 value is used to determine how well one independent variable is described by the other independent variables. A high R^2 value indicates that the variable is highly correlated with the others. This is captured by the VIF, as shown below:

$$VIF = \frac{1}{1-R^2}$$

- - The VIF begins at 1 and can increase indefinitely.
- - A VIF of 1 suggests that there is no correlation between this independent variable and the other variables.
- - A VIF value greater than 5 suggests significant multicollinearity exists between this independent variable and the others.

Principal Component regression:

Principal Components Regression (PCR) is a technique for analysing multicollinear regression data. When there is multicollinearity, least squares estimates are unbiased, but their variances are large, so they may be off by a large amount. Rather than directly regressing the dependent variable on the explanatory variables, the principal components of the explanatory variables are used as regressors in PCR. PCR is a type of regularised procedure as well as a type of shrinkage estimator because it typically uses only a subset of all the principal components for regression. It is hoped that this will result in more reliable estimates.

Ridge Regression

Ridge Regression is a method for analysing multiple regression data that are multicollinear. When there is multicollinearity, least squares estimates are unbiased, but their variances are large, so they may be off by a large amount. Ridge regression reduces standard errors by adding a degree of bias to the regression estimates. It is hoped that the net effect will result in more reliable estimates.

Quantile Regression

Standard linear regression techniques summarize the average relationship between a set of repressors and the outcome variable on the conditional mean function E(y|x). This provides only a partial view of the relationship, as we might be interested in describing the relationship at different points in the conditional distribution of y. Quantile regression provides that capability. Analogous to the conditional mean function of linear regression, we may consider the relationship between the regressors and outcome using the conditional median function Qq(y|x), where the median is the 50th percentile, or quantile q, of the empirical distribution. The quantile q ∈ (0, 1) is that y which splits the data into proportions q below and 1 − q above: F(yq) = q and yq = F −1 (q): for the median, q = 0.5.

- In QR, we minimize the absolute sum of errors:

$$\sum|\varepsilon_i| = \sum|y_i - \beta_0 - \beta_1 x_i|$$

- That is, we find those values of the parameters that minimize the sum of absolute errors.

$$\sum|\varepsilon_i| = \sum|y_i - \beta_0 - \beta_1 x_i|$$

$$\sum|\varepsilon_i| = \sum|y_i - \hat{y}_i|$$

3. Conclusion:

In conclusion, quantile regression offers a valuable alternative to conventional regression methods by providing a more comprehensive and flexible approach to modeling relationships between variables. Its ability to estimate conditional quantiles allows for a deeper understanding of the heterogeneity and variations in the relationship across the distribution of the dependent variable. By capturing the effects at different quantiles, quantile regression accommodates skewed or non-normal data, handles outliers robustly, and provides a more complete picture of the relationship. The advantages of quantile regression extend beyond descriptive analysis, offering practical benefits in policy evaluation, risk analysis, and decision-making. Its robustness to outliers and ability to assess effects at different quantiles make it well-suited for identifying vulnerable groups, evaluating the impact of interventions, and analyzing the effects of policies under different scenarios. Moreover, quantile regression offers a comprehensive set of inference tools that enable researchers to explore conditional effects and detect potential heterogeneity in the relationship between variables. By employing quantile regression, researchers and analysts are able to obtain a better understanding of intricate

datasets, reveal concealed patterns, and make more educated choices in a range of domains, including economics, finance, social sciences, and healthcare.As a versatile and powerful statistical technique, quantile regression opens up new avenues for understanding the dynamics of relationships and provides a valuable tool for addressing the challenges posed by real-world data.

R Codes

```
# Load the quantreg package
library(quantreg)

# Generate sample data
set.seed(123)
x <- rnorm(100)
y <- 2*x + rnorm(100)

# Fit quantile regression at the 0.5 quantile (median)
model <- rq(y ~ x, tau = 0.5)

# Print the model summary
summary(model)

# Predict the median values
new_x <- seq(min(x), max(x), length.out = 100)
pred <- predict(model, newdata = data.frame(x = new_x), type = "quantile")

# Plot the data and the fitted quantile regression line
plot(x, y, main = "Quantile Regression", xlab = "x", ylab = "y")
lines(new_x, pred, col = "red", lwd = 2)
```

Suggested Readings

Draper, N. R. and Smith, H. (1998). *Applied Regression Analysis*, New York: Wiley Eastern Ltd.

Montgomery, D. C., Peck, E. and Vining, G. (2003). *Introduction to Linear Regression Analysis*, 3rd Edition, New York: John Wiley and Sons.

भाकृअनुप ICAR

राअस्ट्रैप्रसं NIASM

Twenty-One Day Online Training Program on

Advanced Statistical and Machine Learning Techniques for Data Analysis Using Open-Source Software for Abiotic Stress Management in Agriculture

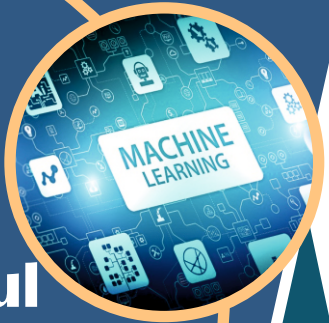
16 July – 5 August 2025

Chief Patron

Dr. K Sammi Reddy, Director, ICAR-NIASM

Patron

Dr. Nitin P Kurade, Head, SSSPS, ICAR-NIASM



Course Directors

Dr. Santosha Rathod
Dr. Nobin Chandra Paul
Ms. Ponnaganti Navyasree
Mr. K Ravi Kumar

Blue Sky SOFTWARE

Google Earth Engine

Organised by:

School of Social Science and Policy Support
ICAR-National Institute of Abiotic Stress Management Baramati, Maharashtra - 413115

About ICAR-NIASM

ICAR-NIASM is the premier institute of ICAR established in 2009 at Baramati. The institute aims at exploring the avenues for the management of abiotic stresses affecting the very sustainability of national food production systems. Besides focusing on developing climate resilient solutions through cutting-edge technologies for managing abiotic stresses, NIASM also aims to enhance scientific capacity through multidisciplinary research and capacity building programs.

About the Training Program

This 21-day online training program offers hands-on experience in advanced statistical, machine learning, and deep learning techniques for analyzing agricultural data. The training is not limited to abiotic stress management; it is applicable across all research disciplines where data analysis plays a critical role. Participants will work with large datasets using open-source tools such as R, Python, QGIS, VassarStats, and BlueSky Statistics through practical, application-oriented sessions.

Key Objectives

The training program aims to:

- Train the participants in multivariate statistics, AI- ML, and deep learning, agroecological modeling tools, remote sensing &GIS
- Provide hands-on experience with open-source software
- Enable independent application of these techniques in research work

Course Content

The program combines theoretical foundations with hands-on practical sessions, enabling participants to apply these techniques to their own datasets efficiently.

Module 1: Software Tools for Data Analysis

- Pre-training session on Installation guide to R/Python/other tools
- Introduction to R
- Introduction to Python
- Introduction to Bluesky Statistics & VassarStats
- Data Visualization in R
- R Shiny and R packages

Module 2: Regression & Multivariate Statistical Methods

- Regression Analysis
- Regression for Categorical Data
- Nonlinear Growth Models
- Regularization Techniques in Regression Models
- Panel Data Regression
- Non-Parametric Analysis
- Data Classificatory Techniques (CA, DA)
- Data Reduction Techniques (FA, PCA)

Module 3: Design of Experiments & Statistical Genetics

- Analysis of Complete and Incomplete Block Designs
- Analysis of Incomplete Block Designs
- Analysis of Groups of Experiments (GOE)
- Response Surface Methodology
- Generation Mean Analysis
- Mating Designs
- Path Analysis
- Stability Analysis
- QTL Analysis
- Transcriptomic Analysis
- Genome Wide Association Studies (GWAS)
- Genomic Selection
- Selection Index
- Meta-QTL Analysis
- Meta-Genomics

Module 4: Machine Learning & Deep Learning Techniques

- Introduction to Machine Learning
- k-nearest neighbor (KNN)
- Artificial Neural Network
- Support Vector Machine
- CART and Decision Tree
- Random Forest Regression
- Extreme Learning Machine
- XGBoost
- Deep Learning: RNN, GRU, CNN, LSTM, Transformer DL
- ML Optimization Techniques
- Yield Forecasting using AI

Module 5: Time Series & Forecasting Methods

- Trend Analysis
- Time Series Analysis
- ARCH Family of Models
- Bayesian Forecasting Models
- Count Time Series Models
- Spatiotemporal Time Series Modelling
- Hybrid Modelling
- Ensemble Modelling
- VAR and Cointegration Analysis

Module 6: Spatial & Environmental Data Analysis

- Introduction to RS & GIS
- Introduction to QGIS
- Introduction to Google Earth Engine
- Spatial Interpolation Techniques
- Introduction to Sampling & Spatial Sampling Strategy
- Application of ML in RS & GIS (ASIS portal)
- Application of UAVs in agricultural data modelling

Module 7: Agro-Ecological Modelling

- Biomass Modelling &Carbon Sequestration using Allometric Models
- CMIP6 GCM Models
- Crop Simulation Modelling (DSSAT and APSIM)
- High-throughput Plant Phenotyping
- Assessment of Extreme Weathers

Module 8: Emerging & Interdisciplinary Topics

- Importance of Data Science in Agricultural Research
- Meta Analysis
- AHP and Grey Model: Technology Forecasting
- Markov Chain Analysis
- Social Network Analysis
- Bibliometric Analysis
- Economic Index Development
- Impact Assessment Modelling, Trend Impact Analysis
- Statistical Modelling in Disease Epidemiology
- Fuzzy Regression Analysis

Expected Learning Outcomes

- By completing this program, participants will master in advanced statistical, machine learning, and deep learning techniques to analyze complex agricultural and environmental datasets.
- They will gain hands-on experience with open-source tools (R, Python, QGIS and others) for data analysis, image processing, and designing efficient workflows to address real-world abiotic stress challenges.

Who Can Apply?

- Researchers and scientists from agriculture, climate science, environmental studies, and allied fields
- Data analysts looking for transition from classical statistics to ML/DL approaches
- Academicians and students seeking proficiency in open-source statistical and geospatial tools

Registration Fee

- ₹ 1000/- for students and research scholars
- ₹ 2000/- for scientists, researchers, faculty members, and working professionals from public organizations
- ₹ 5000/- for participants from private industries

Bank Account Details

Account Holder Name: ICAR UNIT-NIASM, Baramati
Account Number: 30862846914
Name of the Bank: State Bank of India
Branch Address: Afzalpurkar Building, Bhigwan Road,
Baramati, Maharashtra-413102
IFSC: SBIN0000321
UPI Code : icarniasmbmt@sbi

ICAR UNIT NIASM BARAMATI

SCAN & PAY

UPI ID: icarniasmbmt@sbi

Important Dates

- Last Date for Receipt of Applications: 30th June 2025
- Information to Selected Candidate: 2nd July 2025

Registration Link: https://forms.gle/5cMmTxnS19DvWoc48

Contact for Registration Related Queries

Dr. Santosha Rathod
Senior Scientist (Agricultural Statistics)
School of Social Science and Policy Support
ICAR-NIASM, Baramati
Mob: 9900912188

Dr. Nobin Chandra Paul
Scientist (Agricultural Statistics)
School of Social Science and Policy Support
ICAR-NIASM, Baramati
Mob: 8851954194

Ms. Navyasree Ponnaganti
Scientist (ABM)
School of Social Science and Policy Support
ICAR-NIASM, Baramati
Mob: 8639110291

Mr. K. Ravi Kumar
Scientist (Agricultural Extension)
School of Social Science and Policy Support
ICAR-NIASM, Baramati
Mob: 9133120921

Contact Email Id:ssspsniasm@gmail.com

भाकृअनुप
ICAR

हर कदम, हर डगर
किसानों का हमसफर
भारतीय कृषि अनुसंधान परिषद

Agrisearch with a human touch

Application Form

Twenty-One Day Online Training Program

on

Advanced Statistical and Machine Learning Techniques for Data Analysis Using Open-Source Software for Abiotic Stress Management in Agriculture

16 July to5 August 2025

1.	Full Name (in BLOCK letters)				
2.	Highest degree with specialization				
3.	Present Institute Name				
4.	Address for Correspondence				
5.	E-mail address: *Telephone Number Mob/O/R:*				
6.	Date of Birth				
7.	Sex (Male/Female/other)				
8.	Education Qualification:				
	Degree	Subject	Year of passing	Class / Division / Equivalent	University / Institute
	Bachelors Masters Ph.D. Any Other				
9.	Level of Knowledge in Statistics			Beginner / Intermediate / Expert	
10.	Level of Knowledge in R/ Python/ other software			Beginner/Expert	
11.	Area of ongoing research work				
13	Expectations from the training				

**Candidate must fill in all the details*

Signature of the Applicant with date

CERTIFICATE

It is certified that information furnished above is correct.

Signature of the Recommending Authority
/ Head of the Department/ Institute along with Seal


भाकृअनुप
ICAR
हर कदम, हर डगर
किसानों का हमसफर
भारतीय कृषि अनुसंधान परिषद
Agrisearch with a human touch