R commands cheat sheet

vector <- c(data1, data2, data3)

matrix <- matrix (vector, #rows, #columns)

data.frame <- data.frame(column one vector, column two vector)

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

imported.csv <- read.csv(drive:/folder/folder/folder/filename.csv)

view(imported.csv) #read data in new tab

length.of.vector <- length(vector)

one.entry.we.care.about <- filename[row#, column#]

subset.we.care.about <- filename[min row: max row, min column: max column]

subset.with.all.columns <- filename[min row: max row,]

subset.of.particular.rows <- filename[c(row#, row#, row#, row#), min column: max column]

one.column <- filename$columnname

just.data.of.specified.value <- filename[filename$colname==colvalue,]

just.data.of.specified.values <- filename[filename$colname==colvalue & filename$colname2==colvalue2,]

#Binomial calculations (x=# successes, n=# trials, p=prob of success in one trial, q=area in left hand tail)

#qbinom and stuff like that only works with mathematical distributions, see quantile for actual data

#pbinom and stuff like that only works with mathematical distributions, see ecdf for actual data

prob.of.exactly.x.successes <- dbinom(x, n, p)

prob.of.<=x.successes <- pbinom(x, n, p)

prob.of.>=x.successes <- 1-pbinom(x-1, n, p)

smallest.x.such.that.pbinom(x, n, p)>=q <- qbinom(q, n, p)

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

#Poisson calculations (x=# of events, l=average rate per unit time, q=area in left-hand tail)

prob.of.exactly.x.in.one.unit.of.time <- dpois(x, l)

prob.of.<=x.in.one.unit.of.time <- ppois(x, l)

prob.of.>=x.in.one.unit.of.time <- 1-ppois(x-1, l)

smallest.x.such.that.ppois(x, l)>=q <- qpois(q, l)

#Negative binomial calculations (k=#failures, x=#successes, p=prob of 1 success, q=area in left-hand tail)

exactly.k.failures <- dnbinom(k, x, p)

at.most.k.failures <- pnbinom(k, x, p)

smallest.k.such.that.pnbinom(k, x, p)>=q <- qnbinom(q, x, p)

average <- mean(filename$colname)

trimmed.mean <- mean(filename$colname, % trimmed from each tail)

median <- median(filename$colname)

geometric.mean <- prod(filename$colname)^(1/#values in col)

harmonic.mean <- 1/mean(1/filename$colname)

standard.deviation <- sd(filename$colname)

variance <- var(filename$colname)

min.and.max <- range(filename$colname)

IQR <- IQR(filename$colname)

25th.percentile <- quantile(filename$colname, 0.25)

1st.2nd.and.3rd.quartiles <- quantile(filename$colname, (1:3)/4)

middle.95%.of.data <- quantile(filename$colname, c(0.025, 0.975))

special.p.function.for.this.dataset <- ecdf(filename$colname)

percentile.of.x <- special.p.function.for.this.dataset(x)

percentile.of.x <- sum(filename$colname <= x)/length(filename$colname)

sample.from.normal.distribution <- rnorm(sample size, mean, standard deviation)

#Continuous distributions

prob.of.<x.in.normal.distribution <- pnorm(x, mean, std)

prob.of.>x.in.normal.distribution <- 1 – pnorm(x, mean, std)

x.such.that.pnorm(x, mean, std)<probability <- qnorm(probability, mean, std)

#Standard deviation is about individual data point, standard error is about sample means

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

q.bounded.by.t <- pt(t, df)

t.to.cut.off.q <- qt(q, df)

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

shapiro.wilk.normality.test <- shapiro.test(filename$colname) #p<0.01 means unlikely to be normal

n.spaces.to.put.shit <- numeric(n)

for (i in 1:n) {

resample<-sample(filename$colname, length(filename$colname)cut.off, replace=T)

n.spaces.to.put.shit[i]<-mean(resample) #bootstrap with n resamples

}

bootstrap.answer <- quantile(n.spaces.to.put.shit, tail boundary) #can use vector of confidence levels

density.distribution <- plot(density(data))

plot <- plot(y~x, data=filename, xlab=”x label”, ylab=”y label”)

points(y~x, data=filename, xlab="x label)", ylab="y label”, col=color code/name, pch=shape code)

legend <- legend()

legend("topleft", c("male", "female"), col=c("magenta", "cyan"), pch=c(1, 1))

diffs <- data.set$before – data.set$after #setting up paired t-test

standard.error<-sd(diffs)/sqrt(n)

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

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

paired.t.test <- t.test(diffs,alternative="two.sided",conf.level=0.95)

df.2.sample.t.test <- satterthwaite(sd.1, n.1, sd.2, n.2)

2.sample.t.test <- t.test(variable~as explained by, data=filename, alternative=””, conf.level= )

#R subtracts alphabetically!

boot.means.diffs <- numeric(n)

for (i in 1:n) { #rwo sample t-test

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

resample2 <- sample(date2, length(data2), replace=T)

boot.means.diffs[i] <- mean(resample1) – mean(resample2)

}

2.test.bootstrap.answer <- quantile(boot.means.diffs, c(lower p, upper p))

perm.mean.diffs<-numeric(n)

for (i in 1:n) { #permutation test (when CLT is not satisfied, see if actual sample mean diff is in this range)

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

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

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

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

}

permutation.test.p.value <- sum(abs(perm.mean.diffs)>=abs(x.bar.yes-x.bar.no))/n

wilcoxon.rank.summary <- wilcox.test(dependent.variable~independent.variable, data=filename, alternative=" ")

what.order.R.considers.the.grouping.variable <- levels(filename$groupingvariable)

reordered.grouping <- factor(filename$groupingvariable, levels=c("first group"," second group"))

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

get rows and columns: dim(filename)

get first row: head(filename)

get summary: summary(filename)

get what kind of variable each is: str(filename)

get just one column: filename$columnname

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

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

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

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

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

power.t.test(n=#in each group, delta=diff, sd=sigma, sig.level=, power=, type=“”, alternative=“”)

#Replace whichever variable you want to solve for with “NULL”

kruskal.test(dependent.variable~independent.variable, data=filename)

#plot best fit line

Linear.regression.model <- lm(y~x, data=filename)

str(linear.regression.model)

abline(linear.regression.model, col=”color name”)

natural.log <- log()

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

traditional.backward(max model) #default is 0.1

exhaustive.search(filename$y.var, filename[,min.col:max.col]) #adj R^2 is R with penalty for complexity

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

are multiplied together

plot(y.var~x.var, data=filename, xlab=””, ylab=””, type=””)

points(y.var~x.var, data=filename, pch= )

legend(“position.name”, legend=c(), col= , pch= c())

anova(filename) #more in babysfirstregression18 script

diag.plots(filename)

predict(filename, newdata=list(x.var=what.to.predict), interval=”confidence” or “prediction”, level=conf.level)

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

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

data.without.NA <- filename[!is.na(filename$col.name),]

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 in the frame?

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

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

shaffer.summary <-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

read.cvs(“file.directory”)

read\_cvs(“file.driectory”)

change.col.names.to.lower.case <- tolower(colnames(file.name)

headings <- head(file.name)

summary <- summary(file.name)

view dataset <- View(filename)

see.variable.type.of.each.column <- str(filename)

plot(y~x, data=file.name, type= “”, xlab= “”, ylab= “”, main= “”) #type n=empty graph

points(y~x, data=file.name, pch= , col=”color”, cex=size) #numbers or letters for pch (type)

legend(“location”, legend=c(“category1”, “category2” ), pch= c( , ), col=c( , ))

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)

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

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

defined.data.and.aesthetic <- ggplot(data=file.name, aes(x= , y= , colour=some.variable)

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

geom\_smooth(method=”lm”), xlab= , ylab= ) #some things you can do as a shortcut

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

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

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= “”), str(cor.test(x, y))

method= “pearson”

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

dataEllipse(x, y, levels=c(0.5, 0.95)) #draws ellipse onto data

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

points(burrows~totmass, data=LS, pch=19, col=(cooks.distance(lm.model.ls)>0.8)+1) #for 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)

4-D plot, interaction plot: lab video 2-9-2019 (12?)

-Slopes at bottom of the output: p-value 🡪 does adding this make a significantly better model

from the one where all levels were together?

#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

power.t.test(n= , delta= , sd= , sig.level= , power= , alternative=”one.sided”/”two.sided”)

#provide four, it gives you the fifth (n is per group)

#if sd is different between groups, use the geometric mean between the two

K %\*% t(K) #K is a matrix of contrasts coefficients (linear, one side=0), just diag=orthogonal

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

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

Fixed x random factorial possible approaches

\*The ordinary way

\*Fix by transformation (transforming y toward normality)

\*Fix by lme

\*Fix by Monte Carlo

\*Look at variance components

remove.na.from.one.variable <- filename[!is.na(filename$col.name)]

remove.all.na <- na.omit()

Series.if.evenly.spaced.x <- seq(min, max, interval)

Pl0t curved lines <- use lines command on x seq and associated predicted ys

Assumptions

t-test: CLT must apply

paired t-test: differences must satisfy CLT

two-sample t-test: both datasets must satisfy CLT

bootstrap: CLT violated

permutation test: CLT violated

wilcoxon test: CLT violated, two distributions have the same shape, want to know about medians

Ho: distributions are the same

Ha: distributions are the same except one is shifted in relationship to the other

Kruskal-Wallis test: null=groups are identical; alt=same distribution, shifted mean; >=five in each group

ANOVA: normality of residuals (for each level separately), equality of variances

t-test for correlation (Pearson’s?): x and y have to bivariately normally distributed (otherwise use

Kendall’s tau or Spearman’s rho)

Linear regression: x measured without error, linear model is the correct model (not curved), for any

given x, the y’s are independent and normally distributed, large (settle for qqnorm(residuals) is

normal), homoscadicity (if not met you can transform with log(y))

Regression analysis: a linear model is the correct model, x’s are fixed and measured without error,

residuals epsilon are independent and normally distributed at each value of x, homoscedasticity

ANOVA: linear model is correct model (ignore), predictors measured without error (ignore), residuals

independent of one another and normal for each level, variance equal for each factor level

Leverage point: extreme in the x direction

Influential point: high cook’s distance, regression changes a lot when point is removed

spline.1<-gam(burrows~s(totmass),data=LS)

summary(spline.1)

if x is measured w/ error…

bootstrap if non-normal/heteroscadistic

curve regression

model II regression

Two-way anova[random(fixed)?]: normality of aggregated data for treatment (test each hypothesis separately), use entire dataset for the nested variable

Random(random): homoscadicity at upper level, homoscadacity at secondary level, normality at secondary level

Sum of squares: type I 🡪 hierarchy, type II 🡪 orthogonal contrasts (use polynomial or helmert contrasts, even when you intend to do a planned comparison)

Multiple regression model steps:

# 0. Read in your data, and make a working copy. That way, if

# you change your mind about a transformation, you can restore

# the affected column easily from the original.

# 1. Look at a pairs plot. Are there outliers in the x's that

# could be fixed by transformation? Are there outliers that

# are errors and should be discarded? Do any of the x's look

# like they have a straight or curved relationship with y?

# Does it look like there might be collinearity among the x's?

# Make any changes to your dataset (transformations, adding

# squared terms, etc) called for by your analysis.

# 2. Use R's summary statement to look for missing data. Remove

# missing data from your y-variable, but not your x's (yet).

# 3. Build the minimal model and the full model, and look at

# the VIFs.

# 4. Use exploratory techniques--forwards/backwards regression,

# exhaustive search=dredge--to suggest model ideas.

# 5. Pick two to four different models to evaluate. Look at

# diagnostics and summary information. Can any of them be

# eliminated? Are more complex models really better than

# simpler models?

# 6. Make a choice, and plot the residuals against each x in the

# dataset. If you see a consistent pattern for some x, then

# you can probably improve your model by adding that variable.

# 7. Interpret the results.

Test for orthogonality: multiply together each column of coefficients, add them together

If 0 🡪 orthogonal (if you have three or more comparisons, do each pair separately)

If orthogonal: summary(chickwts.glht.2, test=adjusted("none"))

If not: summary(chickwts.glht.2, test=adjusted("free"))

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

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

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

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

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

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

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)

Random additional notes

To determine if quadratic is better than linear🡪 null is that B2=0

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

etc) #make sure everything going in is positive

Curating data steps: transfer data into spreadsheet, include metadata, storing and curating (CSV will

never not be unreadable), check data for errors (sort, look for min and max, boxplots)

Transforming data: logs can fix nonnormality (issues with this… doesn’t answer the question we want),

may be better to do bootstrap

log10/nat log: make right skew more normal, count data 🡪 continuous, failures in homoscad

sq roots: count data

arcsin-sq root: make proportional data more normal

reciprocals: for rates per unit time/unit area

box-cox:

library(MASS)

model <- lm(y~gp, data=made.up)

boxcox(model) #pick a number near the middle and choose a lambda, raise y to that

power)

ANOVA: to see if there is a difference between groups when there are two or more groups

Kendall’s tau<Spearman’s rho<r (does not mean anything about anything though)

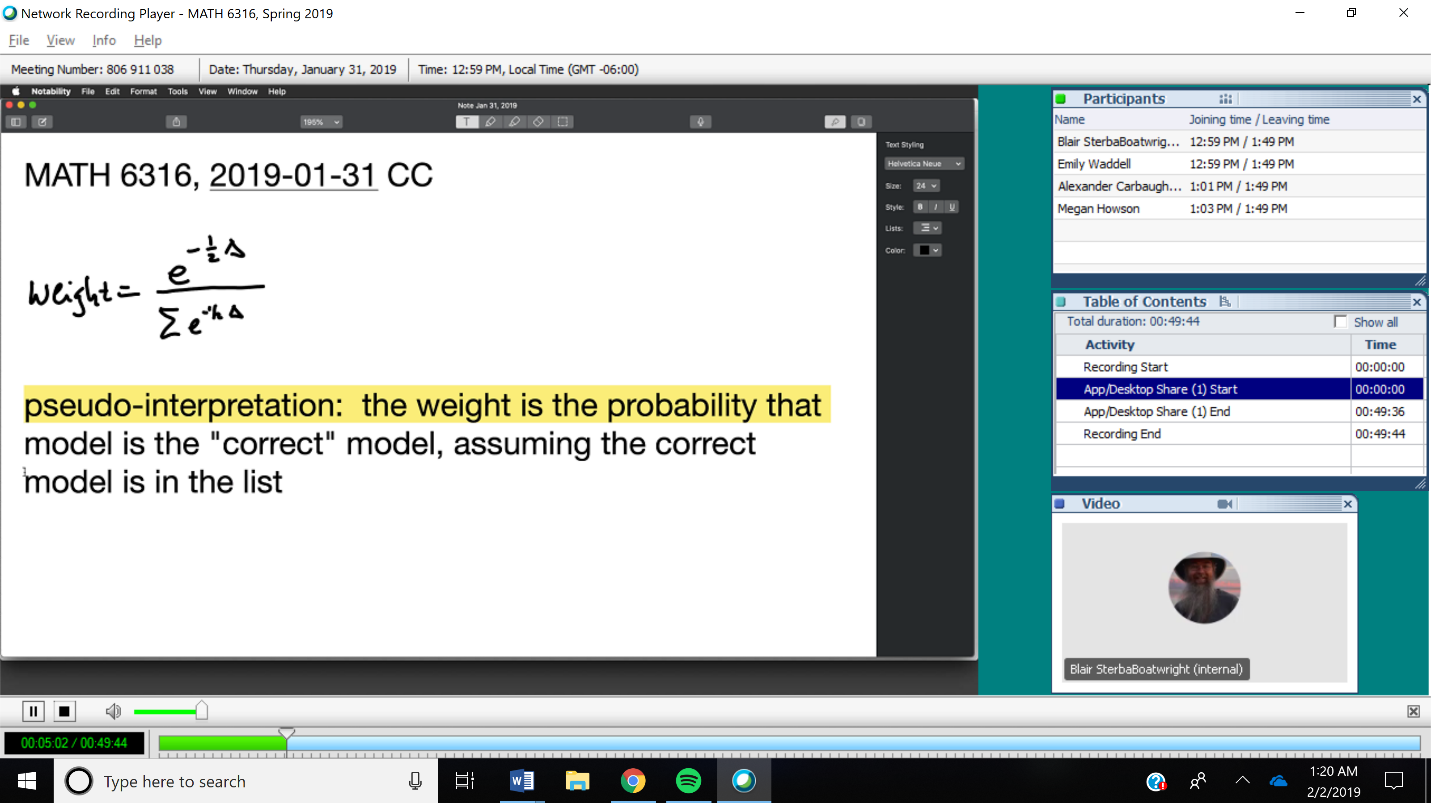
Model II 🡪 the slope is std(y)/std(x) [impossible to have a slope of zero, no hypothesis test]

-Null: slope=1

Bootstrap for regression: two-dimensional bootstrap (last lab of chp5)

-Use with model II, when homoscadicity/normality are not achieved

Splines 🡪 edf: roughly equal to the algebraic degree/order/complexity

Weight 🡪 probability that our model is the “correct” model, assuming the “correct” model is on the list

Definitions

Heteroscedasticy: “trumpeting”

AIC: better models have smaller AIC, no absolutes, just comparison #if <2 difference, they are both fine,

if >10 get rid of the worst, look at the top ones and see which variables frequently show up

Factor: categorical variable

bivariate normality: points lie in elliptical regions around the center (mean x and y). Also any cross

section (any single x value or any single y value) gives a normal distribution

Spearman’s rho: take the actual measurements in x and y, replace with its rank within the dataset, and

then use Pearson’s correlation to compute correlation

Ordinal data: ordered categories without numerical values (like grades)

Monotonicity: tendency to always decrease or always increase going left to right (measured by Kendall’s

tau or Spearman’s rho)

model II regression: minimizes the sum of the area of the triangles

fixed factor: experimenter is specifically interested in comparing the effect of different levels of the

factor; the factor levels were specifically chosen

random factor: levels thought of as a sample of all possible levels; we don’t really care about comparing

different levels of the factor

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