INPUT and OUTPUT

Open data up that’s in your finder.

read.table(file.choose())

read.csv(“filename”,header=TRUE)

oil<-read.csv("crude-oil-imports.csv",header=TRUE)

read.txt(“filename.txt”,header=TRUE)

fix(a) gives you a new window to look at the data

from a library

library()

data(“CommuteAtlanta”)

assign column names

colnames(fly)<-c("winglength","wingwidth","3rdpalplength","3rdpalpwidth","4thpalpwidth","lengthantenna12","lengthantenna13","species")

DATA CREATION

c(…)

matrix(x, nrow, ncol, byrow=TRUE)

**dim(b)<-c(2,4) (shape a vector into a matrix)**

**ls() List**

**rm() remove**

data.frame()

cbind()

factor(x,levels=)

rep(x,times)

rep(c,each=3)

rep(c,time=4)

rep(c,each=3,length=16)

seq(from,to)

seq(from,to,dividing) ie seq(2,4,.1)

cut(lengthOfData,4(number of things you want it cut into)

given.depth1 = co.intervals(quakes$depth,number=4,overlap=.1) Create invervals

names(x)<-c(“one”,’two’,”three”)

if else statements

if (x < 0) {

print("Negative number")

} else if (x > 0) {

print("Positive number")

} else

print("Zero")

SLICING & EXTRACTING, dereferencing

X[n]

X[-n]

X[“name”]

X[c(1,4,2)]

subsetting

early=subset(gvhd10,Days==c(-6,0,6))

MATH

Round()

Sum()

Mean()

Sd()

Table()

quantile(x,.05) get the 5th percentile.

A **percentile** (or a **centile**) is a measure used in [statistics](https://en.wikipedia.org/wiki/Statistics) indicating the value below which a given [percentage](https://en.wikipedia.org/wiki/Percentage) of observations in a group of observations fall. For example, the 20th percentile is the value (or score) below which 20% of the observations may be found.

MATRIX MATH

t(b) Transpose

%\*% multiply

diag(3) Make an identity matrix

diag(id)<-c(4,8,10) assign numbers to the diagonals of an already exisiting matrix

det() determinant

solve() inverse of a matrix

EDA:

Head(data)

Pairs(data)

plotting

dev.off() #turns off the changes you made to the plot space

par(mfrow=c(1,3))

plot(x-axis,y-axis)

plot(density(whatever)

plot(density(

abline(h=median(t)) horizonal line

abline(v=1) vertial line

col=”blue”,”red”,”beige”,”green”,”orange”

plot(x,y,xlab=””,ylab=””,main=”main title”,pch=2,cex=1)

pch=plot character

0=squares

1=circles

2=triangles

3=+

4=x

5=diamond

6=invereted triangles

Plot different colors:

plot(exposure$INHIBIT,exposure$UVB,col=c("red","blue")[DEPTH])

plot(age,survive,xlim=c(15,70),ylim=c(-1.5,2.0),main="survive v.s. age") #set the x and y axis

legend("topleft", legend = c("N= 10 and 10","N=30 and 50"), lty = 2,col=c(2,4))

identify() Lets you click on points on a plot, once you hit escape it’ll give you the observation number.

Points(x,y,pch=,cex=) plot a single point on the graph

hist()

abline(v=mean(x)-2\*sd(x),col=”purple”)

contour(x,y,f) creats a contour plot of 3d data

image(x,y,f) very similar to contour but colored, heatmap

persp(x,y,f,theta=30,phi=20) creates a 3d plot,theta and phi rotate the plot

#dotplots (1d scatters)

#These are ugly and youd never publish

#but they useful for looking at data

stripchart(mpg)

stripchart(mpg,method='stack')

dev.off()

cyl2=as.factor(cyl)

stripchart(mpg~cyl,method='stack')

#bar charts from counts

barplot(BOD$demand,names.arg=BOD$Time)

dev.off()

barplot(BOD$demand,names.arg=BOD$Time,horiz=TRUE,xlim=c(0,20))

dotchart(BOD$demand,labels=BOD$Time)

#pie

pie(BOD$demand,labels=BOD$Time)

par(mfrow=c(1,3))

barplot(BOD$demand,names.arg=BOD$Time,horiz=TRUE,xlim=c(0,20))

dotchart(BOD$demand,labels=BOD$Time)

pie(BOD$demand,labels=BOD$Time)

#You can't compare magnitudes.

#Pie charts are not recommended

loess(y~x,data=,span=,degree=)fits a curve to a scatterplot,degree is 1 if linear, 2 if quadratic

check for normality

qqnorm(residuals(lm.all))

Apply(data,1forrows/2forcolumns,mean/sd/whatever)

tapply

means=tapply(averageincome,INDEX = Province,mean)

Residual Diagnostics.

check for normality

-qqnorm(residuals(lm.all))

### Studentized residuals v.s Observation

-plot(rstudent(LM),main="Studentized residual v.s. observation")

-abline(h=0)

#influence

lm.influence(result)

### Pearson Residuals v.s. observation

plot(residuals(result,type="pearson"),main="pearson residual plot")

### Deviance Residuals v.s. observation

plot(residuals(result,type="deviance"),main="deviance residual plot")

### Hat Diagonal Plot

plot(hatvalues(result),ylab="H",xlab="Case Number Index")

### Intercept DfBeta Plot

plot(dfbetas(result)[,1],ylab="DFBETA0",xlab="Case Number Index")

### Intercept DfBeta Plot

plot(dfbetas(result)[,2],ylab="DFBETA1",xlab="Case Number Index")

##ECDF [Emperical Cumulative Distribution Function] plot

plot(ecdf(all.res))

###ROC curve (receiver-operating characteristic): analyze the effectiveness of a test or process when there’s 2 possible outcomes for y, particularly used for deciding on a “cut off”

getting test statistics

qt(.95,n-p)

missing values

(do math on a vector and just remove the Nas)

sum(a,na.rm=TRUE)

qqplot

log normal distribution

rlnorm(n,meanlog=0,sdlog=1)

sample(x, size, replace = FALSE, prob = NULL)

sample(from x, take size,

# paired t-test  
t.test(y1,y2,paired=TRUE,) # where y1 & y2 are numeric

for 1 tailed test include **alternative="less"** or **alternative="greater"**

factors as an integer

as.integer(types)

convert quantitative variables into categorial

as.factor()

print up levels of a vector

levels(types)

Pulling specific info out of a test.

Simple tests.

tt=t.test(y1,y2)

tt$statistic

tt$p.value

P-value given a critical value

2**\***pt(**-abs**(**t**),df**=**n-1) [2 sided test]

summaries

aov=aov(y~as.factor(pred))

summary(aov)[[1]][["Pr(>F)"]] #pulls out just the p-value, if you wanted the F statistic you’d do [[“F”]] instead of the p

par(mfrow=c(2,2))

put the odd plot things on

dev.off()

get rid of odd plot things

r markdown

subscripts

$\epsilon$~t~ and $\epsilon$~t-1~, $\epsilon$~t-2~

data generation/simulation

runif(x,min,max)

rbinom(size,n,prob of sucess)

rnorm(n,mean,sd)

creating a function

none <-function(x){result <- x; return(result)}

none(3)

#Create a function called mulp() which multiples two numbers.

#Send 4 and 8 to mulp().

mulp <-function(x,y){result <-x\*y; return(result)}

mulp(4,8)

cor() correlation of whatever

Survivial

Example: 2.2,3+,8.4,7.5+

Stime<-c(2.2,3,8.4,7.5)

Status<-c(1,0,1,0)

[if it’s censored it’s status is 0]

Chi Squared Test:

chisq.test(table)

chisq.test(table)$expected #gives the expected values easy peasy

chisq.test(table, correct=FALSE) ## chisq. test of independence

creating a table

table=as.matrix(table(x,y)

Regression

Linaer regression

Lm=lm(y~x)

Plot a regression line

abline(lm(height ~ bodymass))

fitting a logistic Regression

result=glm(y~x,family=binomial(‘logit’))

using the predict function using an lm model.

Predict(lm object,newdata(this is optional, if you leave it out it’ll give the fitted values))

pulling stuff out of a regression summary

summary(result)

coefficients(result)

confint(result) confidence interval

results$fitted (fitted values)

for loops:

ex1:

for (year in 2010:2015){

print(paste("The year is", year))

}

ex 2: Find the mean for each column in the A matrix

mean=NULL

for (i in 1:3){

mean[i]=mean(A[1:4,i])

}

A= matrix(rnorm(12),nrow=4,ncol=3)

mean=NULL

for (i in 1:3){

mean[i]=mean(A[1:4,i])

}

mean

apply:

apply different functions to a matrix or table

apply(what you want it applied to, 1=rows 2=columns, function you want used.

Ex1:

Get means for columns

apply(A,2,mean)

get means for rows

apply(A,1,mean)

outer(x,y,”^”) Really similar to apply but creates an array with dim(x), dim(y)

quit: q()

Add a picture to r markdown

![Caption for the picture.](/path/to/image.png)

library lattice:

density plot

densityplot(~psa,data=psa.dat,groups=factor(ind),plot.points=F,

boxplot bwplot()

strip plot stripplot()

auto.key=list(columns=2)) library ggplot

needs data in the form of a dataframe

basic scatter

#facetting a basic scatter

#defined by rows

qplot(x,y,data=)+facet\_grid(z~.)

#defined by column

facet\_grid(.~z)

geop\_points()

geom\_smooth() #LOESS fit

geom\_smooth(method=’lm’)

c<-ggplot()

create a barchart

bar<-c+geom\_bar(width=1)

convert to a pie

pie<-bar+coord\_polar(theta=y)

libarary(maps)

mp+geom\_points+boarders(database=’world’,size=0.2, colour=’gray21’,fill=’palegreen’)

creating a table in r markdown

\begin{center}

\begin{tabular}{ |c|c|c|c| }

\hline

Model & $p\_D(\text{effective parameters})$ & number of parameters & DIC \\

1 & `r round(sum(dic1$penalty),1)` & 26 & `r round(sum(dic1$deviance),1)+round(sum(dic1$penalty),1)`\\

2 & `r round(sum(dic2$penalty),1)` & 25 & `r round(sum(dic2$deviance),1)+round(sum(dic2$penalty),1)`\\

3 & `r round(sum(dic3$penalty),1)` & 24 & `r round(sum(dic3$deviance),1)+round(sum(dic3$penalty),1)`\\

4 & `r round(sum(dic4$penalty),1)` & 4 & `r round(sum(dic4$deviance),1)+round(sum(dic4$penalty),1)`\\

\hline

\end{tabular}

\end{center}

interaction plot

interaction.plot(ga$year,ga$sex,log(ga$rate),type="l",xlab="year",ylab="log(rate)",trace.label="sex")

variance stabilizing transformation for percent data:

Note: in order for this to work, your percents have to be in .14 for 14%.

**asinTransform <- function(p) { asin(sqrt(p)) }**

**pAsin <- asinTransform(p)  
plot(p, pAsin, type='l', lwd=2, col='blue', las=1, xlab='p', ylab='arcsine(p)')**