Step by Step guide to Learn R

Venkat Reddy

Contents

- Step-1: Basics of R
- Step-2: Data Manipulations
- Step-3: Functions, Graphs and Analytics

Step-1: Basics of R

Step-1: Basics of R ;Contents

- What is R
- R Studio
- R Environment
- R Basics operations
- R packages
- R Vectors and Data frames
- R Scripts and Saving the work
- My First R Program
- R Functions
- R- Help

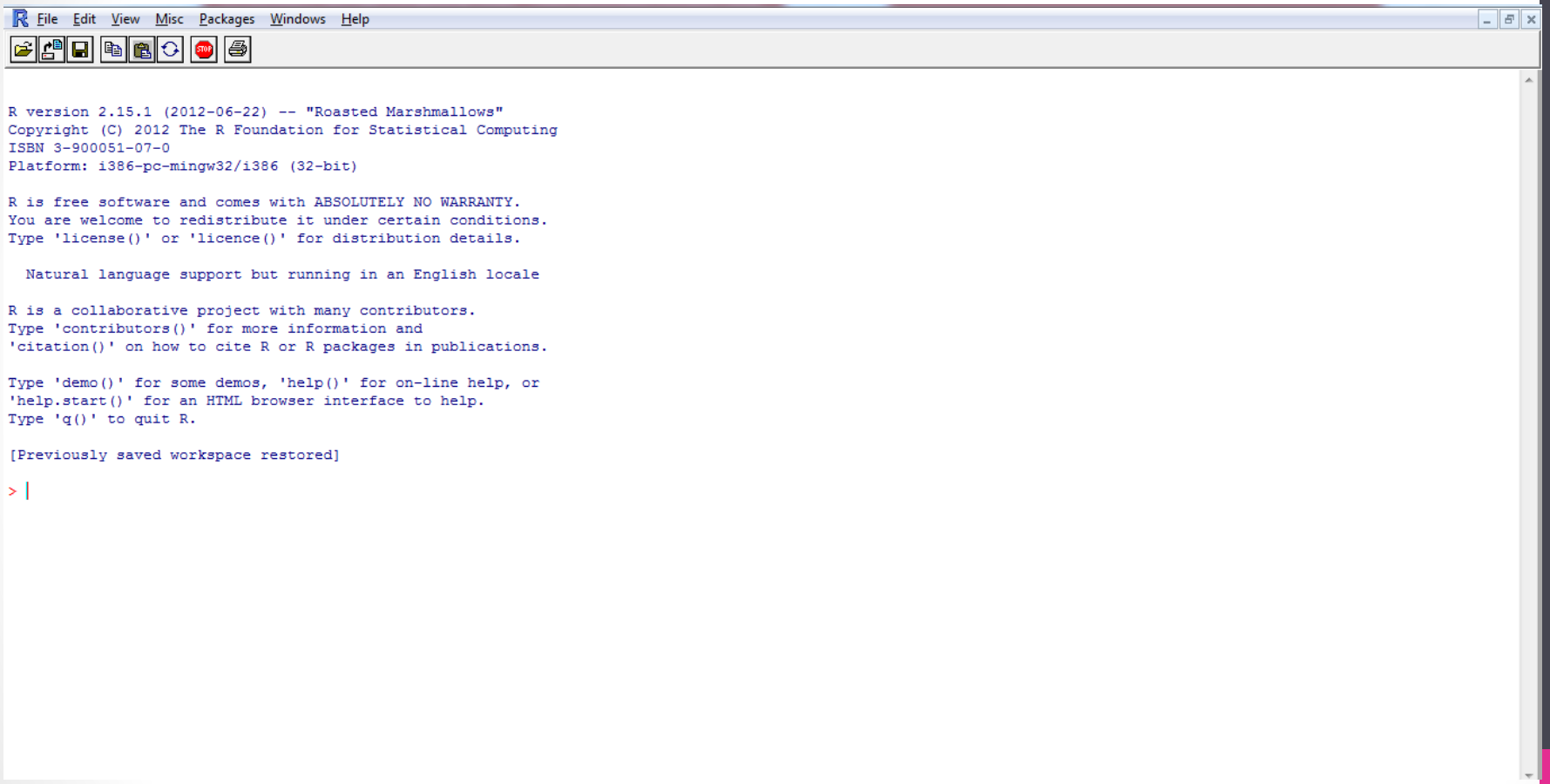
R

- Programming "environment"
- Runs on a variety of platforms including Windows, Unix and MacOS.
- Provides an unparalleled platform for programming new statistical methods in an easy and straightforward manner.
- Object-oriented
- Open source
- Excellent graphics capabilities
- Supported by a large user network

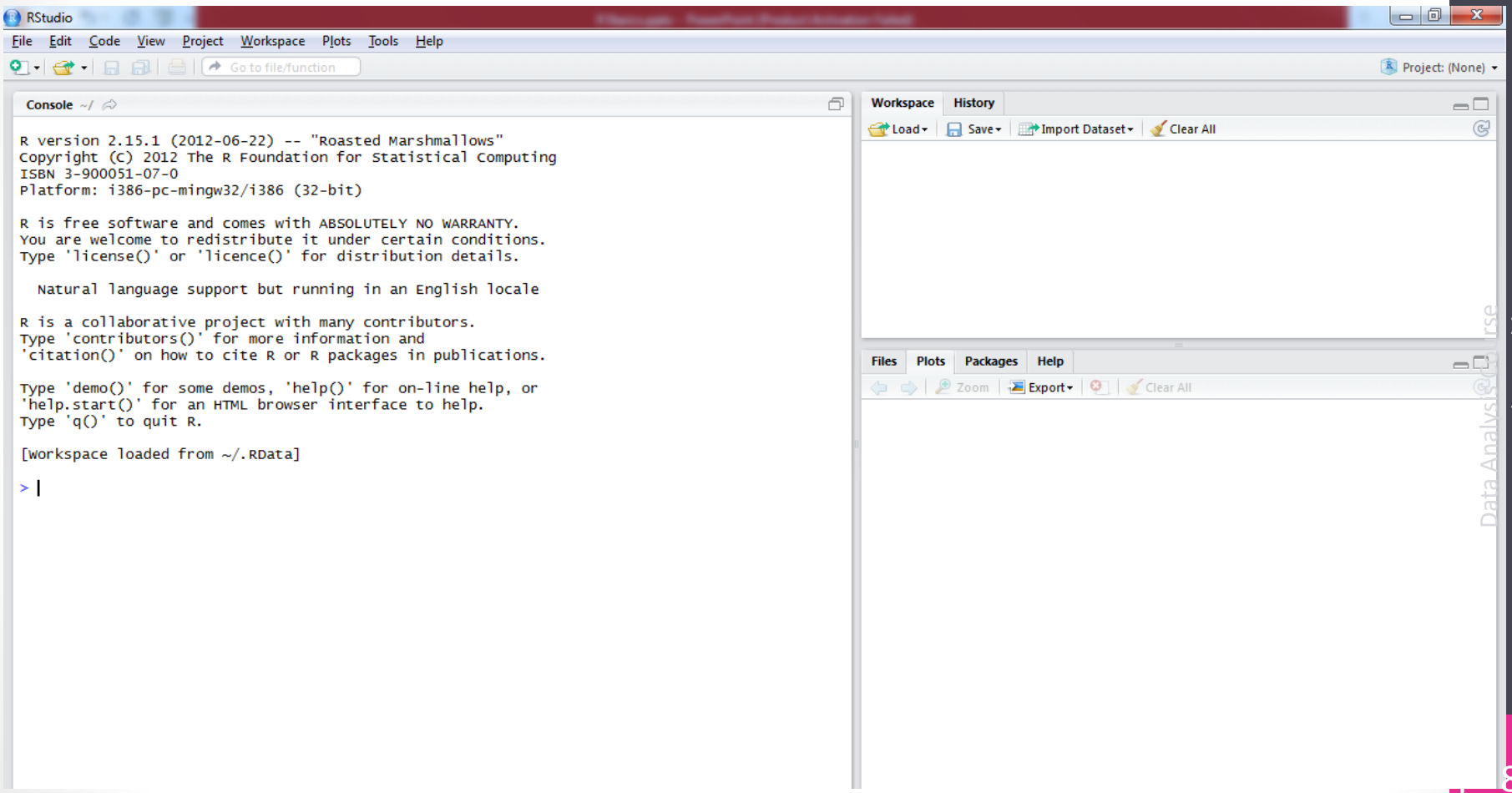
Download R

- Google it using R or CRAN (Comprehensive R Archive Network)
- http://www.r-project.org

R

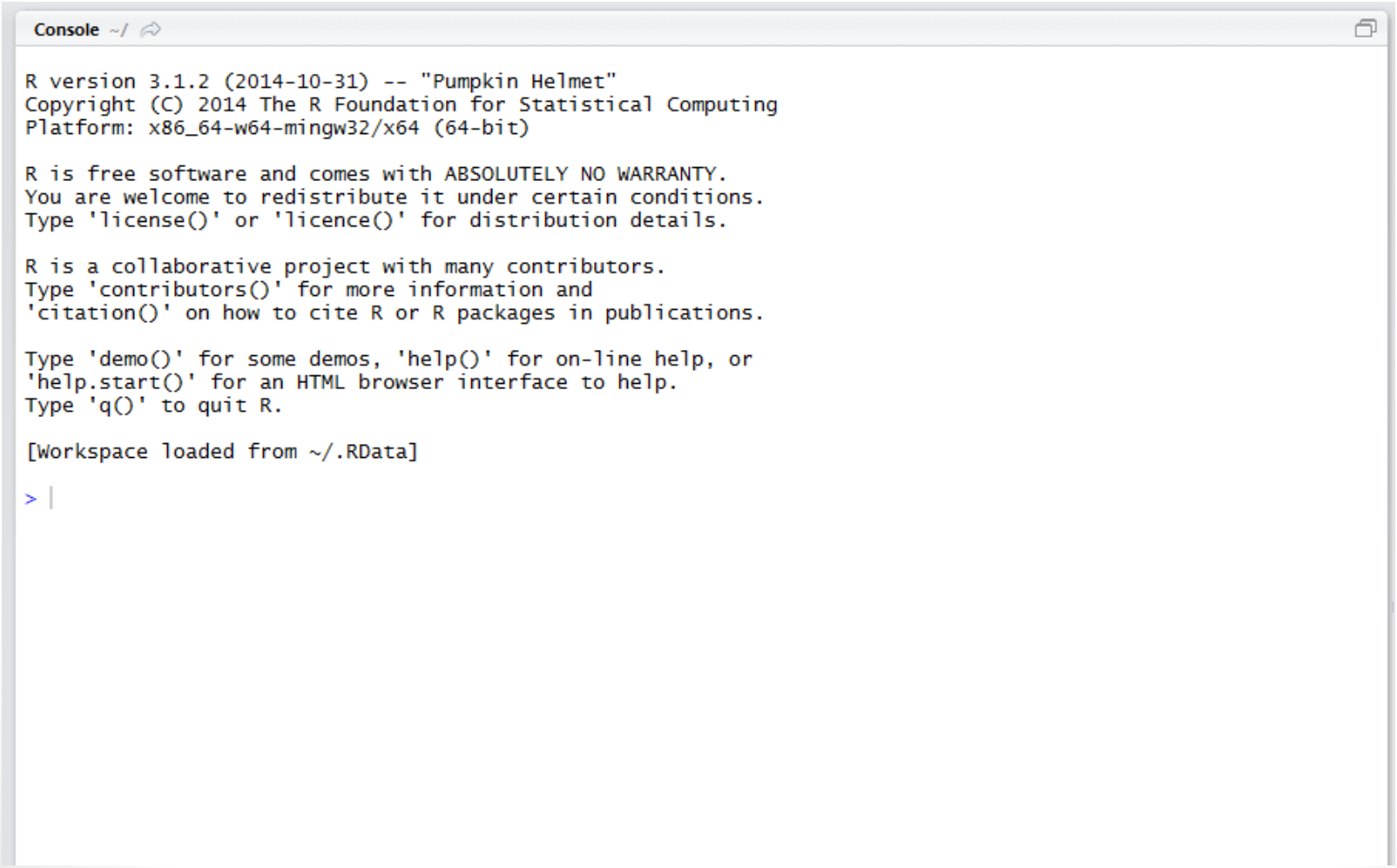

File Edit View Misc Packages Windows Help
R version 2.15.1 (2012-06-22) -- "Roasted Marshmallows"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-pc-mingw32/i386 (32-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Previously saved workspace restored]
>

R Studio



Data Analysis Course
Venkat Reddy

Console


Console ~/
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
>

R-Demo

- 2+2
- log(10)
- exp(5)
- help(log)
- demo(graphics) # pretty pictures...
- summary(airquality)

Workspace

- during an R session, all objects are stored in a temporary, working memory
- Commands are entered interactively at the **R** user prompt. **Up** and **down arrow keys** scroll through your command history.
- list objects `ls()`
- remove objects `rm()`
- `data()`

R-Basics :Naming convention

- must start with a letter (A-Z or a-z)
- can contain letters, digits (0-9), and/or periods "."
- R is a case sensitive language.
  - `mydata` different from `MyData`

R-Basics :Assignment

- "<-" used to indicate assignment
  - `X<-7`
  - `x<-c(1,2,3,4,5,6,7)`
  - `x<-c(1:7)`
  - `x<-1:4`

- Assignment to an object is denoted by "<-" or "->" or "=".
- If you see a notation "= =", you'll looking at a comparison operator.
  - Many other notations can be found from the documentation for the Base package or R.

Lab: Working with R

- x <- rnorm(10,mean=20,sd=5) # simulate data
- x
- mean(x)
- m <- mean(x)
- m
- log(m)
- x- m
- (x - m)^2
- sum((x - m)^2)
- `data()`
- Ukgas
- ls()

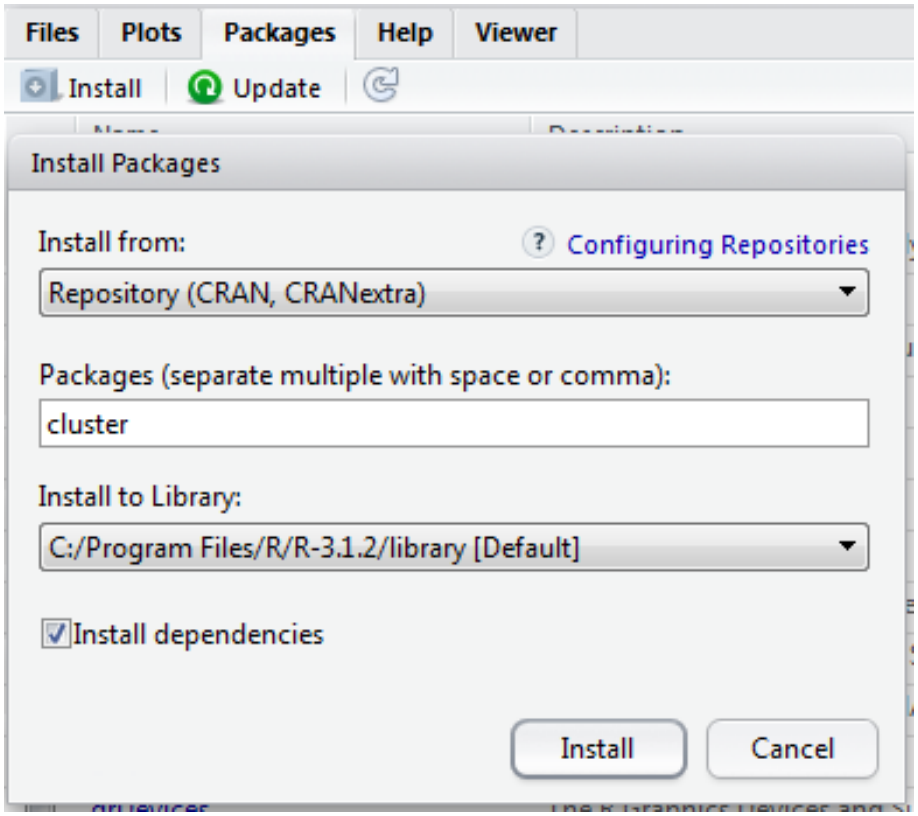
R Packages

- R consists of a core and packages. Packages contain functions that are not available in the core.
- Collections of R functions, data, and compiled code
- Well-defined format that ensures easy installation, a basic standard of documentation, and enhances portability and reliability
- When you download R, already a number (around 30) of packages are downloaded as well.
- You can use the function search to see a list of packages that are currently attached to the system, this list is also called the search path.
- search()

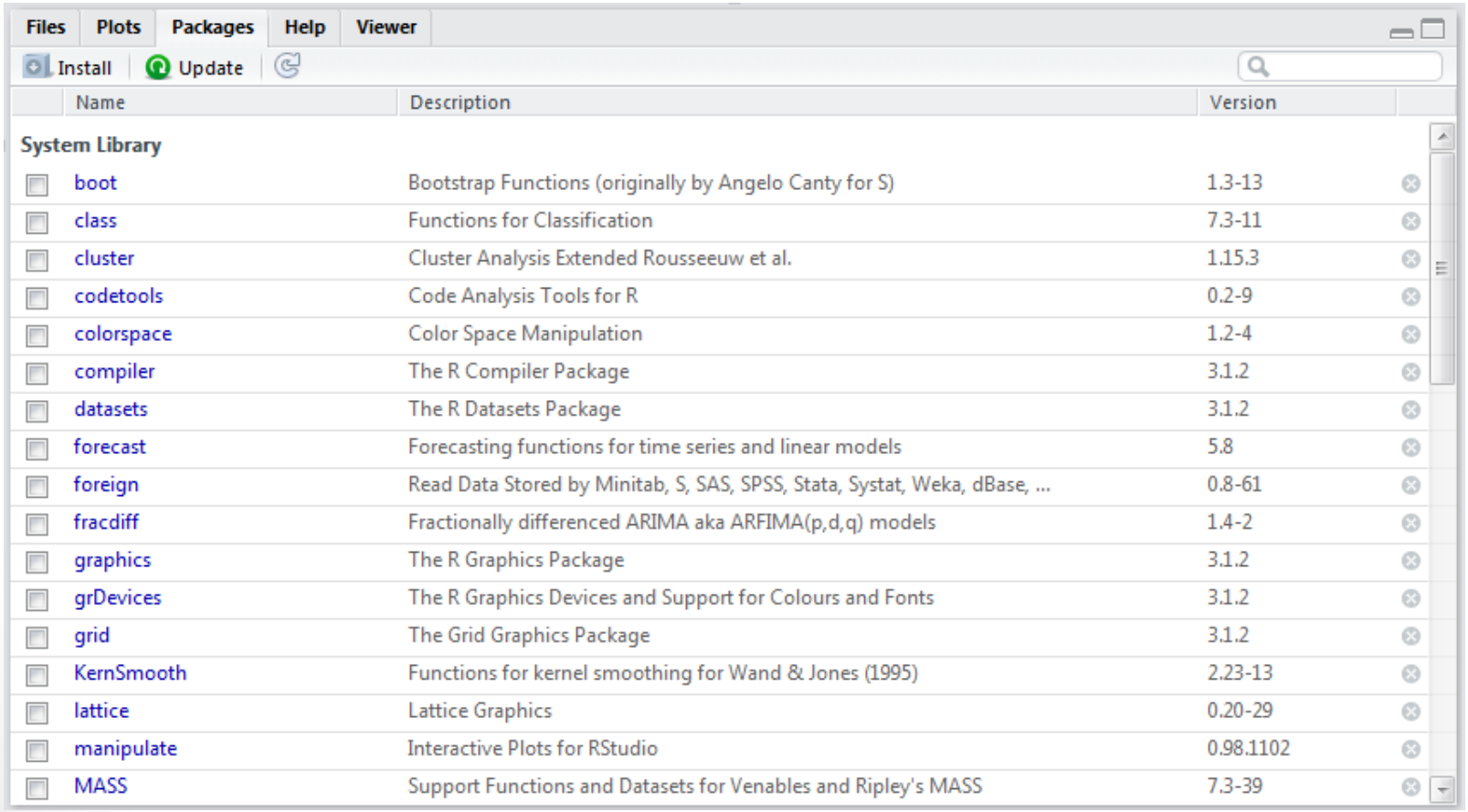
R packages

- Select the `Packages' menu and select `Install package...', a list of available packages on your system will be displayed.
- Select one and click `OK', the package is now attached to your current R session. Via the library function
- The library can also be used to list all the available libraries on your system with a short description. Run the function without any arguments

Download & Install Package



Load a package



R Packages

To load data

- RODBC, RMySQL, RPostgresSQL, RSQLite -
- XLConnect, xlsx - Read and write Micorsoft Excel files from R.
- foreign - Read a SAS data set into R Or an SPSS data set

To visualize data

- ggplot2 ggplot2 lets you use the grammar of graphics to build layered, customizable plots.
- ggvis - Interactive, web based graphics built with the grammar of graphics.
- rgl - Interactive 3D visualizations with R
- htmlwidgets - A fast way to build interactive (javascript based) visualizations with R.

R Packages

Predictive Modeling

- car - car's Anova function is popular for making type II and type III Anova tables.
- mgcv - Generalized Additive Models
- lme4/nlme - Linear and Non-linear mixed effects models
- randomForest - Random forest methods from machine learning
- multcomp - Tools for multiple comparison testing
- vcd - Visualization tools and tests for categorical data
- glmnet - Lasso and elastic-net regression methods with cross validation
- survival - Tools for survival analysis
- caret - Tools for training regression and classification models

Lab

- Download datasets package
- Download and attach forecast package
- Download and attach cluster package
- Install plyr package (for string operations)

R Data types

- Vectors
  - Basic R Type.
- Data Frames
  - Collection of vectors. (Datasets)
- Lists
  - Collection of R objects. (Documents)
- Other type
  - Matrix
  - Factor
  - Array

R vectors

- The basic data structure in R is the vector.
- Vectors are the simplest R objects, an ordered list of primitive R objects of a given type (e.g. real numbers, strings and logical).
- Vectors are indexed by integers starting at 1
- You can create a vector using the c() function which concatenates some elements.

```
name<-"Venkat"
is.vector(name)
Age<-29
is.vector(Age)
```

R Vectors

c() is a concatenate operator

```
Age <- c(15, 17, 16, 15, 16)
Marks1<- c(90, 86, 70, 88, 45)
Marks2<- c(85, 80, 74, 39, 65)
Name<- c("John", "Bob", "Kevin",
"Smith", "Rick")
class(Age)
is.vector(Age)
class(Marks)
is.vector(Marks1)
class(Name)
is.vector(Name)
```

R Vectors

- Most mathematical functions and operators can be applied to vectors(Without loops!)

```
Age+2
Marks1<-Marks1+10
Marks1<80
Marks1+Marks2
Total<-Marks1+Marks2
Total
Age/Total
Four_mult<-seq(0,40, by=4)
```

Accessing Vector Elements

- Use the [] operator to select elements
- To select specific elements:
  - Use index or vector of indexes to identify them
- To exclude specific elements:
  - Negate index or vector of indexes

```
Age
Age[2:5]
Age[-2]
Age[-2:-4]
Age[3]
Age[3]<-19
Age
```

R Data frames

- Collection of related vectors
- Most of the time, when data is Imported from externa ne
- Very Important feature in R

Untitled1* | students

	Name	Age	Marks1	Marks2
1	John	15	90	85
2	Bob	17	86	80
3	Kevin	16	70	74
4	Smith	15	88	39
5	Rick	16	45	65

```
students<-data.frame(Name, Age, Marks1, Marks2)
students
Profile_data<- data.frame(Name)
Profile_data<- data.frame(Name, Age)
students1<-c(Name, Age, Marks1, Marks2 )
students1
str(students)
?str()
```

Data

students 5 obs. of 4 variables

Values

Accessing R Data Frames

- Accessing a row or a Coolum or an element in the data frame

```
students$Name
students$Marks1
students$Marks2

students["Marks2"]
students["Name"]

students[1,]
students[,1]
students[,2:4]
students[,-1]
students[-1,]
```

Difference in Accessed Data frame elements

Three different ways of accessing may not produce same type of results

```
x<-students$Name
y<-students["Name"]
z<-students[,1]
x
y
z
str(x)
str(y)
str(z)
```

Lists

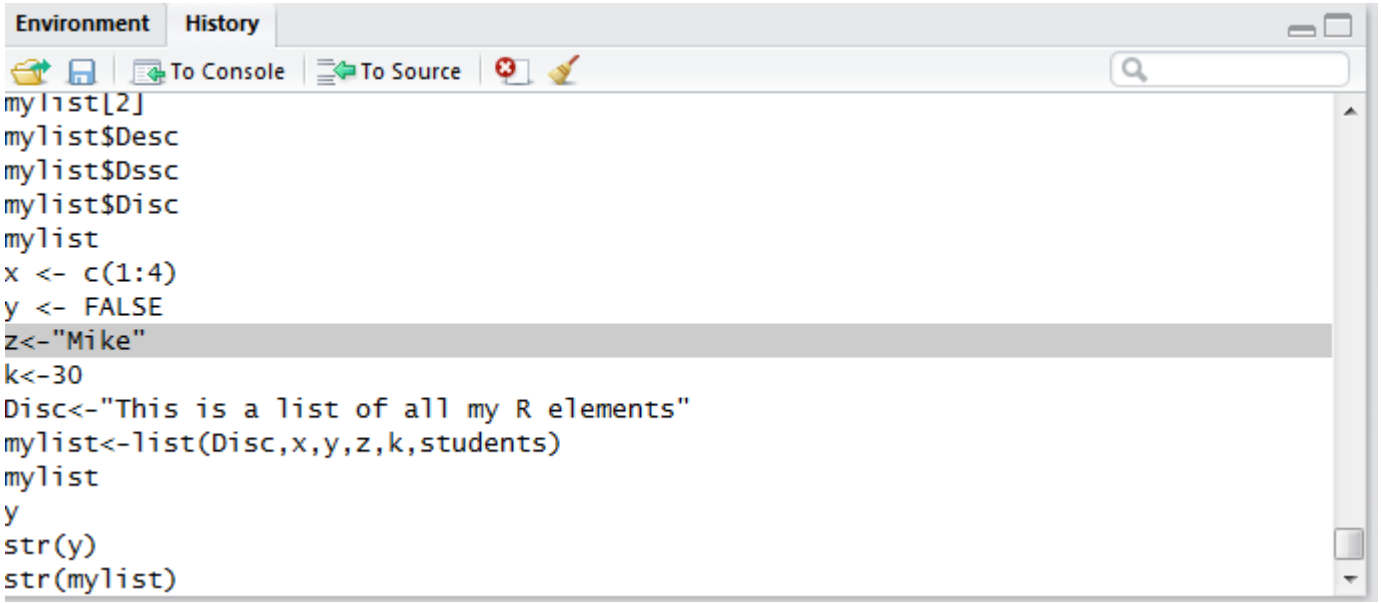
- A list is a collection of R objects / components
- A list allows you to gather a variety of (possibly unrelated) objects under one name.
- list() creates a list.
- The objects in a list **need not** have to be of the same type or length.

```
x <- c(1:20)
y <- FALSE
z<-"Mike"
k<-30
l<-students
Disc<-"This is a list of all my R elements"
str(x)
str(y)
str(z)
str(k)
str(l)
mylist<-list(Disc,x,y,z,k,l)
```

Accessing Lists

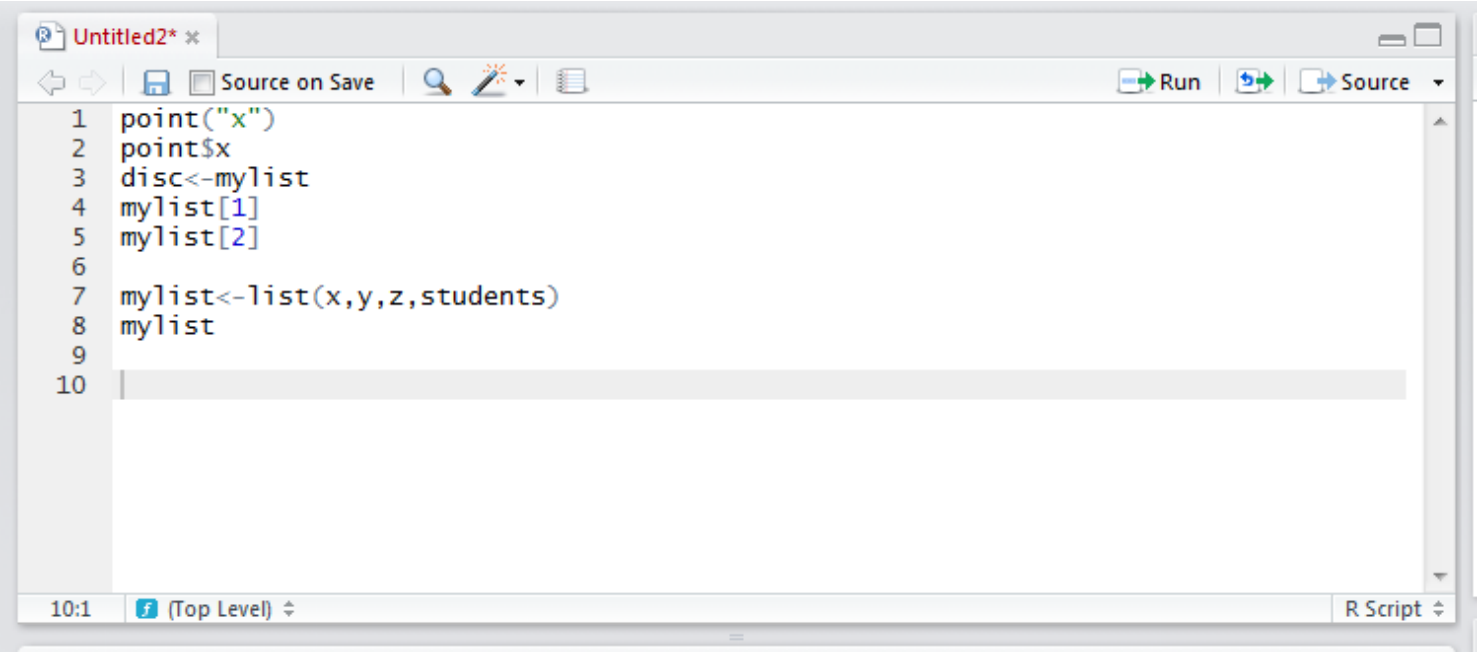
- `str(mylist)`
- `mylist`
- `mylist[1]`
- `mylist[2]`

R History



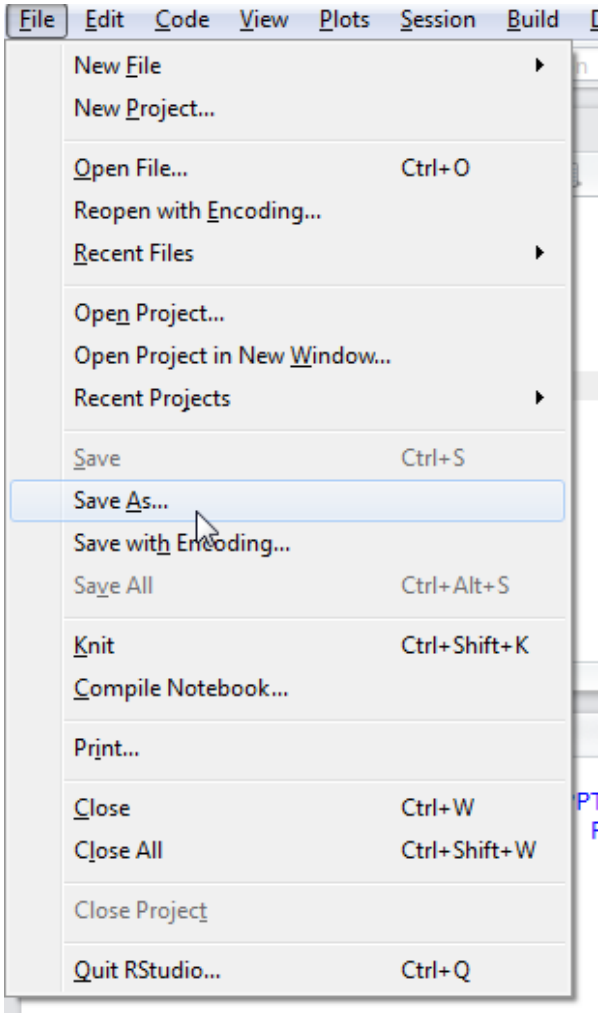
- Helps in accessing previously executed commands
- User can send the selected history to either console or to source

R Source file and Scripts

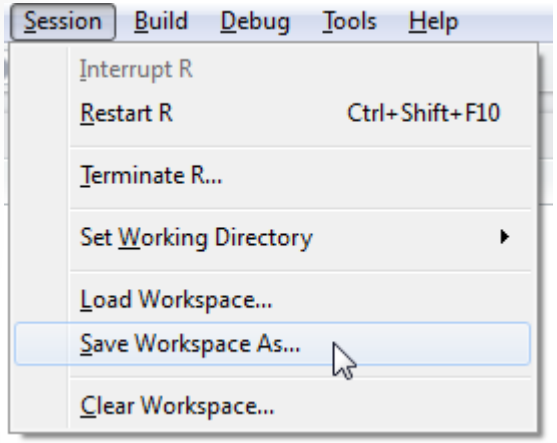


- R script or code file
- Can be used to re execute the stored codes
- Hit Ctrl+enter to execute the commands
- Save R script files for future use.

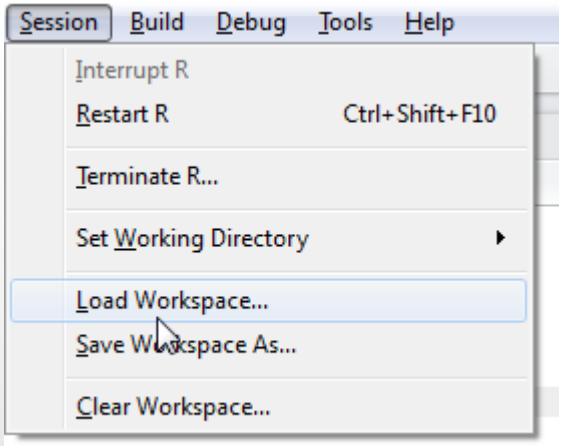
Saving R Script



Saving & Loading R Work Image



Saves all the R objects, including lists, arrays, data frames



Saves all the R objects, including lists, arrays, data frames. Loads the previous working image

LAB- My First R program

- Create income data(vector) for 4 employees with the values 5500, 6700, 8970, 5634
- Create a new variable tax and save 0.2 in it
- Create a new variable year and save 2015 in it
- Create a new variable company and save hp in it
- Derive net_income by deducting tax from the income
- Create Employee name(vector) for 4 employees with the values Redd, Kenn, Finn, Scott
- Create a data frame with Employee name and Net income
- Create a new list with all the above information on company, year, tax, Employee name and Salary dataset

My First R Program

- `Income<- c(5500, 6700, 8970, 5634)`
- `Tax<-0.2`
- `Year<-2015`
- `Company<-"hp"`
- `Net_income<- Income*(1-Tax)`
- `Emp_name<-c("Redd", "Kenn", "Finn", "Scott")`
- `Emp_database<-data.frame(Net_income, Emp_name)`
- `Emp_db_list<-list(Income,Tax, Year, Company, Emp_database)`

R- Functions

Numeric Functions

Function	Description
abs(*x*)	absolute value
sqrt(*x*)	square root
ceiling(*x*)	ceiling(3.475) is 4
floor(*x*)	floor(3.475) is 3
trunc(*x*)	trunc(5.99) is 5
round(*x*, digits=*n*)	round(3.475, digits=2) is 3.48
signif(*x*, digits=*n*)	signif(3.475, digits=2) is 3.5
cos(*x*), sin(*x*), tan(*x*)	also acos(*x*), cosh(*x*), acosh(*x*), etc.
log(*x*)	natural logarithm
log10(*x*)	common logarithm
exp(*x*)	e^*x*

Demo: Numeric Functions

- y<-abs(-20)
- x<-Sum(y+5)
- Z<-Log(x)
- round(Z,1)

Character Functions

Function	Description
substr(*x***, start=***n1***, stop=***n2***)**	Extract or replace substrings in a character vector. x <- "abcdef" substr(x, 2, 4) is "bcd" substr(x, 2, 4) <- "22222" is "a222ef"
grep(*pattern***,** *x* **,** **ignore.case**=FALSE**,** **fixed**=FALSE**)**	Search for *pattern* in *x*. If fixed =FALSE then *pattern* is a regular expression. If fixed=TRUE then *pattern* is a text string. Returns matching indices. grep("A", c("b","A","c"), fixed=TRUE) returns 2
sub(*pattern***,** *replacement***,** *x***,** **ignore.case** =FALSE**,** **fixed**=FALSE**)**	Find *pattern* in *x* and replace with *replacement* text. If fixed=FALSE then *pattern* is a regular expression. If fixed = T then *pattern* is a text string. sub("\\s",".","Hello There") returns "Hello.There"
strsplit(*x***,** *split***)**	Split the elements of character vector *x* at *split*. strsplit("abc", "") returns 3 element vector "a","b","c"
paste(..., sep="")	Concatenate strings after using *sep* string to seperate them. paste("x",1:3,sep="") returns c("x1","x2" "x3") paste("x",1:3,sep="M") returns c("xM1","xM2" "xM3") paste("Today is", date())
toupper(*x***)**	Uppercase

Demo :Character Functions

- cust_id<-"Cust1233416"
- id<-substr(cust_id, 5,10)
- Up=toupper(cust_id)

R-Help

- If you encounter a new command during the exercises, and you'd like to know what it does, please consult the documentation. All R commands are listed nowhere, and the only way to get to know new commands is to read the documentation files, so we'd like you to practise this youself.
- **Tutorials**

  Each of the following tutorials are in PDF format.

  - P. Kuhnert & B. Venables, An Introduction to R: Software for Statistical Modeling & Computing
  - J.H. Maindonald, Using R for Data Analysis and Graphics
  - B. Muenchen, R for SAS and SPSS Users
  - W.J. Owen, The R Guide
  - D. Rossiter, Introduction to the R Project for Statistical Computing for Use at the ITC
  - W.N. Venebles & D. M. Smith, An Introduction to R

R-Tutorials

- Paul Geissler's excellent R tutorial
- Dave Robert's Excellent Labs on Ecological Analysis
- Excellent Tutorials by David Rossitier
- Excellent tutorial an nearly every aspect of R (c/o Rob Kabacoff) **MOST of these notes follow this web page format**
- Introduction to R by Vincent Zoonekynd
- R Cookbook
- Data Manipulation Reference

Step-2:Data Handling in R

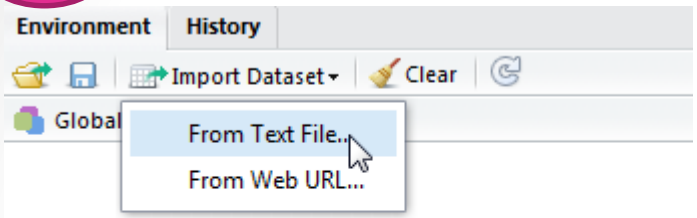
Step-2:Data Handling in R; Contents

- Data imploring from external files
  - csv
  - txt
  - SAS
  - excel
- Working with Datasets
- Creating new variables in R
- Data manipulations in R
- Sorting in R & Removing Duplicates
- Exporting the R datasets into external files
- Data Merging

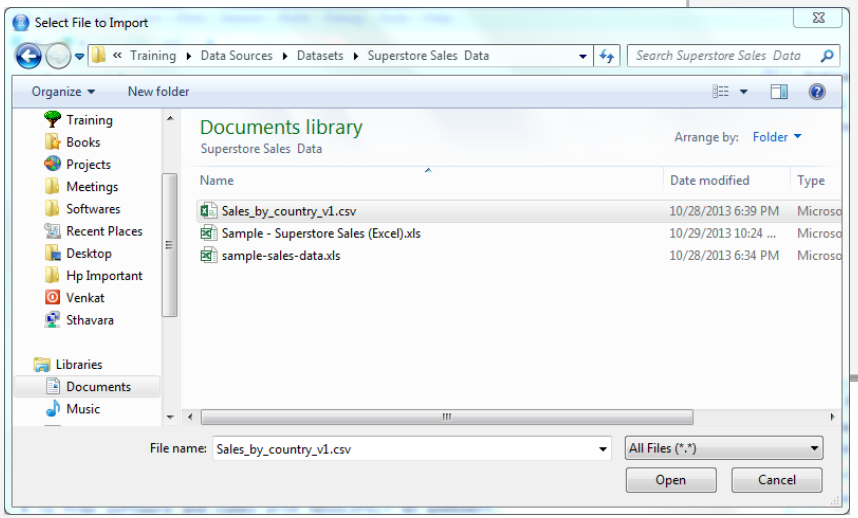
Importing data

Data Importing using GUI option

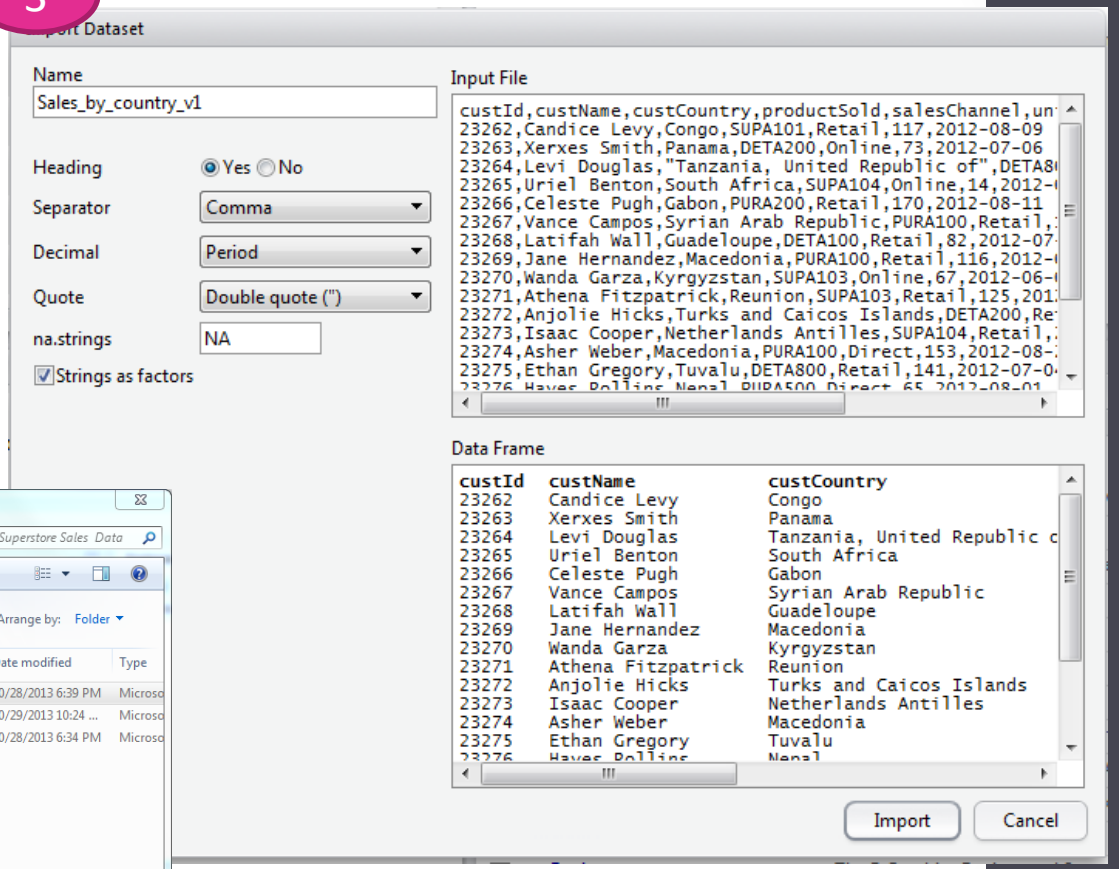
1



2



3



CSV file Importing using R Script

```
Sales_by_country_v1 <- read.csv("Datasets/Superstore Sales Data/Sales_by_country_v1.csv")
```

```
<<R File Name>> <- read.csv("<<Full Path with Extension>>")
```

Lab - Importing

- Import cars data csv file using GUI
  - Use the code to import health care data
- Import Survey data.txt using GUI
  - Is there any change in the code
  - Use the code to import client_manager.txt

Importing SAS files

- Need a new package sas7bdat

```
install.packages("sas7bdat")
library(sas7bdat)
gnpdata <-
read.sas7bdat("C:\\Users\\venk\\Documents\\Training\\Data
Sources\\Datasets\\SAS Datasets\\gnp.sas7bdat")
View(gnpdata)
```

Lab

- Import GNP data from SAS sample datasets
- Import Demographics data

Importing Excel File

Need a customized package

```
install.packages("xlsx")
library(xlsx)
dataset <- read.xlsx("c:/myexcel.xlsx", sheetName =
"mysheet")
```

Many ways to Import from Excel

- ODBC connection - Works only for 32 bit windows
- xlsReadWrite package – Doesn’t support xlsx files
- XLConnect package – Needs Java
- xlsx package –Needs Java
- A self made function
  - source("https://gist.github.com/schaunwheeler/5825002/raw/3526a15b032c06392740e20b6c9a179add2cee49/xlsxToR.r")
  - xlsxToR = **function**("myfile.xlsx", header = **TRUE**)

Connecting to ODBC

- library(RODBC)

```
conn <-odbcConnect("dblink", uid="uid",
pwd="passwd")
salesdata <- sqlFetch(conn, "sales")
ordersdat<- sqlQuery(conn, "select * from
orders")
close(conn)
```

Working with Datasets

Viewing the data and meta info

- Show all R objects
  - ls()
- Show Column names
  - names(gnpdata)
- Total Number of rows and coloumns
  - dim(demo) # Number of rows and columns
- Initial few observations
  - head(gnpdata)
  - head(demographics, n=20)
- Last few observations
  - tail(gnpdata)
  - tail(gnpdata, n=10)
- Complete structure of the data
  - str(gnpdata) # Structure of the data
- Levels in a categorical variable
  - levels(demographics$region)

Printing the data and summary

- Complete structure of the variables in the data
  - str(demographics$region) # Type of the variable
  - str(demographics$pop)
- Printing the data
  - demographics #prints the demographics data
- In view window
  - View(demographics)
- A quick Summary
  - summary(demographics)
- Different Type of summary
  - describe(gnpdata) # install.packages("Hmisc"), if missing
- Remove the dataset
  - New<-head(gnpdata)
  - rm(New)

Lab: Printing the data and meta info

- Import “Petrol Consumption by City.csv” data
- How many rows and columns are there in this dataset?
- Print only column names in the dataset
- Print first 10 observations
- Print the last 5 observations
- Get the summary of the dataset
- Print the structure of the data
- Describe the field petrol consumption
- Create a new dataset by taking first 30 observations from this data
- Print the resultant data
- Remove the new dataset

Manually Editing Data in R

- New_data<-edit(gnpdata)
- head(New_data)

Sub setting the data

- New dataset with selected rows
  - Petrol1<-Petrol[5:20,]
  - Petrol2<- Petrol[c(5,10,20),]
- New dataset by keeping selected columns
  - Petrol3<- Petrol[, 2:4]
  - Petrol4<- Petrol[, c(1,3,5)]
  - Petrol4<- Petrol[, c("City_Index", "Consum_mill_gallons")]
- New dataset with selected rows and columns
  - Petrol5<-Petrol[5:20, c(1,3,5)]
- New dataset with selected rows and excluding columns
  - Petrol6<-Petrol[5:20, c(-3,-5,-6)]

Lab: Sub setting the data

- Import Market_data_one.csv
- Create a new dataset for each of the below filters
  - Select first 40 market campaigns only
  - Select name, start date and end date only
  - Select 700 to 1000 observations along with four variables id, name, budget and reach
  - Select 5000 to 6000 observations drop budget and reach fields

Sub setting the data with variable filter conditions

- Selection with a condition on variables
  - For example, selection of complains where budget is greater than $5000.
- Need to use subset function in R
  - `newdata <- subset(old_data,condition1 | condition2)`
- And condition & filters
  - `market1<-subset(market, budget>5000 & num_assets>10)`
- OR condition & filters
  - `market2<-subset(market, budget>10000 | num_assets>10)`

Sub setting the data with variable filter conditions

- AND, OR condition Numeric and Character filters
  - `market3<-subset(market, (budget>10000 | num_assets>10) & vertical=="Business/Finance" )`
- AND, OR condition , Numeric and Character filters & selected fields
  - `market4<-subset(market, (budget>10000 | num_assets>10) & vertical=="Business/Finance", select=c (name , start_date, end_date, vertical ))`

Lab: Sub setting the data with variable filter conditions

- Import Cars.csv data
- Create a new dataset for exclusively Audi cars
  - `Audi<-subset(Cars, Make=="Audi")`
- Create a new dataset for all cars with Horsepower>300 and more than 6 Cylinders.
- Create a new dataset by taking only SUV cars. Keep only four variables(Make, Model, Type and MSRP) in the final dataset.
- Create a new dataset by taking AUDI, BMW or Ford company makes. Keep only four variables(Make, Model, Type and MSRP) in the final dataset.

Creating Calculated variable in R

Attaching and Detaching the data

- demographics$NAME to print the NAME variable
- Direct variable name doesn't work here.
  - `FemaleSchoolpct`
  - `Error: object 'FemaleSchoolpct' not found`
- Attach to access the variables directly
  - attach(demographics)
  - FemaleSchoolpct
- Don't forget to detach once you are done with the operations
  - detach(demographics)

Calculated Fields in R

- Use the assignment operator <- to create new variables.
  - dataset$sum <- dataset$x1 + dataset$x2
  - dataset$mean <- (dataset$x1 + dataset$x2)/2
  - attach(dataset)
  - dataset$sum <- x1 + x2
  - dataset$mean <- (x1 + x2)/2
  - detach(dataset)

Demo-Calculated Fields in R

- `Cars$HW_Ratio<-Cars$Horsepower/Cars$Weight`
- `attach(Cars)`
- `Length_new<-Length*(1.2)`
- `detach(Cars)`

New variable creation doesn't work without dataset name

- `attach(Cars)`
- `Cars$Length_new<-Length*(1.2)`
- `detach(Cars)`

Calculated Fields using if then else

- Its like excel if then else loop
- Newvar<-ifelse(Condition, True Value, False Value)
- ```
  online_sales$price_ind<-
  ifelse((listPrice>10000), "High", "Low")
  ```
  ```

LAB-Calculated Fields in R

- Import AMSProductSales.csv data
- Create a new variables in the data set based on below conditions
  - GDP_new by taking 90% of Real GDP
  - GDP rate by taking ratio of GDP and Population..Total
  - Calculate the unemployed population by multiplying population and Unemployment.Rate (be careful with %)
- Create a new variable “target”
  - If the sales is less than 700,000MM Then “Missed”
  - Between 700,000 to 900,000 MM Then “Reached”
  - More than 900,000 MM then “Exceeded”

LAB-Calculated Fields in R

- Import market_data_one
- Create name_new by taking first 20 characters of the name variable
- Create a new variable by converting the name of the campaign into uppercase
- Create a flag variable that takes value 1 If campaign starting month is not equal to campaign ending month, else 0

Sorting and duplicates records

Sorting the data

- Newdata<-olddata[order(variables),]
- Its ascending by default
  - `online_sales_sort<-`
    `online_sales[order(online_sales$listPrice),]`
- Use –ve sign before the variable for descending
  - `online_sales_sort1<-`
  - `online_sales[order(-online_sales$listPrice),]`
- Sorting based on multiple variables
  - `online_sales_sort2<-online_sales[order(-`
    `online_sales$listPrice,`
    `online_sales$avRating ),]`

LAB: Sorting the data

- Import Bank Customer Attrition Data.csv data
- Sort the data based on Age of the customer
- Sort age descending and Survey_OverallSatisfactionpercent descending

Identifying Duplicates

- Identifying duplicates Using duplicated() function. These are overall record level duplicates

```
Dupes_in_bill<-duplicated(Bill)
summary(Dupes_in_bill)
Bill[Dupes_in_bill,]
Dupes_in_complaints<-duplicated(Complaints)
summary(Dupes_in_complaints)
Complaints[Dupes_in_complaints,]
```

- Duplicates based on Key

```
Dupes_in_bill_key<-duplicated(Bill$DEL_NO)
summary(Dupes_in_bill_key)
Bill[Dupes_in_bill_key,]
Dupes_in_complaints_key<-duplicated(Complaints$DEL_NO)
summary(Dupes_in_complaints_key)
Complaints[Dupes_in_complaints_key,]
```

Removing Duplicates

- Remove overall Duplicates
  - Use unique function. It will simply consider the unique records
  - `unique(Bill)`
  - `unique(Complaints)`
- Removing duplicates based on variables. Add ! Sign to print the unique values

```
Dupes_in_bill_key<-duplicated(Bill$DEL_NO)
summary(Dupes_in_bill_key)
Bill[!Dupes_in_bill_key,]
dim(Bill[!Dupes_in_bill_key,])
dim(Bill)
```

LAB: Removing Duplicates

- Import Bill and Complaints datasets from telecom data
- Identify overall duplicates in Bill data
- Identify overall duplicates in complaints data
- Remove duplicates based on DEL_NO from Bill data and save all the unique records in the first set in bill_unique dataset
- Remove duplicates based on DEL_NO from Complaints data and Save all the unique records in the second set in complaints_unique dataset

- TV commercial data
  - Orders
    - Overall & Key
  - Slots
    - Overall & Key

Exporting the data Out of R

Exporting data

- To a Tab delimited text File
  - write.table(longley, "C:\\Users\\venk\\Documents\\Economic Data.txt", sep="\t")
- To a CSV file
  - write.csv(longley, "C:\\Users\\venk\\Documents\\Economic Data.csv")
- To an Excel Spreadsheet
  - library(xlsReadWrite)
  - write.xls(dataset, "c:/dataset.xls")
- To SAS
  - library(foreign)
  - write.foreign(dataset, "c:/dataset.sas", package="SAS")

Demo: Exporting data

- write.table(sales_data, "C:\\Users\\VENKAT\\Google Drive\\Training\\R\\Data\\sales_export.txt", sep="\t")
- write.table(sales_data, "C:\\Users\\VENKAT\\Google Drive\\Training\\R\\Data\\sales_export.csv", sep=",")

Data sets merging and Joining

Merging Syntax

- With a single primary key
  - Newdata <- merge(dataone, datatwo ,by="primary_key")
- With composite keys
  - Newdata <- merge(dataone, datatwo ,by=c("primary_key1", "primary_key2")
- Its inner join by default.
- Joins
  - Inner Join
    - Newdata <- merge(dataone, datatwo ,by="primary_key", all=FALSE)
  - Outer Join
    - Newdata <- merge(dataone, datatwo ,by="primary_key", all=TRUE)
  - Left Outer Join
    - Newdata <- merge(dataone, datatwo ,by="primary_key", all.x=TRUE)
  - Right Outer Join
    - Newdata <- merge(dataone, datatwo ,by="primary_key", all.y=TRUE)

Demo Joins

- Orders <- read.csv("~/Training/Data Sources/Datasets/TV Commercial Slots Analysis/Orders.csv")
- slots <- read.csv("~/Training/Data Sources/Datasets/TV Commercial Slots Analysis/slots.csv")
- Union
  - Newdata1 <- merge(Orders, slots, by=c("ISCI.AD.iD","Date","Time"), all=TRUE)
- Intersection
  - Newdata2 <- merge(Orders, slots, by=c("ISCI.AD.iD","Date","Time"), all=FALSE)
- All orders data
  - Newdata3 <- merge(Orders, slots, by=c("ISCI.AD.iD","Date","Time"), all.x=TRUE)
- All Slots data
  - Newdata4 <- merge(Orders, slots, by=c("ISCI.AD.iD","Date","Time"), all.y=TRUE)

LAB: Data Joins

- Import Telecom bill & complaints data
- Remove duplicates based on DEL_NO
- Create a dataset for each of these requirements
  - All the customers who either have bill data or complaints data available
  - All the customers who appear in both bill data and complaints data
  - Customers who have bill data along with their complaints, if any
  - Customers who have Complaints data along with their bill info, if any

Step-3: Graphs, Reporting and Analytics on R

Step-3: Graphs, Reporting and Analytics on R; Contents

- User Defined Functions
- Descriptive Statistics
- Graphs
- Analytics using R

User Defined Functions

For Loop in R

- Finding Squares of first twenty numbers

```
square<-1 # Any random value
for (i in 1 : 20 )
{
square[i]<-i^2
}
```

- Adding cumulative column to Air travel data

```
Air_travel$Cumulative[1]= Air_travel$AIR[1]
for (i in 2:nrow(Air_travel))
{
Air_travel$Cumulative[i]= Air_travel$Cumulative[i-1]+
Air_travel$AIR[i]
}
```

Lab For Loop

- Add cumulative claim amount in health claim data using for loop
- Create a delta variable in air travel data. Where delta is the difference between current month and previous month's AIR travel number

Distance Calculation function

UserFunction <- function(arg1, arg2, ...){

statements

return(object)

}

Distance Function

```
mydistance<-function(x1,y1,x2,y2)
{
sqrt((x1-x2)^2+(y1-y2)^2)
}
```

Lab: User Defined Functions

- Create a function that calculates the Absolute percentage difference between two input values. Take second value as reference
- Create a function that takes vector as input and gives the sum of squares of input vector values as output

```
mysumsquare<-function(x)
{
  sum=0
    for (j in 1 : length(x))
    {
      sum=sum+(x[j])^2
    }
    return (sum)
}
```

- `mysumsquare(c(1,2,3,4,5,6,7))`

Basic Descriptive Statistics

Descriptive Statistics

```
# Reading CSV file
healthClaim =
read.csv("C:/Users/trendwise/Desktop/project/wiley/Analyst/W
3Labs/Data/2.2 Health_claim.csv");
# Names of variables/fields in data
names(healthClaim)
# viewing data (top few lines)
head(healthClaim)
# frequency for month
table(healthClaim$Month)
```

Descriptive Statistics

```
# Summary Statistics
# viewing highest & lowest for claim_Amount
summary(healthClaim)
# Find the frequency of age variable
table(healthClaim$age)
```

Descriptive Statistics

```
# Create a distribution chart(bar chart)
hist(healthClaim$age)
# Is the distribution Bell shaped/right skewed/left skewed?
# 3- histograme for Num_medical_bills
table(healthClaim$Num_medical_bills)
hist(healthClaim$Num_medical_bills,freq=T)
```

Lab: Descriptive Statistics

- Import Credit risk data
- Find out the average monthly income
- Draw a histogram for age of customers
- How many customers have less than 3 loans?
  - Find the frequency of customers with different number of loans.

Creating Graphs

Creating Graphs on R

- **Creating a histogram**
- Download price web data
  - Price_web_data <- read.csv("C:/Users/VENKAT/Projects/Wiley Training/TTT/Final Docs/2.2 Module 2&3/Data/Price Data/Price_web_data.csv")
- Creating a histogram on price
  - hist(Price_web_data$listPrice, breaks=5, col="red")
- **Creating a bar chart for brand**
  - Create the counts table first
    - counts<-table(Price_web_data$brand)
    - barplot(counts)
    - barplot(counts, las=2)
  - Horizontal bard Chart
    - barplot(counts, las=2,horiz=TRUE)

Lab: Creating Graphs on R

- Import Market_data_one.csv into R
- Crete a histogram on reach
- Customize the histogram with 10 bins and blue color
- Crete a bar chart on number of campaigns by vertical
- Display all the labels
- Change the graph to horizontal bar chart
- Create a pie chart for number of campaigns by vertical
  - Pie(count)
- Import Petrol consumption data
  - Draw a scatter plot between Prop_pop_drivers_licenses and Consum_mill_gallons
  - Draw a scatter plot between Petrol_tax_cents_per_gallon and Consum_mill_gallons

Advanced Graph options

- 3D Graphs
- Lattice Graphs
- ggplot2 Graphs
- Mosaic Plots
- Correlograms

Analytics using R

Correlation in R

```
petrolData = read.csv("C:/Users/trendwise/Desktop/project/wiley/Analyst/W3Labs/W3S2/Data/3.1_Petrol Consuption.csv")
# view what type of data is
head(petrolData)
# summary of data
summary(petrolData)
# correlation bet diff variables
cor(data.frame(petrolData$Petrol_tax_cents_per_gallon,petrolData$Average_income_dollars,petrolData$Prop_pop_drivers_licenses,petrolData$Consum_mill_gallons))
plot(petrolData$Prop_pop_drivers_licenses,petrolData$Consum_mill_gallons,col="purple")
```

Simple Linear Regression

```
petrolReg =lm(petrolData$Consum_mill_gallons ~ petrolData$Prop_pop_drivers_licenses)
petrolReg
summary(petrolReg) # Summary
coef(petrolReg) # coefficients
resid(petrolReg) # Residuals
fitted(petrolReg) # predicted values
anova(petrolReg) # Analysis of Variance table
# Plotting result of regression
layout(matrix(1:4,2,2))
plot(petrolReg)
petrolReg2 = lm(petrolData$Consum_mill_gallons ~ petrolData$Petrol_tax_cents_per_gallon)
petrolReg2
```

Multiple Linear Regression

```
# Multiple Linear Regression on petrol data
petrolMultiReg =lm(petrolData$Consum_mill_gallons~
petrolData$Prop_pop_drivers_licenses +
petrolData$Petrol_tax_cents_per_gallon)
summary(petrolMultiReg)
petrolMultiReg2 =lm(petrolData$Consum_mill_gallons~
petrolData$Prop_pop_drivers_licenses +
petrolData$Petrol_tax_cents_per_gallon +
petrolData$Average_income_dollars)
petrolMultiReg2
summary(petrolMultiReg2)
```

Logistic Regression

```
head(loanData)
logRegModel = glm(loanData$SeriousDlqin2yrs ~ loanData$util +
loanData$age1 + loanData$DebtRatio1 +
loanData$MonthlyIncome1 + loanData$num_loans +
loanData$depend ,family=binomial())
summary(logRegModel)
predLoanData = predict.glm(logRegModel,type="response")
predLoanData[1:30]
```

Thank you