R Code (Comprehensive)

Note: generic object names will be shown as:

dfname = a data frame object

dataset = a given dataset to be specified

dataset$variable = dataset name with a variable (unspecified type; use as applicable)

dataset$continuous\_variable = dataset name with continuous variable specified

dataset$categorical\_variable = dataset name with categorical variable specified

**Maintenance**

# What is my working directory? Find out with:

getwd()

# Running an entire script: save a script to C:/Users/allen.baumgarten/Documents and then call with the following. Be sure to specify *which* script as command shown below is a generic example only:

source("name\_of\_script.r") # can add ,echo = TRUE if we want to print all command lines but this is normally unnecessary

# What packages are currently installed?

search()

# Export dataset or column as a csv file:

write.csv(dataset, file = "filename.csv") # exports to working directory

write.csv(dataset, file = 'C:/CRC/filename.csv', row.names = TRUE) # exports to a specific folder on C: drive

# Assign to dummy variable 'd' the datasets in a package, e.g., AppliedPredictiveModeling:

d <- data(package = "AppliedPredictiveModeling")

d$results[, "Item"] # print datasets included

# Search for attached datasets and detach if necessary (can cause confusion if variables named similarly):

intersect(search(), objects())

# "mixedall" was seen after a search() to be returned on line one, position two just after ".GlobalEnv". Run the following position 2 to detach:

detach(mixedall, pos = 2L, unload = FALSE, character.only = FALSE, force = FALSE)

# Detach lists:

rm(list = setdiff(ls(), lsf.str()))

# Create a data frame, add columns to it:

mynewdataframe <- data.frame(variableA, variableB, variableC)

mynewdataframe$colD <- (variableA + variableB / variableC)

**Useful Libraries**

library(nortest) # Anderson-Darling and other statistical tests

library(pastecs)

library(tigerstats)

library(readr)

🡪 my\_data <- read\_csv(“datafile.csv”) # loads a csv into memory but file must first be in the working directory getwd()

🡪 my\_data <- read\_tsv(“datafile.txt”) # loads a txt into memory but file must first be in the working directory getwd()

library(quantreg) # see <https://www.r-bloggers.com/quantile-regression-in-r-exercises/>

# see <http://data.library.virginia.edu/getting-started-with-quantile-regression/>

library(xlsx)

mydata <- read.xlsx("c:/myexcel.xlsx", 1) # read in the first worksheet from the workbook myexcel.xlsx; first row contains variable names

mydata <- read.xlsx("c:/myexcel.xlsx", sheetName = "mysheet") # read in the worksheet named mysheet

**General Statistics**

# Basic stats:

tapply(dataset$continuous\_variable, dataset$continuous\_variable, FUN=sum) # aggregate by categorical variable

mean(dataset$variable)

median(dataset$variable)

sd(dataset$variable)

hist(dataset$variable)

plot(density(dataset$continuous\_variable, na.rm = TRUE), main = "") # kernal density estimation plot

quantile(dataset$variable, probs = c(.25..50..75)) # quantile is the function and desired probabilities are inputted as a vector

# Aggregation functions:

(newdataset <- aggregate(dataset[,6:7], list(dataset$Categorical\_variable), sum)) # aggregates column #’s 6 and 7 by category

aggregate(dataset$continuous\_variable ~ dataset$categorical\_variable, dataset, mean) # aggregates based on categorical variable

Example using Base-R: see https://www.rforexcelusers.com/make-pivottable-in-r/

> abcd

a b c d

1 Yes 5 new This

2 Yes 4 new This

3 No 7 new This

4 No 6 new That

5 Maybe 5 new That

> aggregate(abcd$b, by=list(abcd$a), FUN=mean, na.rm=TRUE)

Group.1 x

1 Maybe 5.0

2 No 6.5

3 Yes 4.5

> aggregate(abcd$b, by=list(abcd$d,abcd$a), FUN=mean, na.rm=TRUE)

Group.1 Group.2 x

1 That Maybe 5.0

2 That No 6.0

3 This No 7.0

4 This Yes 4.5

Example using dplyr:

> reduced\_dataset <- filter(dataset, dataset$Categorical\_variable %in% c("Jan","Feb","Mar"))

> reduced\_dataset <- group\_by(reduced\_dataset, Category1, Category2)

> reduced\_dataset <- summarise(reduced\_dataset, sum\_amount = sum(Continuous\_variable))

table(dataset$categorical\_variable)

NROW(dataset$variable) # row number for a variable; obviously should match other variables of the same table

fivenum(dataset$continuous\_variable) # Be aware of the fact that fivenum() and summary() have differing ways of calculating the medians, and 1st and 3rd quartiles. Check with stat.desc() first.

# Calculate a mean (or other quantity) but with zeros (or any other value) excluded. Also works with other functions like sd() or sum(). Can also use other logical operators like ‘&’ to exclude/include certain multiple values. Example:

dataset <- c(2,5,4,2,0,0,0)

mean(dataset) # zeros are included

[1] 1.857143

mean(dataset[dataset!=0]) # zeros are excluded

[1] 3.25

mean(dataset[dataset!=2 & dataset!=0]) # zeros and ‘2’ are excluded

[1] 4.5

# Center a variable:

c\_var <- dataset$continuous\_variable - mean(dataset$continuous\_variable)

# Center and standardize a variable:

cs\_var <- scale(dataset$continuous\_variable)

# Using the pastec library for statistical summaries. First two are general; six lines following give by group:

library(pastecs)

stat.desc(dataset)

mean <- tapply(dataset$continuous\_variable, dataset$categorical\_variable, mean)

sd <- tapply(dataset$continuous\_variable, dataset$categorical\_variable, sd)

median <- tapply(dataset$continuous\_variable, dataset$categorical\_variable, median)

min <- tapply(dataset$continuous\_variable, dataset$categorical\_variable, min)

max <- tapply(dataset$continuous\_variable, dataset$categorical\_variable, max)

cbind(mean, sd, median, min, max)

sampledataset <- stat.desc(dataset) # creates an object with statistical summaries of a dataset

write.csv(sampledatset, “filename\_desired.csv”) # exports that object to a csv file in your working directory

unique(dataset$categorical\_variable, incomparables = FALSE) # lists unique categorical values in a specified column

# Probability assessments and plots:

library(nortest)

ad.test(dataset$ continuous\_variable)

shapiro.test(dataset$ continuous\_variable)

wilcox.test(dataset$continuous\_variable1, dataset$continuous\_variable2, conf.int = TRUE, conf.level = 0.95) # Between two continuous variables. Specifies to do a confidence interval for the median by a 95% CI.

wilcox.test(dataset$continuous\_variable1 ~ dataset$categorical\_binary\_variable, conf.int = TRUE, conf.level = 0.95) # Stratified by a binary variable. Also specifies a 95% CI for the median.

t.test(dataset$continuous\_variable1, dataset$continuous\_variable2) # independent 2-group test

t.test(dataset$continuous\_variable1 ~ dataset$categorical\_binary\_variable2) # x is a binary factor

t.test(dataset$continuous\_variable1, dataset$continuous\_variable2, paired= TRUE) # paired test

t.test(dataset$continuous\_variable1, mu=3) # H0: mu=3

qqnorm(dataset$ continuous\_variable)

qqline(dataset$ continuous\_variable, col = "red")

**General Graphing Techniques**

# Graphing

boxplot(variable1, variable2, data = dataset, main = "Title", xlab = "x axis label", ylab = "y axis label") # With labels

boxplot(dataset$continuous\_variable ~ dataset$categorical\_variable) # continuous variable ~ categorical variable

boxplot(dataset$continuous\_variable ~ dataset$categorical\_variable + dataset$categorical\_variable) # continuous variable ~ categorical variable 1 and categorical sub-variable 2

pairs(~ variable1 + variable2 + variable3 + variable4, data=dataset, main = "Title TBD") # Creates scatterplot matrix

hist(Dataset$variable) # simple histogram

hist(Dataset$variable, main="Main Title here", freq=FALSE, ylab="y axis label", xlab="x axis label") # relative frequency histogram

plot(variable1, variable2, data = dataset, main = "Title", xlab = "x axis label", ylab = "y axis label ") # simple scatter plot

abline(v=c(10,20,30), lty = 2, col = “gray”) # Adds vertical gray lines at axis points 10, 20, and 30. Can change to any values.

abline(lm(Yvariable ~ Xvariable), col = "red") # regression line (y ~ x))

lines(lowess(Xvariable, Yvariable), col = "blue") # lowess line (x , y)

# Boxplot with categorical covariates reordered as desired

newcategory <- factor(dataset$categorical\_variable, levels = c(“First covariate”, “Second covariate”, “Third”)) # creates factors but MUST use the same exact covariates as already existing in this variable!

boxplot(dataset$continuous\_variable ~ newcategory, main = “Title”, xlab = "x axis label", ylab = "y axis label ")

# Colored scatterplot

ggplot(dataset, aes(dataset $Xvariable, dataset $Yvariable, color = dataset$categorical\_variable)) + geom\_point()

**Least Squares Linear Regression**

# Regressed backwards compared to text by looking at Minutes as an explanation of Units:

model <- lm(Yvariable ~ Xvariable, data = dataset)

summary(model)

plot(Yvariable ~ Xvariable, data = dataset, xlab = " x axis label ", ylab = " y axis label ", main = "Title")

lines(lowess(Xvariable, Yvariable), col = "blue") # lowess line (x, y)

abline(lm(Yvariable ~ Xvariable), col = "red") # regression line (y ~ x))

anova(model)

# Multivariate Adaptive Regression Splines

model <- earth(dataset[c("x", "y")], dataset$Yvariable, degree = 3)

plot(model)

plot.earth.models(model)

evimp(model) # summarizes importances of the variable(s)

predicted <- (predict(model, dataset)

summary(predicted)

mse <- mean(dataset#Yvarible - predicted)^2)

print(mse)

**Logistic Regression**

# ROC and confusion matrix (see Hilbe, p. 84ff):

attach(dataset)

mylogitmodel <- glm(categorical\_variable ~ dataset\_variable, family = binomial, data = dataset)

out1 <- ROCtest(mylogitmodel, 10, type = "Sensitivity")

out1

out2 <- ROCtest(mylogitmodel, fold = 10, type = "ROC")

out2

confusion\_stat(out1$Predicted, out1$Observed)

**Generalized Linear Models**

**Generalized Additive Models**

# Use GAM to create slope profile for a continuous predictor against a logistic binomial outcome variable:

library(mgcv)

gam\_model\_name <- gam(dataset$yesnooutcome\_var ~ s(dataset$continuous\_var), family = binomial, data = dataset)

summary(gam\_model\_name)

plot(gam\_model\_name)

**Cluster Analysis**

# Call a cluster algorithm using Average (or Complete or other) linkage, Euclidean distance; with a dendrogram:

d <- dist(as.matrix(dataset))

hc <- hclust(d, method = "average", members = NULL) # Could substitute “Complete”

plot(hc)

summary(hc)

**ROC Curves** (see: https://hopstat.wordpress.com/2014/12/19/a-small-introduction-to-the-rocr-package/)

pred <- prediction(adr$Age, adr$Detection)

perf <- performance(pred, measure = "tpr", x.measure = "fpr")

plot(perf)

roc.perf = performance(pred, measure = "tpr", x.measure = "fpr") # Now we plot a ROC curve. Note: defined ‘pred’ above!

plot(roc.perf)

abline(a=0, b=1)

**Using dplyr**

http://genomicsclass.github.io/book/pages/dplyr\_tutorial.html

http://www.listendata.com/2016/08/dplyr-tutorial.html

https://www.r-bloggers.com/dplyr-example-1/

library(dplyr) # Load a dataset, then create subset objects (variables) based on your data

subdataset <- select(dataset, variable1, variable2, …) # this creastes an object

Note: the head function can be ommitted if desired:

head(select(dataset, variable1, variable2, …)) # selects and displays subset of columns

head(select(dataset, -variable2)) # selects and displays all columns except “variable 2”

head(select(dataset, variable1:variable3)) # selects and displays a range of columns, variables 1-3

head(select(dataset, starts\_with(“br”))) # selects and displays subset of all columns beginning with “br” in their names

head(select(dataset, ends\_with(“at”))) # selects and displays subset of all columns ending with “at” in their names

head(select(dataset, contains(“er”))) # selects and displays subset of all columns that contain “er” in their names

head(select(dataset, matches(“em”))) # selects and displays subset of all columns that contain “em” (matches on character strings)

head(select(dataset, one\_of(“er”))) # selects and displays subset of all columns that are from a group of names

filter(dataset, variable1 >= 5) # selects all rows from variable1 containing numerics >= 5

filter(dataset, variable1 >= 5, variable2 = 4) # selects all rows from variable1 containing numerics >= 5 AND variable2 = 4

filter(dataset, variable1 %in% c(“category 1”, “catetory 3)) # selects all rows from variabl1 containing “category 1” or “category 3” covariates.

For discussion on pipe operator %>%, see genomicsclass.github site above. Great discussion! Here is an example of ‘piping’ data:

dataset %>% # we ‘pipe’ the dataset data to the select function below

+ select(variable1, variable2) %>% # we now ‘pipe’ the new data frame of variable1 and 2 to the head() function below

+ head() # displays our new data frame subset

Returns these two variables in head summary format

Mutate creates new columns, one or more at a time:

dataset %>%

+ mutate(newcol1 = variable1/variable3, newcol2 = variable1 \* variable4) %>%

+ head # or push this into another object data frame