**R commands cheat sheet**

**Syntax**

A <- B: set variable A equal to value B

function(variable, variable): run function using variable(s) as arguments

A == B: test is A is equal to B

&: and

filename[row#, col#]: return single value in dataframe that we care about

filename[min.row:max.row, min.col:max.col]: return subset of data we care about

filename[min.row:max.row,]: return subset of data with all columns

filename[c(row#, row#, row#), min.col:max.col]: return subset of particular rows

filename$colname: return a single column

filename[filename$colname==value,]: return only the rows where a column equals value

**Commands (general)**

c(data1, data2, data3): make vector with values data1, data2, and data3

matrix (vector, #rows, #columns): make matrix out of values in vector

data.frame(colonevec, coltwovec): make data frame from a series of vectors

rownames(data.frame) <- c("row.one", "row.two", "row.three"): rename rows

read.csv(drive:/folder/folder/folder/filename.csv): read in csv

View(file.csv): view data in new tab

length(vector): get vector length

plot(density(data)): density distribution

levels(filename$variable): test what order r considers a grouping variable to be in

factor(filename$variable, levels=c("first group", "second group")): reorder grouping variable

class(data): figure out the format of your data

dim(filename): get rows and columns

head(filename): get first row

summary(filename): get summary

str(filename): see what kind of variable each column is

boxplot(filename$colname, ylab=”y label”, main=”title”): get boxplot

boxplot(y ~ x, data=filename): get comparative boxplot

abline(h=horizontal.coordinate, v=vertical.coordinate, col=color): draw line

source(file.choose(file)): choose script to access

set.seed(whole#): set seed (makes bootstrap results the same every time

log(val): natural log

filename[!is.na(filename$col.name),]: return data without NAs

read.cvs(“file.directory”)

read\_cvs(“file.driectory”)

tolower(thing): change to lower case

lme() #one factor random, one fixed, nested

seq(min, max, interval): series if evenly spaced x

glimpse(data): similar to str, But maybe a little neater, a little less info

skim(data): search for missing data (tells completion rate for each column), plus other info

drop\_na(data): remove any row that contains NA

filter(data, …): return rows with matching conditions (check help)

filter(data, !is.na(colname)): return all rows where the value of colname is not NA

data %>% mutate(colname = ifelse(is.na(colname), num, colname): change NAs in a col to num

mutate(data, …): adds new variables, preserves existing ones (see help; dplyr)

transmute(data, …): adds new variables and drops existing ones (see help; dyplr)

group\_by(variable): group by a variable (useful to pipe into summarise)

summarise(): summarize data (see help)

pivot\_longer(cols=col1:col2, names\_to=newcol, values\_to=newval): collapsing col1 and col2,

pivot table; new col is name of new col containing col1 and col2 names, new val is the original values of those two columns but now in one column

data %>% mutate(colname=ifelse(condition, fityes, fitno): if colname fits condition, change value

to fityes, otherwise change value to fitno

**Bootstrap**

**Basic resample**

#bootstrap with n number of resamples from vector (n=vector length)

nframe <- numeric(n)

for (i in 1:n) {

resample<-sample(vector, length(vector), replace=T)

nframe[i]<-mean(resample)

}

quantile(nframe, tail.boundary): bootstrap answer

**Two sample t-test**

#two samples t-test, data1 is one sample and data2 is the other

nframe <- numeric(n)

for (i in 1:n) {

resample1 <- sample(data1, length(data1), replace=T)

resample2 <- sample(data2, length(data2), replace=T)

nframe[i] <- mean(resample1) – mean(resample2)

}

quantile(nframe, tail.boundary): bootstrap answer

**Permutation test**

nframe<-numeric(n)

for (i in 1:n) {

perm.yns<-sample(filename$colname, length(filename$colname), replace=F)

pseudo.yes<-filename$colname[permu.yns=="Yes"]

pseudo.no<-filename$colname[perm.yns=="No"]

nframe[i]<-mean(pseudo.yes)-mean(pseudo.no)

}

sum(abs(nframe)>=abs(x.bar.yes-x.bar.no))/n: perm test p-value

**Some calculations**

mean(vector): average value of vector

mean(vector, percent.trimmed.from.each.tail): trimmed mean

median(vector): median value of vector

prod(vector)^(1/vector.length): geometric mean

1/mean(1/vector): harmonic mean

sd(vector): standard deviation of values in vector

var(vector): variance of values in a vector

range(vector): range of values in a vector

IQR(vector): interquartile range of values in vector

quantile(vector, x): xth percentile (eg. x=0.25 for 25th percentile); x can be a series of values

pfunct <- ecdf(file$colname): a special p function for this dataset

pfunct(value): the percentile of value

sum(filename$colname <= value)/length(filename$colname): percentile of value

**Misc tests**

shapiro.test(vector): Shapiro-Wilkes test of normality (p<0.01 means unlikely to be normal)

wilcox.test(dependent.variable~independent.variable, data=filename, alternative="")

power.t.test(n=, delta=, sd=, sig.level=, power=, type=, alternative=): check help

kruskal.test(y~x, data=filename)

anova(filename)

cor(x, y) #correlation test, prone to reacting to non-linear strong trends

cor(y~x, data=) #no implicit causation/directionality

cor.test(x, y, alternative= “”, method= “”) #correlation

str(cor.test(x, y)) #see results

qqnorm.with.sim.bounds(filename$colname, robust=T/F)

**Binomial calculations**

#x=num. successes, n=num. trials, p=prob. of success in one trial, q=area in left hand tail

#qbinom, etc only works on mathematical distributions (for real data, use quantiles)

#pbinom, etc only works for mathematical distributions (for real data, use ecdf)

dbinom(x, n, p): probability of exactly x successes

pbinom(x, n, p): probability of <= x successes

1-pbinom(x-1, n, p): probability of >= x successes

qbinom(q, n, p): smallest x such that pbinom(x, n, p)>=q

qbinom(1-q, n, p): largest x such that pbinom(x,n,p)<=q

**Poisson calculations**

#x=num. of events, l=average rate per unit time, q=area in left-hand tail

dpois(x, l): probability of exactly x in one unit of time

ppois(x, l): probability of <=x in one unit of time

1-ppois(x-1, l): probability of >=x in one unit of time

qpois(q, l): smallest x such that ppois(x,l)>=q

**Negative binomial calculations**

#k=num. of failures, x= num. of successes, p= probability of one success, q=are in left-hand tail

dnbinom(k, x, p): probability of exactly k failures

pnbinom(k, x, p): probability of at most k failures

qnbinom(q, x, p): smallest k such that pnbinom(k, x, p)

**Normal distributions**

#n=sample size, m=mean, sd=standard deviation, x=value, p=probability

rnorm(s, m, sd): sample s values from normal distribution

pnorm(x, m, sd): probability of <x in normal distribution

1 - pnorm(x, m, sd): probability of >x in normal distribution

qnorm(p, m, sd): x such that pnorm(x, m, sd)<p

**T distribution**

#t=number of standard errors away from mean, df=n-1, q=area of left-hand tail

pt(t, df): q bounded by t

qt(q, df): t to cut of q

t.test(filename$colname, conf.level=condfidence.level)$conf.int: confidence interval

**Plotting**

#y=response variable, x=explanatory variable, file=name of file, xlab=x label, ylab= y label

plot(y~x, data=file, xlab=”x label”, ylab=“y label”, cex=point\_size): plot the data stored in file

points(y~x, data=file, xlab="x label", ylab="y label”, col=col.code, pch=shape.code, cex=

point\_size): add points to existing plot

legend <- legend(position, categories, col=colors, pch=point\_types, cex=text size))

palette(palette\_source(n=n), name=palette\_name): set colors as first n of palette\_name

Ex palette(brewer.pal(n=3, name=”Dark2”)): set cols to first 3 in Dark2 in palette brewer

par(mar=c(bottom, left, top, right): set margin sizes for each margin

par(xpd=TRUE): allow r to put the legend outside of the figure

**Best fit line**

linear.regression.model <- lm(y~x, data=file) #create linear regression model

linear.regression.model <- lm(y~x1 + x2, data=file) #regression model, multiple x

linear.regression.model <- lm(y~x1 + x2 + x1 \* x2, data=file) #with interaction

plot(linear.regression.model): take a look graphically at how well your model is working

str(linear.regression.model) #take a peek at the regression model

abline(linear.regression.model, col=”col.code”, lwd=line\_wdith) #plot the best fit line

abline(v=value, col=”color”, lwd=line\_width): vertical line at x=value

abline(h=value, col=”color”, lwd=line\_width): horizontal lines at y=value

col=as.numeric(file.name$variable.we.care.about) #easy way to set colors

scatterplot(y~x, data=file.name, regLine=T.or.F, smooth=T.or.F)

pdf(file=filename): save plot as a pdf (use this to define dimensions so it’s the same every time

dev.off(): save and exit the pdf

coplot(y~x, data=file.name)

xyplot(y~x, data=file.name)

dataEllipse(x, y, levels=c(lower, upper)) #draws ellipse onto data

**ggplot**

plot <- ggplot(data=file.name, aes(x=x , y=y , colour=some.variable): save plot with set

aesthetics

plot <- ggplotly(): same idea, but it has some cool extra features (hovering over a point

will tell you about it)

Some things you can add to your plot aesthetics (plot)

+geom\_point(): add data points

+geom\_point(aes()): add data points with particular aesthetics\

+scale\_color\_manual(): manually set colors

+theme\_something(): an easy preset theme

theme\_classic: I like this one

theme\_dark

theme\_bw

+geom\_smooth(method=”method”): smoothed line, calculated by method

+scale\_color\_brewer(palette=”palette”): set to color brewer palette

+geom\_bar(stat=”identity” or “plot”): bar plot

+geom\_errorbar(): error bars

+geom\_jitter: make points not overlap so much

+labs(x=”xlab”, y= “ylab”, title=”main”)

+theme(): manually modify theme elements

axis.text=element\_blank: remove text elements

axis.text=element\_text(color=, suze=, angle=)

+facet\_grid(colname~.): plot each level un colname in its own plot, vertical

+facet\_grid(.~colname): plot each level un colname in its own plot, horizontal

+facet\_grid(colname1-colname2): a grid of all combos of the two

**ggplot examples**

plot + geom\_point(aes(color=factor(variable))): set point colors to correspond with a

variable

plot + geom\_point(size=s) + theme\_bw() + scale\_colour\_manual(values=ccbPalette) +

geom\_smooth(method=”lm”), xlab=x.label , ylab=y.label ) #some suggested

aesthetics to set (orange can be set as other pre-set things)

**Paired t-test**

#data$before is the same order and length (n) as data$after (each datapoint is paired)

#CL=confidence level (1.0-alpha)

diffs <- data.set$before – data.set$after #differences between before and after sets

std.error <- sd(diffs)/sqrt(n) #standard error of each observation

t.val <- (mean(diffs)-0)/std.error #t value

p.val <- 2\*pt(t.val,n-1) #p value

t.test(diffs, alternative="alternative hypothesis", conf.level=CL): paired t test

**Two sample t test**

#y=response variable, x=explanatory variable, file= file where data is stored

#CL=confidence level (1.0-alpha),

df <- satterthwaite(sd.1, n.1, sd.2, n.2) #find df for two sample t test?

t.test(y~x, data=file, alternative=”alternative hypothesis”, conf.level=CL): two-sample t test

**Sources**

source('C:/Users/Gray Ryan/Desktop/TAMUCC fall 2018 classes/MATH 6315 - STATS/Datasets/augPairs.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Old stuff/TAMUCC fall 2018 classes/MATH 6315 - STATS/Datasets/augPairs.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Old stuff/TAMUCC fall 2018 classes/MATH 6315 - STATS/Datasets/diagPlots.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Old stuff/TAMUCC fall 2018 classes/MATH 6315 - STATS/Datasets/traditionalForwardBackward\_lm.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Old stuff/TAMUCC fall 2018 classes/MATH 6315 - STATS/Datasets/multcompUtilities.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Old stuff/TAMUCC fall 2018 classes/MATH 6315 - STATS/Datasets/loessCrossvalidation.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Old stuff/TAMUCC fall 2018 classes/MATH 6315 - STATS/Datasets/mvShapiroSmallNTest.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Old stuff/TAMUCC fall 2018 classes/MATH 6315 - STATS/Datasets/permCorTest.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Spring 2019 classes/Stats II/Datasets/multcompUtilities.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Spring 2019 classes/Stats II/Datasets/kruskalPosthoc.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Spring 2019 classes/Stats II/Datasets/correctedLMEanova.R', encoding = 'UTF-8')

source('C:/Users/Gray Ryan/Desktop/Spring 2019 classes/Stats II/Datasets/profilePlot.R', encoding = 'UTF-8')

library(car)

library(MuMIn)

library(multcomp)

library(ggplot2)

library(lattice)

library(readr)

library(mnormt)

library(mvShapiroTest)

library(lmodel2)

library(mgcv)

library(hier.part)

library(DAAG)

library(plyr)

library(sciplot)

library(nlme)

**Tidyverse**

install.packages(“tidyverse”): install tidyverse

library(tidyverse): make tidyverse usable

dataframename <- read\_delim(“filename.txt”, delim= “delimiter”): read data into tidyverse

as.matrix(dataframe): change to matrix

%>%: pipe character

new <- dataframe %>% rename(new\_colname = old\_colname): rename a column

new <- dataframe %>% select(colname1, colname2, … colnamen): reorder columns

**RMarkdown**

**About**

\*filenames end in .Rmd

\*For authoring html, pdf, powerpoint slides, embedded r scripts

\*assumes everything is just text and not code unless you indicate to run in r (‘’’{r}, ‘’’)

**Syntax**

‘’’{r} start of chunk

‘’’ end of chunk

**Not entirely sure**

**Searches?**

traditional.forward(min.model, file[columns], alpha.in=) #alpha.in=0.05 as default

traditional.backward(max.model) #default is 0.1

exhaustive.search(file$y.var, file[,min.col:max.col]) #adj R^2 is R w/ penalty for

complexity

exhaustive.search.2(file$y.var, file[,min.col:max.col]) #considers interactions, eg. two x’s

are multiplied together

**Other**

nls(y~a+b\*x1+c\*x2…etc…d\*xn^k, data=file, start=list(a=prediction,

b=prediction, etc): curved model, everything needs to be positive

diag.plots(filename)

predict(file, newdata=list(x.var=what.to.predict), interval=”” level=conf.level)

vector.of.cooks.distances <- cooks.distance(filename)

**Models and tests**

aug.pairs(data) #parcimony means use the simpler model when possible

vif(model) #Variance-inflation factors for linear, gen linear, and other models; measure

of collinearity, problem if over a vif of 5

dredge() # full.model <- update(full.model, na.action=na.fail) necessary for dredge

min.model <- lm(y~1, data=filename)

max.model <- lm(y~x1 + x2 + x3 + …. +xn, data=filename)

max.model <- lm(y~., data=filename)

lm.but.for.anova <-aov(y~x,data=filename)

par(mfrow=c(2,3)) #I really don’t know what this does, sets how many graphs?

model.glht <- glht(aov.model, linfct=mcp(factor.name="Tukey")) #sets up contrasts

summary(model.glht, test=adjusted("Westfall")) #if Tukey is well-balanced

summary(model.glht, test=adjusted("Shaffer")) #if Tukey is not well-balanced

holm.summary <-summary(model.glht, test=adjusted("Holm"))

multcomp.summary.to.graph(westfall.summary) #summary to graph

attr(K, "dimnames")[[1]] <- c("c1", "c2")

colnames(K)[1] <- "feedcasein" #for easier reading

mvShapiroTest(as.matrix(filename[, #:#])) #goodness of fit, null=multivariate normality

mv.shapiro.small.n.test(filename[, #:#]) #same but n<12

perm.cor.test(file$colname, file$colname, method= “”, n.perms=#.of.permutations)

points(y~x, data=LS, pch=19, col=(cooks.distance(lm.model.ls)>0.8)+1) #finding the biggest cook’s distance

predict(lm.model.ls, newdata=data.frame(totmass=3.5))

predict(lm.model.ls, newdata=data.frame(totmass=3.5), interval="confidence", level=0.99)

predict(lm.model.ls, newdata=data.frame(totmass=3.5), interval="prediction", level=0.99)

abline(model2.rma, model2.rma$regression.results[3,2], model2.rma$regression.results[3,3], col="green")

hr <- hier.part(filename$colname, data[, #define columns ], cof=”Rsqu”, barplot=F)

colnames(hp$IJ) <- c(“Independent”, “Joint”, “Total”)

scale(data) # to center second order models (necessary)

**Prediction model**

PRESS <- function(a.model) {

deleted.resids <- resid(a.model)/(1-hatvalues(a.model))

return(sum(deleted.resids^2))

}

PRESS.rse <- function(a.model) {return(sqrt(PRESS(a.model)/a.model$df.residual))}

PRESS(model); PRESS.rse(model); summary(model)$sigma

**Model averaging**

avg.model<-summary(model.avg(dredge(full.model)))

attr(avg.model$msTable, "term.codes")

avg.model$msTable[1:20,]

avg.model$coefmat.full

avg.model$importance

**lme**

library(nlme)

model<-lme(y~x, random=~1|patch, data=filename)

model.wls <- lme(y~fixed.variable, random=~1 | random.variable, data=filename, weights=varIdent(form=~1 | random.variable)

anova(model.wls)

marginal sums of squares (type III)use gls, unless you don’t care about the main factors

**glht**

file.glht<-glht(file.aov, linfct=exp(variable.name=c(vector of null hypotheses) )

summary(filename.glht, test=adjusted(“none”)

summary(filename.glht, test=adjusted(“free”) #no orthogonality

file$diff.colname <- factor(paste(“g.”, file$col.name, sep=””)) #change to categorical

file.glht$alternative <- “greater” #set alternative hypothesis

PRESS/PRESS.rse 🡪#loyn02 choosing a model #sometimes prediction is more useful than explanation