# ========================================

# Day 1, March 10, 2016

# Session 1 ==============================

# ======================================

# see what directory R is currently set to

# ======================================

getwd()

# ======================================

# change your "working directory"

# to a location of your choice

# ======================================

setwd('C:/MyGithub/CDCRworkshopCode')

# ======================================

# check location again

# ======================================

getwd()

# ======================================

# Typing in commands - do simple math

# ======================================

2 + 2

3 + (4\*8)

3+4\*8

6 + (8\*\*2) # exponents can be done using two

# asterix's \*\* or you can use

# the caret ^ symbol

12^2

# ======================================

pi # pi is a built in Constant

4 \* pi

# ======================================

help(Constants)

LETTERS

letters

month.abb

month.name

# ====================================

# create x which had 1 numeric value

# ======================================

x = 3 \* 5

x <- 3 \* 5 # notice nothing is shown in the console

# you have to type the name of the object

# to "see" it

x

# ======================================

# create y which is a numeric (integer) vector

# with 12 elements

# ======================================

y <- 1:12

y

# you'll notice that y is a vector of integers

# you can convert between numeric and integer type

# data classes using the as.numeric()

# and as.integer() functions.

ynum <- as.numeric(y)

yint <- as.integer(ynum)

# ======================================

# create z a numeric vector

# containing a sequence of numbers from

# 0 to 10 in units of 0.5

# ======================================

z <- seq(0,10,.5)

z

# ======================================

# the above code can also be done with the

# explict function arguments defined

# see the examples in help(seq)

# ======================================

help(seq)

z <- seq(from=0, to=10, by=0.5)

z

# ======================================

# create new object sinz which is a numeric

# vector now containing the sin (sine function)

# of the z values

# ======================================

sinz <- sin(z)

# ======================================

# look at objects created get simple listing

# and get listing with structure information

# ======================================

ls()

ls.str()

# ========================================

# while you can see information about each of these

# objects in the Global Environment (top right)

# of the RStudio window, the function length()

# is useful for determining how many elements are

# in a given vector/data object.

# ========================================

length(x)

length(y)

length(z)

length(sinz)

# ========================================

# other functions that are helpful for finding

# out about data objects is the str() "structure"

# function and the class() function.

# ====================================

# NOTE ON ADDING COMMENTS

# ====================================

# highlight all comment lines

# then hit CTRL+SHIFT+C to toggle back and forth

# between code and comments.

#'this is

#'a new line of commenting that begins with

#'a hashtag or number # sign followed by a single quote

#'then when you hit return the comment delimiter #'

#'is automatically added until you type in a

#'single hashtag without the single quote.

# then remove single quote and begin new line for r code

# ========================================

str(y)

class(y)

str(z)

class(z)

# ======================================

# create some vectors with different objects

#

# ======================================

a1 <- c(1,2,3,4,10,11)

a1

a2 <- c('a','g','f','r','t','s')

a2

a3 <- c(TRUE,FALSE,TRUE,TRUE,FALSE,TRUE)

a3

a4 <- c(1,2,'a','b',TRUE,FALSE)

a4

a5 <- c(1,2,3,4,TRUE,FALSE)

a5

a6 <- c(5,10,x)

# ======================================

# use the class() function to investigate these further

# ======================================

class(a1)

class(a2)

class(a3)

class(a4)

class(a5)

# ======================================

# using cbind() - this makes a matrix

# where every element has to be the same

# type - these are now all characters

# ======================================

a1to5cbind <- cbind(a1,a2,a3,a4,a5)

a1to5cbind

class(a1to5cbind)

str(a1to5cbind)

dim(a1to5cbind)

# ======================================

# using rbind() - like cbind()

# but now the vectors come in as rows

# ======================================

a1to5rbind <- rbind(a1,a2,a3,a4,a5)

a1to5rbind

class(a1to5rbind)

str(a1to5rbind)

dim(a1to5rbind)

# ======================================

# using list() - notice the dim()

# function doesn't return anything

# dim() does not work for lists

# ======================================

a1to5list <- list(a1,a2,a3,a4,a5)

a1to5list

class(a1to5list)

str(a1to5list)

dim(a1to5list)

# ======================================

# create some lists made up of other lists

# ======================================

alist1 <- list(x, z, sinz, a1to5list)

alist1

str(alist1)

class(alist1)

# ======================================

# using data.frame() - this is a special

# kind of list - this time dim() works

# ======================================

a1to5df <- data.frame(a1,a2,a3,a4,a5)

a1to5df

class(a1to5df)

str(a1to5df)

dim(a1to5df)

# ======================================

# this time with stringsAsFactors

# set to FALSE

# ======================================

a1to5dfnf <- data.frame(a1,a2,a3,a4,a5,

stringsAsFactors = FALSE)

a1to5dfnf

class(a1to5dfnf)

str(a1to5dfnf)

dim(a1to5dfnf)

# ======================================

# EXERCISE 1

# Create a new data frame object called `df1`

# using the # `data.frame()` command. Combine

# the object `y` with the built in constant

# for `month.name` and `month.abb`.

y

month.name

month.abb

mydf <- data.frame(y, month.name, month.abb)

c(y,month.name,month.abb)

#

# Create the data frame again and call it

# `df2` using the same 3 objects

# (`y`, `month.name` and `month.abb`) and

# set `stringsAsFactors` to FALSE.

# ======================================

# exercise 1 - key

df1 <- data.frame(y, month.name, month.abb)

df1

str(df1)

df2 <- data.frame(y, month.name, month.abb,

stringsAsFactors = FALSE)

df2

str(df2)

# Session 2 ==============================

# ======================================

# look at data objects - you can use

# the fix() command to edit your data

# ======================================

fix(z)

fix(alist1)

fix(df2)

# ======================================

# use head() and tail() to only look

# at a few rows at a time

# ======================================

head(df1)

tail(df1)

# ======================================

# you can also invoke the viewer

# with View()

# ======================================

View(df1)

view(df1)

# ======================================

# let's make a simple plot

# ======================================

plot(z,sinz)

# ======================================

# add some customization

# ======================================

plot(z, sinz,

xlab='Z = Sequence 0 to 10 by 0.5',

ylab='Sin(Z)',

main='Main title',

sub='example subtitle')

# ======================================

# add a BLUE line using lines()

# ======================================

lines(z,sinz,col='blue')

# ======================================

# customize the points using points()

# plotting character pch 23 is a filled diamond

# col defines the color

# bg defines the filled or background color

# ======================================

points(z,sinz,pch=23,col='red',bg='black')

# ======================================

# select code above, right click and "run selection"

# or highlight code and click CTRL-R

# specifically run the following code all together

# the spaces and line returns added for clarity

# Note: RStudio helps with good formatting practices

# ======================================

# ======================================

# all together in one block

# ======================================

plot(z, sinz,

xlab = 'Z = Sequence 0 to 10 by 0.5',

ylab = 'Sin(Z)',main='Main title',

sub = 'example subtitle')

lines(z, sinz, col = 'blue')

points(z, sinz, pch = 23, col = 'red', bg = 'black')

# ======================================

# use graphics device to make a PDF

# of the plot

# ======================================

pdf(file = "plot1.pdf")

plot(z, sinz,

xlab = 'Z = Sequence 0 to 10 by 0.5',

ylab = 'Sin(Z)',main='Main title as of today my name is Melinda',

sub = 'example subtitle')

lines(z, sinz, col = 'blue')

points(z, sinz, pch = 23, col = 'red', bg = 'black')

dev.off()

# ======================================

# EXERCISE 2

# Look up which "device" will create and

# save the plot as a JPEG. Use the commands

# above as your guide and create and save the

# figure as a JPEG formatted file.

#

# Create a second JPEG where the width is

# 750 pixels and the height is 500 pixels

# and set the background color to yellow

# and the quality to 50.

# ======================================

# exercise 2 - key

jpeg(file = "plot1.jpg")

plot(z, sinz,

xlab = 'Z = Sequence 0 to 10 by 0.5',

ylab = 'Sin(Z)',main='Main title',

sub = 'example subtitle')

lines(z, sinz, col = 'blue')

points(z, sinz, pch = 23, col = 'red', bg = 'black')

dev.off()

jpeg(file = "plot1yellow.jpg",

width=750, height=500,

bg = "yellow",

quality = 50)

plot(z, sinz,

xlab = 'Z = Sequence 0 to 10 by 0.5',

ylab = 'Sin(Z)',

main='Main title',

sub = 'example subtitle')

lines(z, sinz, col = 'blue')

points(z, sinz, pch = 23, col = 'red', bg = 'black')

dev.off()

# ======================================

# let's install a package

# ======================================

install.packages("ggplot2")

# ======================================

# load the package using library()

# ======================================

library(ggplot2)

# ======================================

# once it is loaded we can get info

# and help on the package - see

# more in the "Packages" window

# ======================================

help(package = "ggplot2")

# ======================================

# now that ggplot2 is loaded we can

# use the qplot() function

qplot(z, sinz,

geom = c("point", "line"),

xlab = 'Z = Sequence 0 to 10 by 0.5',

ylab = 'Sin(Z)',

main = 'Main title')

# ======================================

# we can also call variables inside a data frame

# ======================================

df2 <- data.frame(z,sinz)

ggplot(df2, aes(x=z, y=sinz)) +

geom\_line(colour = "red", linetype = "dashed") +

geom\_point(shape = 23,

colour = "red",

fill = "black") +

xlab("z is sequence from 0 to 10 in units of 0.5") +

ylab("Sin(z)") +

ggtitle("Here is a Main Title") +

theme\_light()

# ======================================

# here is the plot using basic R

# graphics

# ======================================

plot(z, sinz,

xlab = 'Z = Sequence 0 to 10 by 0.5',

ylab = 'Sin(Z)',main='Main title',

sub = 'example subtitle')

lines(z, sinz, col = 'blue')

points(z, sinz, pch = 23, col = 'red', bg = 'black')

# ======================================

# using ggplot2 to build the plot layer by layer

# ======================================

# ======================================

# make the basic plot window

# ======================================

p <- ggplot(df2, aes(x=z, y=sinz))

p

# ======================================

# add a line geom (geometric object)

# ======================================

p <- p + geom\_line(colour = "red", linetype = "dashed")

p

# ======================================

# add the points geom

# ======================================

p <- p + geom\_point(shape = 23,

colour = "red",

fill = "black")

p

# ======================================

# add some labels and a title

# ======================================

p <- p + xlab("z is sequence from 0 to 10 in units of 0.5") +

ylab("Sin(z)") +

ggtitle("Here is a Main Title")

p

# ======================================

# we can apply a basic "theme" for the overall

# look and style of the plot

# ======================================

p <- p + theme\_light()

p

# ======================================

# Let's also add the ggthemes package

# and really try out some cool

# plot styles

# ======================================

library(ggthemes)

p <- p + theme\_economist() + ggtitle("The Economist Theme")

p

p <- p + theme\_fivethirtyeight() + ggtitle("The 538 Theme")

p

p <- p + theme\_tufte() + ggtitle("The Edward Tufte Theme")

p

p <- p + theme\_wsj() + ggtitle("The WSJ Theme")

p

# ======================================

# it is always a good idea to make sure

# you cite the packages you used and give

# the creators proper credit

# it also helps document the version

# you are using

# ======================================

citation(package = "base")

citation(package = "ggplot2")

citation(package = "ggthemes")

# ======================================

# also document all of current

# session settings, add-ons, versions,

# and computer system info.

# ======================================

sessionInfo()

# session 3 ===============================

# ======================================

# let's make some more data objects

# ======================================

x <- 3 \* 8 + 6

y <- 1:12

y2 <- y\*\*2

# ======================================

# create a matrix

# ======================================

df1 <- cbind(y, y2, month.abb, month.name)

class(df1)

str(df1)

# ======================================

# create a data frame

# ======================================

df1a <- data.frame(y,y2,month.abb,month.name)

class(df1a)

str(df1a)

# ======================================

# create a list

# ======================================

list1 <- list(x,y,df1a)

class(list1)

str(list1)

# you can use the fix() function

# to edit data and save the changes

fix(df1a)

# ======================================

# select the element on the 3rd row

# and 3rd column

# ======================================

df1[3,3]

# ======================================

# select the whole 3rd column

# ======================================

df1[,3]

# ======================================

# select the 5th row and every column

# in that row

# ======================================

df1[5,]

# ======================================

# compare the 1st column in the

# data matrix df1 and 1st column in

# the data frame df1a

# ======================================

df1[,1]

class(df1[,1])

str(df1[,1])

df1a[,1]

class(df1a[,1])

str(df1a[,1])

# ======================================

# create a sequence of numbers

# put them into a vector z

# perform math on that vector and

# save it. then combine both

# vectors into a data frame

# ======================================

z <- seq(0,10,.5)

sinz <- sin(z)

df2 <- data.frame(z, sinz)

# ======================================

# now let's clean up some of our objects

# using the rm() command to selectively

# remove objects no longer needed

# ======================================

ls()

rm(y, y2, z, sinz)

ls()

# ======================================

# other ways to select a column of data

# i.e. select a variable in a dataset.

# ======================================

df2[,1] # use the column number

df2$z # select by column name

# ======================================

# after selecting a variable

# by the column name, then select the

# 3rd element (3rd row of the month.name

# variable) which is "March"

# ======================================

df1a$month.name[3]

# find the row for which y equals 5

# in df1a

pickrow <- df1a$y == 5

df1a[pickrow, ]

pickrow <- df1a$y > 6

df1a[pickrow, ]

df1a[df1a$y == 5, ]

# ======================================

# use save.image() or save() to save all

# or selected objects - these are saved

# as \*.RData files

# ======================================

save.image("allobjects.RData")

save(df2, file="df2.RData")

# ======================================

# remove all of the objects

# and then load them back - either one

# at a time or all of them together

# ======================================

rm(list = ls())

load(file="df2.RData")

load("allobjects.RData")

ls()

# ======================================

# get data from dropbox at

# https://www.dropbox.com/sh/vlo5bzrl5ayo1bk/AADD0WieyuEdyGwiveuCoRr-a?dl=0

# download these files and put them

# into your working directory

# for this next exercise

# ======================================

# ======================================

# read in as comma delimited

# ======================================

data.rt <- read.table(file="Dataset\_01\_comma.csv",header=TRUE,sep=",")

data.rt

# ======================================

# read in a CSV

# ======================================

data.csv <- read.csv(file="Dataset\_01\_comma.csv")

data.csv

# ======================================

# read in a TAB delimited TXT file

# ======================================

data.tab <- read.delim(file="Dataset\_01\_tab.txt")

data.tab

# ======================================

# read in a XLS Excel file

# functions from readxl package

library(readxl)

# ======================================

data.xls <- read\_excel("Dataset\_01.xls", sheet=1)

data.xls

# ======================================

# read in a XLSX Excel file

# ======================================

data.xlsx <- read\_excel("Dataset\_01.xlsx", sheet=1)

data.xlsx

# suppose I have a 2nd sheet named "new"

datanew.xlsx <- read\_excel("Dataset\_01addsheet.xlsx", sheet="new")

datanew.xlsx

data.xlsx <- read\_excel("Dataset\_01.xlsx", sheet=1)

data.xlsx

# ======================================

# read in a SPSS SAV file

library(foreign)

# ======================================

data.spss <- read.spss(file = "Dataset\_01.sav",

to.data.frame=TRUE)

data.spss

# ======================================

# read in a SAS Export XPT file

# ======================================

data.xpt <- read.xport(file = "Dataset\_01.xpt")

data.xpt

# Suggestion in class

# try HAVEN package

install.packages("haven")

library(haven)

test1.sas <- read\_sas("http://crn.cancer.gov/resources/ctcodes-procedures.sas7bdat")

test2.sas <- read\_sas("ctcodes-procedures.sas7bdat")

# ======================================

# IGNORE THIS - USE HAVEN EXAMPLE

# ABOVE WITH read\_sas() FUNCTION

# Using the sas7bdat packafe

# try reading the SAS file directly

# this is experimental and didn't work

# ======================================

#data.sas <- read.sas7bdat(file = "Dataset\_01.sas7bdat")

#data.sas

# session 4 ==========================

# ======================================

# create new variables BMI from

# weight and height

# ======================================

data.csv$bmiPRE <- (data.csv$WeightPRE\*703)/((data.csv$Height\*12)\*\*2)

data.csv$bmiPOST <- (data.csv$WeightPOST\*703)/((data.csv$Height\*12)\*\*2)

# ======================================

# you can also use attach and detach

# datasets and then call the variables

# without using the $ selector

# ======================================

attach(data.csv)

diff <- bmiPOST - bmiPRE # creates diff variables

# stands alone not attached

mean(diff)

# create using the variables inside the dataset

# but attach them back to the dataset

data.csv$diff2 <- bmiPOST - bmiPRE

detach(data.csv)

# ADD ENTIRE DATA UPDATE with attach \*\*

# ======================================

# once the dataset is detached

# go back to using the $ selector

# ======================================

data.csv$diff <- diff

# after assigning the new difference

# score diff to the dataset, remove it

# from the global environment.

rm(diff)

attach(data.csv)

bmiPRE

detach(data.csv)

# ======================================

# WRITING or SAVING data

# you can save the file out to \*.RData

# ======================================

save(data.csv, file="datacsv.RData")

# ======================================

# you can write datasets out as CSV

# ======================================

write.csv(data.csv,

file="datacsv.csv")

# ======================================

# write a dataset out as a TAB delimited

# TXT file

# ======================================

write.table(data.csv,

file="datacsv.txt",

sep="\t")

# ======================================

# we'll recreate the BMI variables

# and make a histogram

# ======================================

data.csv$bmiPRE <- (data.csv$WeightPRE\*703)/((data.csv$Height\*12)\*\*2)

data.csv$bmiPOST <- (data.csv$WeightPOST\*703)/((data.csv$Height\*12)\*\*2)

hist(data.csv$bmiPRE)

# ======================================

# find the typo and fix it

# this seems trivial but it is important

# to document and track data corrections

# like these

# recalculate and redo the histogram

# this time will use probabilities

# instead of frequencies for the histogram

# ======================================

attach(data.csv)

data.csv[18,"Height"] <- 5.6

data.csv$bmiPRE <- (WeightPRE\*703)/((Height\*12)\*\*2)

data.csv$bmiPOST <- (WeightPOST\*703)/((Height\*12)\*\*2)

detach(data.csv)

hist(data.csv$bmiPRE, freq=FALSE)

lines(density(data.csv$bmiPRE))

# However, I advise again using attach()

# and detach(), see more at

# http://www.r-bloggers.com/to-attach-or-not-attach-that-is-the-question/

# ======================================

# make a plot of the PRE vs POST BMI

# add a linear fit line and a

# lowess smoothed fit line

# ======================================

plot(data.csv$bmiPRE, data.csv$bmiPOST, "p")

abline(lm(data.csv$bmiPOST ~ data.csv$bmiPRE), col="red")

lines(lowess(data.csv$bmiPRE, data.csv$bmiPOST), col="blue")

# ======================================

# do the plot again using ggplot options

# ======================================

p <- ggplot(data.csv, aes(bmiPRE, bmiPOST))

p

p <- p + geom\_point()

p

p + geom\_smooth(method="lm") +

facet\_wrap(~GenderCoded) +

ggtitle("Panels for Gender")

# ======================================

# let's create Gender as a Factor

# instead of as just number codes

# Factors are useful in plots

# and tables providing labels for the

# labels. Factors are also helpful

# in various models.

# ======================================

data.csv$GenderFactor <- factor(data.csv$GenderCoded,

levels = c(1,2),

labels = c("Male","Female"))

data.csv$GenderFactor

str(data.csv$GenderFactor)

class(data.csv$GenderFactor)

table(data.csv$GenderFactor)

# ======================================

# we'll use the GenderFactor to split

# the plots into different panels by gender

# ======================================

p <- ggplot(data.csv, aes(bmiPRE, bmiPOST)) +

geom\_point() +

geom\_smooth(method = "loess", colour = "red", se = FALSE) +

geom\_smooth(method = "lm", colour = "blue") +

facet\_wrap(~GenderFactor) +

ggtitle("Panels for Gender, RED smoothed line, BLUE linear fit line")

p

# ======================================

# we can also use this factor to color

# code the points and associated model fits

# ======================================

p <- ggplot(data.csv, aes(bmiPRE, bmiPOST)) +

geom\_point(aes(colour = GenderFactor)) +

geom\_smooth(method = "lm", aes(colour = GenderFactor)) +

ggtitle("Colored by Gender")

p

# ======================================

# let's take a quick look at the linear

# model object that has the output

# from fitting a linear model to the

# PRE and POST BMI data.

# ======================================

fit1 <- lm(bmiPOST ~ bmiPRE, data=data.csv)

fit1

summary(fit1)

coef(fit1)

anova(fit1)

# DAY 2 ==========================================

getwd()

setwd("C:/MyGithub/CDCRworkshopCode")

# ======================================

# more on data management

# creating new variables

# recoding and missing data

# ======================================

# ======================================

# let's create Gender as a Factor

# instead of as just number codes

# Factors are useful in plots

# and tables providing labels for the

# labels. Factors are also helpful

# in various models.

# ======================================

data.csv <- read.csv(file="Dataset\_01\_comma.csv")

data.csv

data.csv[18,"Height"] <- 5.6

data.csv$bmiPRE <- (data.csv$WeightPRE\*703)/((data.csv$Height\*12)\*\*2)

data.csv$bmiPOST <- (data.csv$WeightPOST\*703)/((data.csv$Height\*12)\*\*2)

data.csv$GenderFactor <- factor(data.csv$GenderCoded,

levels = c(1,2),

labels = c("Male","Female"))

data.csv

head(data.csv[,c("GenderCoded","GenderFactor")])

data.csv$GenderFactor

str(data.csv$GenderFactor)

class(data.csv$GenderFactor)

# table() is similar to PROC FREQ.

table(data.csv$GenderFactor, useNA = "ifany")

barplot(table(data.csv$GenderCoded))

barplot(table(data.csv$GenderFactor))

# try some more recoding

# create new variable with BMI categories

# we'll use within() function to make the

# coding easier

# the code below creates 2 new variables

# with character labels for each category

# note you need to initialize the new variable

# bmiPREcat with NAs (missing) to start

data.csv$Age == 45

data.csv$Age != 45

data.csv$GenderFactor != "Female"

data.csv$bmiPREcat <- NULL

data.csv <- within(data.csv,{

bmiPREcat <- NA

bmiPREcat[bmiPRE < 18.5] <- "underweight"

bmiPREcat[bmiPRE >= 18.5 & bmiPRE < 25] <- "normal"

bmiPREcat[bmiPRE >= 25 & bmiPRE < 30] <- "overweight"

bmiPREcat[bmiPRE >= 30 & bmiPRE < 60] <- "obese"

})

table(data.csv$bmiPREcat, exclude = NULL)

table(data.csv$bmiPREcat)

# now do it again for the POST BMI

data.csv <- within(data.csv,{

bmiPOSTcat <- NA

bmiPOSTcat[bmiPOST < 18.5] <- "underweight"

bmiPOSTcat[bmiPOST >= 18.5 & bmiPOST < 25] <- "normal"

bmiPOSTcat[bmiPOST >= 25 & bmiPOST < 30] <- "overweight"

bmiPOSTcat[bmiPOST >= 30] <- "obese"

})

a <- NA

a == NA

a < NA

a > NA

is.na(a)

table(data.csv$bmiPOSTcat)

table(data.csv$bmiPOSTcat, exclude=NULL)

str(data.csv$bmiPREcat)

str(data.csv$bmiPOSTcat)

# notice that the table lists

# the categories in alphabetical order

# which is not what we want

# we would like overweight listed

# before obese.

# so let's assign an order to our levels

# we will create an ordered factor

data.csv$bmiPREcat.or <- factor(data.csv$bmiPREcat,

order = TRUE,

levels = c("underweight",

"normal",

"overweight",

"obese"))

data.csv$bmiPOSTcat.or <- factor(data.csv$bmiPOSTcat,

order = TRUE,

levels = c("underweight",

"normal",

"overweight",

"obese"))

str(data.csv$bmiPREcat)

str(data.csv$bmiPREcat.or)

str(data.csv$bmiPOSTcat.or)

table(data.csv$bmiPREcat)

table(data.csv$bmiPREcat.or)

table(data.csv$bmiPOSTcat.or)

# using these categories

# create a barplot from the table frequencies

barplot(table(data.csv$bmiPREcat.or))

# alternative using with with() function

with(data.csv,

barplot(table(bmiPREcat.or)))

# \*\*\*\*\*\*\*\*\*\*\*\*\* gender recoding \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# this is the variable we need to fix

data.csv$GenderSTR

# make a copy of this variable

# set all text case to lower case

data.csv$GenderNew <- data.csv$GenderSTR

data.csv$GenderNew <- tolower(data.csv$GenderNew)

# use nested ifelse() statements to fix this

# 1. find all subjects with gender as "m" or "male" and not missing

# 2. if true set the fixed value to "Male" if not true

# 3. then ask if gender was "f" or "female"

# 4. if true set to female

# 5. if false set the rest to missing NA

data.csv$GenderFixed <- ifelse(((data.csv$GenderNew == "m") |

(data.csv$GenderNew == "male")) &

(data.csv$GenderNew != ""),

"Male",

ifelse(((data.csv$GenderNew == "f") |

(data.csv$GenderNew == "female")) &

(data.csv$GenderNew != ""),

"Female",

NA))

# run a quick check to compare all the ways gender was coded for

# a quick comparison

vars <- c("GenderSTR", "GenderCoded", "GenderFactor", "GenderNew", "GenderFixed")

data.csv[,vars]

# if desired make a list of the variables

# you don't want to keep and remove them

# all but GenderFixed

# 1. make a list of the variables you want to remove

# 2. create a variable index (TRUE's or "FALSE's) for

# which variables you want to remove by comparing

# these to the whole variable list in the dataset

# 3. use this index to keep all of the variables

# for which this is not true

vars <- c("GenderSTR", "GenderCoded", "GenderFactor", "GenderNew")

rmlist <- names(data.csv) %in% vars

data.csv <- data.csv[!rmlist]

# \*\*\*\*\*\*\*\*\* another request \*\*\*\*\*\*\*\*\*\*\*\*\*

# capture if a subject has any of the

# conditions across a set of variables

# and only code missing if all are missing

# and if one is a zero then final is 0...

x1 <- c( 0, 1,NA,NA, 1, 1, 0, 0,NA, 0)

x2 <- c( 0,NA,NA,NA, 1, 0, 1, 0, 0, 0)

x3 <- c( 0, 1,NA,NA,NA, 0, 0, 0,NA, 0)

x4 <- c( 1, 1,NA,NA, 0, 1, 0, 0, 1, 1)

x5 <- c( 0, 1,NA,NA, 0, 1, 1, 0,NA, 1)

xdf <- data.frame(x1,x2,x3,x4,x5)

xdf

is.na(xdf)

rowSums(is.na(xdf))

xdf$sum <- ifelse(rowSums(is.na(xdf[,1:5]))!=5,

rowSums(xdf[,1:5], na.rm=TRUE),

NA)

xdf

xdf$yn <- ifelse(rowSums(is.na(xdf[,1:5]))!=5,

as.numeric(rowSums(xdf[,1:5], na.rm=TRUE)>1),

NA)

xdf

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# suppose we wanted to set all subjects

# under the age of 35 to missing

data.csv$Age35plus <- data.csv$Age

data.csv[data.csv$Age35plus < 35, "Age35plus"] <- NA

# let's summarise this data

# look at Age and the new variable Age35plus

summary(data.csv[,c("Age", "Age35plus")])

summary(data.csv)

# define the variable list you're intereted in

# then use that as a filter to select only these variables

myvars <- c("Age", "bmiPRE", "bmiPOST", "GenderFactor")

summary(data.csv[,myvars])

# suppose I just want the women

# define a filter for the rows or subjects

myrows <- data.csv$GenderFactor == "Female"

myrows <- data.csv$bmiPRE > 30

summary(data.csv[myrows,myvars])

table(data.csv$GenderSTR)

# find mean Age with and without missing data

mean(data.csv$Age)

mean(data.csv$Age35plus)

mean(data.csv$Age35plus, na.rm=TRUE)

# code above run using with() function

with(data.csv, mean(Age))

with(data.csv, mean(Age35plus))

with(data.csv, mean(Age35plus, na.rm=TRUE))

# find out how much missing data there is

# using the function is.na()

is.na(data.csv$Age35plus)

sum(is.na(data.csv$Age35plus))

with(data.csv,

sum(is.na(Age35plus)))

# how much in the whole dataset

sum(is.na(data.csv))

# keep only data for complete cases

# subject 9 missing gender

# subjects 5 and 15 missing Age35plus

dataall <- na.omit(data.csv)

# data without subjects 5, 9 and 15

# find which cases have missing data

# create an index to find the cases with

# no missing data using complete.cases() function

i <- complete.cases(data.csv)

i

# use this index to retrieve the list of subjects with

# complete data

data.csv$SubjectID[i]

# since i is a logical vector, we can use the

# not operator ! to find all the cases where

# i is not TRUE or is FALSE

data.csv$SubjectID[!i]

data.csv$SubjectID[i==FALSE]

# we can use the logic vector i to keep

# all of the cases with complete data where i is TRUE

dataall2 <- data.csv[i,]

save.image("mycurrentwork.RData")

load("mycurrentwork.RData")

# make a plot using ggplot2

# for bmiPRE vs bmiPOST

library(ggplot2)

p <- ggplot(data.csv, aes(bmiPRE, bmiPOST)) +

geom\_point() +

geom\_smooth(method = "loess", colour = "red", se = FALSE) +

geom\_smooth(method = "lm", colour = "blue") +

ggtitle("BMI PRE v POST: RED smoothed line, BLUE linear fit line")

p

# ======================================

# we'll use the GenderFactor to split

# the plots into different panels by gender

# ======================================

p <- ggplot(data.csv, aes(bmiPRE, bmiPOST)) +

geom\_point() +

geom\_smooth(method = "loess", colour = "red", se = FALSE) +

geom\_smooth(method = "lm", colour = "blue") +

facet\_wrap(~GenderFactor) +

ggtitle("Panels for Gender, RED smoothed line, BLUE linear fit line")

p

# ======================================

# we can also use this factor to color

# code the points and associated model fits

# ======================================

p <- ggplot(data.csv, aes(bmiPRE, bmiPOST)) +

geom\_point(aes(colour = GenderFactor)) +

geom\_smooth(method = "lm", aes(colour = GenderFactor)) +

ggtitle("Colored by Gender")

p

# ======================================

# let's take a quick look at the linear

# model object that has the output

# from fitting a linear model to the

# PRE and POST BMI data.

# ======================================

fit1 <- lm(bmiPOST ~ bmiPRE, data=data.csv)

fit1

class(fit1)

summary(fit1)

coef(fit1)

anova(fit1)

library(knitr)

kable(as.data.frame(summary(fit1)$coefficients))

# =================================

# find the column or variable

# names in a data set use names() function

names(mtcars)

# session 1 =========================

# summary stats

help(package = "datasets")

data()

mtcars

str(mtcars)

# from the quick-R website

library(car)

fit <- lm(mpg~disp+hp+wt+drat, data=mtcars)

summary(fit)

summary(mtcars[,c("mpg","cyl","disp","hp")])

mycars <- mtcars

mycars$cyl <- as.factor(mtcars$cyl)

summary(mycars[,c("mpg","cyl","disp","hp")])

mycarsNA <- mycars

mycarsNA[(mycarsNA$cyl)==6,"mpg"] <- NA

summary(mycarsNA[,c("mpg","cyl","disp","hp")])

summary(mtcars)

mean(mtcars$mpg)

sd(mtcars$mpg)

fivenum(mtcars$mpg)

quantile(mtcars$mpg, c(.25,.75))

fivenum(mycarsNA$mpg)

mystats <- function(x, na.omit=FALSE){

if(na.omit)

x <- x[!is.na(x)]

m <- mean(x)

n <- length(x)

s <- sd(x)

med <- median(x)

q5 <- quantile(x, .05)

q25 <- quantile(x, .25)

q75 <- quantile(x, .75)

q95 <- quantile(x, .95)

return(c(n=n, mean=m, stdev=s, Median=med,

q5, q25,

q75, q95))

}

mystats(mtcars$mpg)

# let's run mystats for every

# variable in mtcars

sapply(X=mtcars, FUN=mean)

sapply(X=mtcars, FUN=fivenum)

sapply(X=mtcars, FUN=mystats)

myoutput <- sapply(X=mtcars, FUN=mystats)

mystats(mycarsNA$mpg, na.omit=TRUE)

# let's apply these to a range of variables

myvars <- c("mpg","cyl","disp","hp")

sapply(mycars[myvars], fivenum)

sapply(mtcars[myvars], fivenum)

out <- sapply(mtcars[myvars], fivenum)

kable(out)

sapply(mtcars[myvars], mystats)

out <- sapply(mtcars[myvars], mystats)

kable(out)

# select only rows of cars that have 6 cylinders

d1 <- mtcars[mtcars$cyl==6,myvars]

sapply(d1, mystats)

# select cars that have 4 cylinders (cyl==4) and

# are automatic transmission (am==0)

d2 <- mtcars[(mtcars$cyl==4 & mtcars$am==0),myvars]

sapply(d2, mystats)

# can also use the subset() function

d3 <- subset(mtcars, cyl==4 & am==0, select=myvars)

d3

# Hmisc package

library(Hmisc)

myvars <- c("mpg","cyl","disp","hp")

describe(mtcars[myvars])

# pastecs package

library(pastecs)

stat.desc(mtcars[myvars])

# psych package

library(psych)

describe(mtcars[myvars])

# avoid masking problems using 2 colons ::

Hmisc::describe(mtcars[myvars])

psych::describe(mtcars[myvars])

# session 2 =========================

# descriptions by group - use the by() function

by(mtcars$mpg, mtcars$cyl, mean)

dstats <- function(x)sapply(x,mystats)

by(mtcars[myvars], mtcars$cyl, dstats)

out <- by(mtcars[myvars], mtcars$cyl, dstats)

kable(out[[1]], caption=paste(mtcars$cyl[1]," cylinders"))

kable(out[[2]], caption=paste(mtcars$cyl[2]," cylinders"))

kable(out[[3]], caption=paste(mtcars$cyl[3]," cylinders"))

# doBy package

library(doBy)

out <- summaryBy(mpg+disp+hp~cyl, data=mycars, FUN=mystats)

kable(out)

# frequency and contingency tables

library(vcd)

library(gmodels)

data()

head(Arthritis)

str(Arthritis)

table(Arthritis$Improved)

with(Arthritis, table(Improved))

xtabs(~ Treatment + Improved, data=Arthritis)

mytable <- xtabs(~ Treatment + Improved, data=Arthritis)

mytable

margin.table(mytable, 1)

margin.table(mytable, 2)

prop.table(mytable)

prop.table(mytable, 1)

prop.table(mytable, 2)

addmargins(mytable)

addmargins(prop.table(mytable))

addmargins(prop.table(mytable,1), 2)

addmargins(prop.table(mytable,1)\*100, 2)

addmargins(prop.table(mytable,2)\*100, 1)

chisq.test(mytable)

mytable2 <- xtabs(~Improved+Sex, data=Arthritis)

mytable2

chisq.test(mytable2)

fisher.test(mytable2)

mytable3 <- xtabs(~Treatment+Improved+Sex, data=Arthritis)

mytable3

mantelhaen.test(mytable3)

# results suggest that the treatment received and

# improvements reported were not independent within

# each level of sex - in other words treated individuals

# improved more than those receving placebos when

# controlling for sex

# plots for categorical data

attach(Arthritis)

counts <- table(Treatment, Improved)

spine(counts, main="Spinogram of Treatment by Group",

gp=gpar(col = "blue", fill=rainbow(6)))

detach(Arthritis)

mosaic(~Treatment+Improved,

data=Arthritis,

gp = shading\_max, split\_vertical = TRUE,

main="Arthritis: [Treatment] [Improved]")

# correlations

head(state.x77)

states <- state.x77[,1:6]

cor(states)

cor(states, method="spearman")

cor.test(states[,2], states[,3])

sum(is.na(states))

library(psych)

corr.test(states, use="complete")

print(corr.test(states, use="complete"), short=FALSE)

# scatterplot matrices

library(car)

myvars <- c("Murder", "Population", "Illiteracy", "Income", "Frost")

states <- as.data.frame(state.x77[,myvars])

palette("default")

scatterplotMatrix(states)

scatterplotMatrix(states, spread=FALSE)

scatterplotMatrix(states, spread=FALSE,

diagonal = "boxplot")

# graphical correlation matrix - corrgram package

library(corrgram)

corrgram(states, order=TRUE,

lower.panel = panel.pts,

upper.panel = panel.ellipse,

diag.panel = panel.density,

main = "Corrgram of 5 variables from States.x77 dataset")

corrgram(states, order=TRUE,

lower.panel = panel.shade,

upper.panel = panel.pie,

text.panel = panel.txt,

main = "Corrgram of 5 variables from States.x77 dataset")

t.test(mpg ~ am, data=mtcars)

boxplot(mpg ~ am, data=mtcars,

col = c("blue","green"))

ggplot(data=mtcars, aes(as.factor(am), mpg)) +

geom\_boxplot()

# skip for now.....

boxplot(len ~ dose, data = ToothGrowth,

boxwex = 0.25, at = 1:3 - 0.2,

subset = supp == "VC", col = "yellow",

main = "Guinea Pigs' Tooth Growth",

xlab = "Vitamin C dose mg",

ylab = "tooth length",

xlim = c(0.5, 3.5), ylim = c(0, 35), yaxs = "i")

boxplot(len ~ dose, data = ToothGrowth, add = TRUE,

boxwex = 0.25, at = 1:3 + 0.2,

subset = supp == "OJ", col = "orange")

legend(2, 9, c("Ascorbic acid", "Orange juice"),

fill = c("yellow", "orange"))

# session 3 =========================

# linear models

head(state.x77)

colnames(state.x77)

states <- as.data.frame(state.x77[,1:5])

names(states)

q <- names(states)[4]

names(states)[4] <- "LifeExp"

names(states)

fit1 <- lm(Murder ~ ., data=states)

summary(fit1)

fit2 <- lm(Murder ~ Population + Income + Illiteracy, data=states)

summary(fit2)

anova(fit1, fit2)

fit3 <- lm(Murder ~ LifeExp, data=states)

summary(fit3)

# try variable selection methods with stepAIC in MASS package

fit1 <- lm(Murder ~ ., data=states)

summary(fit1)

stepAIC(fit1, direction="backward")

stepAIC(fit1, direction="both")

# library leaps - all subsets regression

library(leaps)

leaps <- regsubsets(Murder ~ ., data=states, nbest=4)

plot(leaps, scale="adjr2")

library(car)

subsets(leaps, statistic="cp",

main="Cp Plot for All subsets regression")

abline(1,1,lty=2,col="red")

# looking at interaction effects

fit <- lm(mpg ~ hp + wt + hp:wt, data=mtcars)

summary(fit)

library(effects)

plot(effect(term="hp:wt",

mod=fit,

vcov.=vcov,

xlevels=list(wt=c(2.2, 3.2, 4.2))),

multiline=TRUE)

# simple anova

library(multcomp)

head(cholesterol)

str(cholesterol)

table(cholesterol$trt)

df <- cholesterol

aggregate(df$response,

by=list(df$trt),

FUN=mean)

aggregate(df$response,

by=list(df$trt),

FUN=sd)

mystats <- function(x){

m <- mean(x)

n <- length(x)

s <- sd(x)

return(c(n=n, mean=m, stdev=s))

}

aggregate(df$response,

by=list(df$trt),

FUN=mystats)

fit <- aov(response ~ trt, df)

summary(fit)

library(gplots)

plotmeans(response ~ trt, data=df,

xlab="Treatment Groups",

ylab="Response",

main="Means Plot with 95% CI's")

# map example

head(USArrests)

aa <- tolower(rownames(USArrests))

aa2 <- rownames(USArrests)

crimes <- data.frame(state = tolower(rownames(USArrests)),

USArrests)

head(crimes)

library(maps)

states\_map <- map\_data("state")

crime\_map <- merge(states\_map, crimes, by.x="region", by.y="state")

head(crime\_map)

library(plyr)

library(mapproj)

crime\_map2 <- arrange(crime\_map, group, order)

head(crime\_map2)

ggplot(crime\_map2, aes(x=long, y=lat, group=group, fill=Assault)) +

geom\_polygon(colour="black") +

scale\_fill\_gradient2(low="#559999", mid="grey90", high="#BB650B",

midpoint=median(crimes$Assault)) +

expand\_limits(x=states\_map$long, y=states\_map$lat) +

coord\_map("polyconic")

# do one for Murder

ggplot(crime\_map2, aes(x=long, y=lat, group=group, fill=Murder)) +

geom\_polygon(colour="black") +

scale\_fill\_gradient2(low="#559999", mid="grey90", high="#BB650B",

midpoint=median(crimes$Murder)) +

expand\_limits(x=states\_map$long, y=states\_map$lat) +

coord\_map("polyconic")

# dates and times with lubridate

Sys.Date()

date()

today <- Sys.Date()

dob <- as.Date("1921-07-06") # Nancy Reagan's birthday

difftime(today, dob, units="weeks")

as.numeric(difftime(today, dob, units="days")/365)

diff1 <- today - dob

diff1

diff1/365

library(lubridate)

as.duration(diff1)

year(today) - year(dob)

wday(dob)

wday(dob, label=TRUE)