Code Portfolio

Aniket Naik Desai

June 9, 2019

## Data Selection

df[4] #print 4th item in the df

df[-4] #print All but 4th

df[2:4] #print two to four

df[-(2:4)] #print all elements except 2,3,4

df[c(1, 5)] #print 1 , 5

df[x == 5] #print the element which have value equal to integer 5, can also compare with string df[x < 6] #print the elements which have value less than integer 6

df[x %in% c(1, 2, 5)] #this is like SQL IN. #for two dimentional df, use “,” after row selection to allow all columns to be printed for the selected rows

## Sequence

a:b # creates a sequence from number a to number b

seq(2, 3, by=0.5) # creates a sequence from 2 to 3, increments by .5 (2 and 3 included)

rep(1:2, times=3) #Repeat the vector - 1 2 1 2 1 2

rep(1:2, each=3) #Repeats each element - 1 1 1 2 2 2

## Create a vector

c(2, 4, 6) #vector with content 2,4,6 c(1:5) #vector with content 1,2,3,4,5

names(df) = c(“first\_name”, “last\_name”) #assign names

df[“first\_name”] #print element with name “first\_name”

## Matrix

m <- matrix(x, nrow = 3, ncol = 3,dimnames = list(rownames, colnames)) #vector x is split by 3 rows and 3 columns, also need to give bycol=True or byrow=True. defaults to byrow # dimnames is the names of the row and cols that you just created # matrix can be manipulated with +,-,\*,/

t(m) # gives the transpose

cbind(m, n) #to combine 2 matixes columns i.e. horizontally expanding

rbind(m,n) #for rowwise

c(m) #will combine the matrix into a vector by the column

## Dataframe functions

head(df) # Top few rows

tail(df) # Last few rows

View(df) # Shows whole dataset in new window

nrow(df) # total number of rows

ncol(df) # total number of columns

dim(df) # total count of row and column

rownames(df) # SHows row names

colnames(df) # shows column names

names(df) # show column names

summary(df) # Statistical Summarization of dataset

str(df) # Statistical information with more details

## Working Directory

setwd(“C://file/path”) #set the working directory, the slash needs to be “/”. "" gives error

getwd() #find the location of the working directory

## Library and functions

install.packages(“dplyr”) #installs the library “dplyr”

Library(dplyr) # loads the library

?filter #tells details of filterfunction, use only after library is loaded

## User Function

fun\_name <- function(var)

{ print “hello”

return(new\_variable) }

# Allows to create new functions incase some code is repeatedly used

# var is the variable you want to pass to the code

## Loop

### For loop

for (var in sequence/vector/result\_set/list){

print “for”

}

# can be used to parse through a list created through another fuctions eg list of files in a directory

### While loop

while (condition if true){

print “while”

}

#condition is a logical comparison, some languages treat any integer>0 as true

## Sort functions -

sort(df$x) #sort ascending

rev(df$x) #reverse the sort

table(df$x) #create a tabular count by frequency of each element of column x

unique(df$x) #returns unique values

## Logical Flow

if (condition){

print “if”

}

else { print “else” }

#condition is a logical comparison, some languages treat any integer>0 as true #if works as a array operator and as object operator. you will get error if you pass array to the object operator one

## Logical operators

== #Equal to

!= #Not equal to

> #left greater than right

< #left less than right

>= #left greater or equal to right

<= #left less or equal to right

is.na(a) #will tell if NA values is found in a, works for object and array

is.null(a) #will tell if null values is found in a, works for object and array

# Additional ones is.factor(a),is.numeric(a),is.integer()

## Data Import

### Comma Delimited Files

read\_csv(“file.csv”)

### Semi-colon Delimited Files

read\_csv2(“file.csv”)

### Files with Any Delimiter

read\_delim(“file.txt”, delim = “|”) #The delimiter needs to be specified

### Fixed Width Files

read\_fwf(“file.fwf”, col\_positions = c(1, 3, 5)) #need to check the column locations before running command

### Tab Delimited Files

read\_tsv(“file.tsv”) # read\_table() can be used in some cases

### Load from R data file

load(“file.RData”) save(a, b, .., file = “myimage.RData”) save(df, file = “file.Rdata”)

Additional options

read\_csv(f, col\_names = c(“x”, “y”, “z”)) #read only some columns read\_csv(f, col\_names = FALSE) #If there is no header read\_csv(f, skip = x) #To skip a x lines from the top of file

## Export Data

### Comma delimited file

write\_csv(df, path, na = “NA”) #append = FALSE, col\_names = !append options in all to append new data to existing file #delim defaults to " " for write\_delim(), “,” for write\_excel\_csv() and “;” for write\_excel\_csv2()

### File with arbitrary delimiter

write\_delim(df, path, delim = " “, na =”NA")

### CSV for excel

write\_excel\_csv(df, path, na = “NA”)

### String to file

write\_file(df, path, append = FALSE)

### String vector to file, one element per line

write\_lines(df,path, na = “NA”, append = FALSE) #does not have col\_names = !append option

### Object to RDS file

write\_rds(df, path, compress = c(“none”, “gz”,“bz2”, “xz”), …)

### Tab delimited files

write\_tsv(df, path, na = “NA”, append = FALSE, col\_names = !append)

### Save to R data file

save(a, b, .., file = “myimage.RData”)

save(df, file = “file.Rdata”)

## Typecasting

as.logical #change to true and false

as.numeric #change to number

as.character #changes to string, needed if you want to save a number or factor as a string as.factor #change to a factor with levels, cannot specify anything else. sometimes better to use factor() function

## Useful functions

factor(x = df$x, levels = levels\_in\_df, labels = levels\_i\_want,exclude = NA, ordered = TRUE/FALSE, nmax = NA) #useful to use a complete option than to use defaults

rm(a, b, ..) #remove some objects

rm(list = ls()) #remove all objects

list.files() # List all files in WD

ls() # List all objects in workspace

log(x) #returns log to base e aka natural log

exp(x) # returns e^x

max(x)

min(x)

round(x,d) #returns x rounded to d decimal points

signif(x, d) #round x to n significant figures, signif(12346789, 3) gives 12300000

quantile(dep\_delay, 0.75) #75th percentile

cor(x, y) #correlation between x and y, check before you run the regression, or you can also use a scatterplot

sum(x) #will return the sum of the elements of x, better use is sum(is.na(x)) sum of all empty cells is vector x, used to find % data missing in each row

paste(x, y, sep = ’ ’) #Join multiple vectors together. # x <- c(“a”, “b”,“c”), y <- c(“p”, “q”,“r”) will give result “ap” “bq” “cr”

paste(x, collapse = ’ ’) #Join elements of a vector together. # x <- c(“a”, “b”,“c”) will give “a b c”

grep(pattern, x) #like GREP in Linux

gsub(pattern, new\_pattern, x) #replaces pattern with new\_pattern in x

toupper(x) #to uppercase

tolower(x) #to lowercase

nchar(x) # count of characters in x

## Plots

### Strip Charts -

stripchart(df$x) # basic plot along the x-axis

stripchart(w1$vals,method=“stack”, main=‘Leaf BioMass in High CO2 Environment’, xlab=‘BioMass of Leaves’)

# to identify repetition use stack, main is the title of the graph, xlab - x axis title

### Histograms -

hist(df$x, breaks = 20,xlim=c(0,10), col = “green”) #basic frequency plot with range divided into 20 blocks, color is set to green # xlim tells what is the range that is plotted

### Boxplots -

boxplot(df$x) # basic box plot

boxplot(df$x~df$y) # box plot x based on the classification povided through y abline(a,b) #plots a line on the graph with intercept and slope values # h= , v= will plot horizontal and vertical lines respectively through the intercept

### Scatter Plots -

plot(dfy) # x,y scatterplot, helpful in identifying the relationship between 2 variables

### Multiple Plots

par(mfrow = c(1, 2), mar = c(5, 4, 2, 1))

with(subset(pollution, region == “west”), plot(latitude, pm25, main = “West”)) with(subset(pollution, region == “east”), plot(latitude, pm25, main = “East”))

### Normal QQ Plots

qqnorm(df

x) # both functions used together to check if data is normally distributed

# Dplyr

### Select

sleepData <- select(msleep, name, sleep\_total) #df followed by var names

select(msleep, -name) # Select all other than name

select(msleep, name:order) # Select all from name till order

select(msleep, starts\_with(“sl”))

select(msleep,ends\_with(“ad”)) select(msleep,contains(“ad”)) select(msleep,matches(“one\_string|or\_the\_other”))#regex select(msleep,one\_of(c(“one”,“two”,“three”))) #from a group of names

### Group\_by

group\_by(flights, date) #df and variable #you can also use n = n() to find count of rows in each group

### Arrange

flights %>% arrange(desc(arr\_delay),.by\_group = TRUE)

#default ascending; need desc(variable) otherwise #by group is required else arrange will ignore grouping # NA is sorted to the end for local data even with desc() and depends on backend for remote

### Tally

flights %>% group\_by(carrier, flight, dest) %>% tally(sort = TRUE) %>% filter(n == 365) #Tally calls a n() or sum(n) it adds a column n() to the table. #Tally will collapse all the other columns

### Ranking functions

min\_rank(c(1, 1, 2, 3)) #Will do a tie for rank 1 then skip rank 2 and move to 3

dense\_rank(c(1, 1, 2, 3)) #Will do a tie for rank 1 then move to rank 2

row\_number(c(1, 1, 3, 2)) #No Tie, will print rank in asc order. For tie, first come first rank

### Mutate

per\_hour <- flights %>% filter(cancelled == 0) %>% mutate(time = hour + minute / 60) #calculates and adds a column with the result of the calculation

### Lead Lag

lag(1:10, 1) lead(1:10, 1) #To find values in a vector after a certain lead or lag

### Summarize

summarise(by\_day, dep = mean(dep\_delay, na.rm = TRUE), arr = mean(arr\_delay, na.rm = TRUE))#df and the aggregate fuctions, create var and assign aggregate value

#combination of filter and summarize with pipe

daily\_delay <- by\_day %>% filter(!is.na(dep\_delay)) %>% summarise( mean = mean(dep\_delay), median = median(dep\_delay), q75 = quantile(dep\_delay, 0.75), over\_15 = mean(dep\_delay > 15), over\_30 = mean(dep\_delay > 30), over\_60 = mean(dep\_delay > 60) )

#first the filter is applied and then summarized with different criteria

# Reshape

### Gather and Spread

gather(df,colname1,colname2,key=newcol,value=newcol\_values) #colname1 and 2 will be changed to values from col names and the values in those cols with be entered besides them.

Changes df from wide to long format.

spread(df,colname1,colname2) #colname1 and 2 will be changed from values to col names and the values besides those cols with be entered below them. Changes df from long to wide format.

### Split and combine cells -

separate(df, oldcol ,into = c(“newcol1”, “newcol2”)) #There is also option to give a separator but / is taken as separator by default

separate\_rows(df, oldcol) #instead of getting 2 cols, this will give 2 rows. Wide to long format

unite(df, oldcol1, oldcol2, col = “newcol”, sep = "“) #combine cols, what you put in”" for sep with be used to separate the values of the 2 cols in each cell

### NA manipulation -

drop\_na(df,colname) #drops the rows with NA in the specified column replace\_na(df,list(colname = newvalue)) #replace NA in the colname column with the newvalue fill(df,colname,direction=c(“down”,“up”)) #Fill in NA’s in colname with most recent non-NA value in the column. Direction down is default.

# GGPlot2 -

### Use color to set the color of choice by keeping colour outside aes

ggplot(data = mpg) + geom\_point(mapping = aes(x = displ, y = hwy), color = “blue”) # geom\_point tells that points need to be plotted

### Color

ggplot(data = mpg) + geom\_point(mapping = aes(x = displ, y = hwy, color = class))

### Size

ggplot(data = mpg) + geom\_point(mapping = aes(x = displ, y = hwy, size = class))

### Alpha

ggplot(data = mpg) + geom\_point(mapping = aes(x = displ, y = hwy, alpha = class)) #alpha controls transparency level

### Shape

ggplot(data = mpg) + geom\_point(mapping = aes(x = displ, y = hwy, shape = class))

### Facet

#break the plot with variable into individual plots in 2 rows ggplot(data = mpg) + geom\_point(mapping = aes(x = displ, y = hwy)) + facet\_wrap(~ class, nrow = 2)

#facet grid plots in a square with some plots empty ggplot(data = mpg) + geom\_point(mapping = aes(x = displ, y = hwy)) + facet\_grid(drv ~ cyl)

### Plot a line with ggplot2

ggplot(filter(per\_hour, n > 30), aes(time, arr\_delay)) + geom\_vline(xintercept = 5:24, colour = “white”, size = 2) + geom\_point() # Data is obtained through the filter in this case

## Clustering

### Kmeans Cluster

#### Libraries used for Kmeans Cluster

tidyverse # data manipulation

cluster # clustering algorithms

factoextra # clustering algorithms & visualization

#### Data preparation

df <- na.omit(df) #always omit missing data

df <- scale(df) # scale data to avoid dependency on variable unit

distance <- get\_dist(df) # gets distances for visualization

fviz\_dist(distance, gradient = list(low = “#00AFBB”, mid = “white”, high = “#FC4E07”)) # Plots the distance for visual inspection

k2 <- kmeans(df, centers = 2, nstart = 25) # 2 Clusters, 25 initial configurations str(k2) # Print the details of the cluster

fviz\_cluster(k2, data = df) # Visualize the cluster alongside the data

#### Optimal number of clusters

set.seed(123)

fviz\_nbclust(df, kmeans, method = “wss”) # Elbow method

fviz\_nbclust(df, kmeans, method = “silhouette”) # Average Siloutee Method

fviz\_nbclust(df, kmeans, method = “gap\_stat”, k.max = 10) # Gap Method # We can use k.max to specify the number of clusters that we want

Another way for Gap method gap\_stat <- clusGap(df, FUN = kmeans, nstart = 25, K.max = 10, B = 50) # Print the result print(gap\_stat, method = “firstmax”) fviz\_gap\_stat(gap\_stat) # visualize the result

### Hierarchical Cluster–

#### Libraries used for Hierarchical Cluster

tidyverse # data manipulation

cluster # clustering algorithms

factoextra # clustering algorithms & visualization

dendextend # for comparing two dendrograms

#### Data preparation

df <- na.omit(df) #always omit missing data

df <- scale(df) # scale data to avoid dependency on variable unit

### Agglomerative Hierarchical Clustering

#### Dissimilarity matrix

dist <- dist(df, method = “euclidean”) # Can also use Manhattan dist

#### Hierarchical clustering using Complete Linkage

hc1 <- hclust(d, method = “complete” ) # Method can be “complete”, “average”, “single”, “ward.D2”

#### Plot the obtained dendrogram

plot(hc1, cex = 0.6, hang = -1)

#### Compute with agnes

hc2 <- agnes(df, method = “complete”) #agnes gives the Agglomerative coefficient. 1 strong clustering, 0 weak clustering

#### Agglomerative coefficient

hc2$ac

### Divisive Hierarchical Clustering

#### compute divisive hierarchical clustering

hc4 <- diana(df)

#### Divise coefficient

hc4$dc # 1 strong clustering, 0 weak clustering

#### plot dendrogram

pltree(hc4, cex = 0.6, hang = -1, main = “Dendrogram of diana”)

#### Optimal number of clusters

set.seed(123) fviz\_nbclust(df, FUN = hcut, method = “wss”) # Elbow method fviz\_nbclust(df, FUN = hcut, method = “silhouette”) # Average Siloutee Method fviz\_nbclust(df, FUN = hcut, method = “gap\_stat”, k.max = 10) # Gap Method # We can use k.max to specify the number of clusters that we want

### Cuttree

#### Cut Dendrogram into 4 groups

sub\_grp <- cutree(hc4, k = 4)

plot with borders plot(hc4, cex = 0.6) rect.hclust(hc4, k = 4, border = 2:5)

fviz\_cluster(list(data = df, cluster = sub\_grp))

#### To use cutree with agnes and diana

**Cut agnes() tree into 4 groups**

agg <- agnes(df, method = “ward”) cutree(as.hclust(agg), k = 4)

**Cut diana() tree into 4 groups**

dvi <- diana(df) cutree(as.hclust(dvi), k = 4)