Common STAT 101 Commands for RStudio

Professor Halima Bensmail

1. Example through regression

#Get the dataset by the following R commands

```
install.packages("UsingR")

library(UsingR)

data(father.son)
```

#Then the data frame father.son contains the 1078 observations on 2 variables: fheight (father's height in inches, x) and sheight (adult son's height in inches, y).

#Draw a scatter plot of son's height versus father's height. Does the relationship appear linear?

```
fatherson < as.matrix(father.son)

summary(fatherson)

plot(fatherson,xlab="Father's height (in)", ylab="Son's height (in)", xlim=c(58,78), ylim=c(58,80), bty="l", pch=20).
```

#Fit the simple linear regression of son's height on father's height. What are the estimated regression coefficients, a and b, respectively?

#First let's do it using the equations:

#…….. Write down the equation……..

$$b = \frac{\sum_{i=1}^{i=n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{i=n}(x_i-\bar{x})2} \qquad \text{(slope)}$$

$$a = \bar{y} - b\bar{x} \qquad \text{(intercept)}$$

After that, let's do t using R:

```
m <- lm(fatherson[,2] ~ fatherson[,1])
```

#Add regression line

abline(lm(fatherson[,2] ~ fatherson[,1]), lty=1, lwd=2)

Calculate Pearson correlation coefficient r between father's height and son's height. Perform a proper test to test the null hypothesis $\rho = 0$, where ρ is the population correlation coefficient

we will do this later during this semester

cor(fatherson[,1], fatherson[,2])

Calculate the coefficient determination R^2. What does the R^2 statistic mean?

Draw a residual plot. Are the residuals normally distributed with constant variance?

#The model assumptions of normal distribution and constant variance seem valid based on the

residual plot below.

#What are the estimated means of son's height given that his father's height is 72, 75, 60, and 63

inches, respectively?

2. Draw a sample randomly

```
v<-sample(1:2500, 100, replace=TRUE)
x<-c(rep(1,1500),rep(0,2500-1000))
mean(x[v])
```

Draw 1000 samples each of size 1078 with replacement from the 1078 pairs of father-son heights, then from each calculate the mean find the distribution of the mean.

Let's do it for father height for example.

```
n <- 1078
mfather <- rep(NA, 1000)
for (i in 1:1000){
v <- sample(1:n, n, replace=TRUE)
mfather[i]<- mean(fatherson[,1][v])
}
```

3. QQplot:

help(qqplot)

#not using a software

#Sample question:

#Do the following values come from a normal distribution?

#7.19, 6.31, 5.89, 4.5, 3.77, 4.25, 5.19, 5.79, 6.79.

#Step 1: Order the items from smallest to largest.

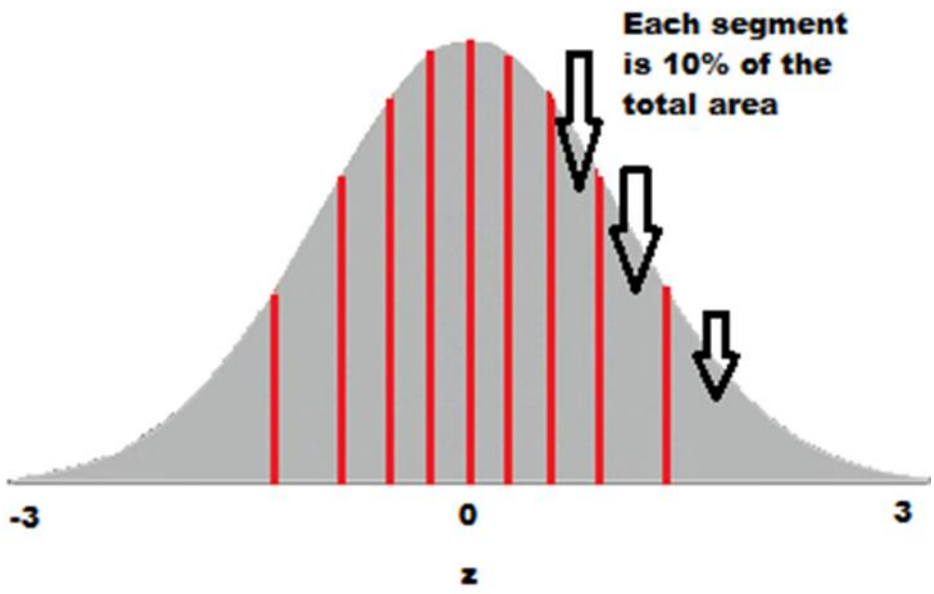
#3.77, 4.25, 4.50, 5.19, 5.89, 5.79, 6.31, 6.79, 7.19

#Step 2: Draw a normal distribution curve.

Divide the curve into n+1 segments.

We have 9 values, so divide the curve into 10 equally-sized areas.

For this example, each segment is 10% of the area (because 100% / 10 = 10%).



#qnorm(0.10)

#qnorm(0.20)

#qnorm(0.40)

#qnorm(0.50)

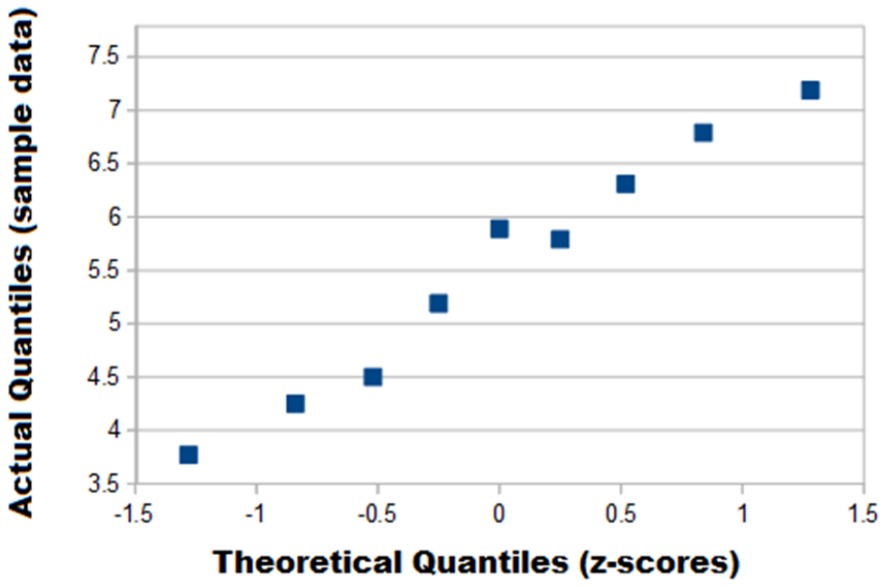
#qnorm(0.60)

#qnorm(0.70)

#qnorm(0.80)

#qnorm(0.90)

Step 4: Plot your data set values (Step 1) against your normal distribution cut-off points (Step 3). I used Open Office for this chart:



#Using R

```
x <-c(7.19, 6.31, 5.89, 4.5, 3.77, 4.25, 5.19, 5.79, 6.79)
x<-sort(x)
y <- c(-1.28, -0.84, -0.52, -0.25, 0, 0.25, 0.52, 0.84, 1.28)
plot (x,y)
```

#using R

```
x <-c(7.19, 6.31, 5.89, 4.5, 3.77, 4.25, 5.19, 5.79, 6.79)
x<-sort(x)
y<-qqnorm(x)
print(y)
qqline(x, col = 2)
y <- rnorm(1000, 68, 2.6)
qqnorm(y)
qqline(y, col = 2)
qqnorm(fatherson[,1])
qqline(fatherson[,1], col = 2)
```

4. One Categorical Variable

```
table(x)
barplot(table(x))
resample(x)
#gives a bootstrap distribution and CI for proportion
#level = confidence level (default: .95)
pnorm(z)
#gives probability in tail of N(0,1) below z
#lower.tail=FALSE for upper tail above z
qnorm(0.975)
#replace 0.975 with desired percentile of N(0,1)
prop.test(count,n, p=null.value)
#normal based inference for a single proportion
#conf.level = confidence level (default: .95)
#alternative = "two.sided","less", or "greater" (default: "two.sided")
chisq.test(table(x))
#goodness of fit test
#p = c(p1, p2, ...) (null values) (default null: all equal)
#simulate.p.value = TRUE (calculates p-value by simulation)
#B = 10000 (number of simulations used) (default = 2000)
chisq.test(c(count1, count2, count3, ...))
#if entering counts from table
```

5. One Quantitative Variable

```
hist(y)
#xlab = label for x-axis
#main = title of plot
mean(y)
#na.rm=TRUE to get rid of NA values
median(y)
sd(y)
summary(y)
resample(y)
#gives a bootstrap distribution and CI for mean
#level = confidence level (default: .95)
pt(t,df)
#gives probability in tail of a t-distribution below t
#lower.tail=FALSE for upper tail above t

qt(0.975, df)
#replace 0.975 with desired percentile of t-distribution
t.test(y,mu=null.value)
#t-based inference for a single mean
#conf.level = confidence level (default: .95)
#alternative = "two.sided", "less", or "greater" (default: "two.sided")
```

6. Two Categorical Variables

table(x,y)

barplot(table(x,y))

#beside = TRUE for side-by-side barplot

#legend = TRUE to include a color legend

mosaicplot(table(x,y))

resample(x,y)

#gives a bootstrap distribution and CI for difference in proportions

#level = confidence level (default: .95)

reallocate(x,y)

#randomization test for difference in proportions

pnorm(z)

#gives probability in tail of N(0,1) below z

#lower.tail=FALSE for upper tail above z

qnorm(0.975)

#replace 0.975 with desired percentile of N(0,1)

prop.test(c(count1, count2), c(n1,n2))

#normal based inference for a difference in proportions

#conf.level = confidence level (default: .95)

#alternative = "two.sided", "less", or "greater" (default: "two.sided")

chisq.test(table(x,y))

#chi-squared test for an association between x and y

#simulate.p.value = TRUE (calculates p-value by simulation)

#B = 10000 (number of simulations used) (default = 2000)

4 One Categorical and One Quantitative Variable

y = quantitative

x = categorical

by(y, x, mean)

```
#na.rm=TRUE to get rid of NA values
by(y, x, sd)
boxplot(y~x)
resample(y, x)
#gives a bootstrap distribution and CI for difference in means
#level = confidence level (default: .95)
reallocate(x,y)
#randomization test for difference in means
pt(t,df)
#gives probability in tail of a t-distribution below t

#lower.tail=FALSE for upper tail above t
qt(0.975, df)
#replace 0.975 with desired percentile of t-distribution
t.test(y~x)
#t-based inference for a difference in means
#conf.level = confidence level (default: .95)
#alternative = "two.sided", "less", or "greater" (default: "two.sided")
t.test(y1,y2)
#if the two categories are given as different vectors of quantitative variables
summary(aov(y~x))
#ANOVA for difference in means
```

8. Two Quantitative Variables

plot(x,y)

#xlab = label for x-axis, ylab = label for y-axis

#main = title for plot

cor(x,y)

#use = "complete.obs" to get rid of NA values

resample(x,y)

#gives a bootstrap distribution and CI for correlation

#level = confidence level (default: .95)

reallocate(x,y)

#randomization test for correlation

pt(t,df)

#gives probability in tail of a t-distribution below t

#lower.tail=FALSE for upper tail above t

qt(0.975, df)

#replace 0.975 with desired percentile of t-distribution

cor.test(x,y)

#t-based inference for a correlation

#conf.level = confidence level (default: .95)

#alternative = "two.sided", "less", or "greater" (default: "two.sided")

8. Regression

```
model = lm(y~x1+x2, data=dataname)
#y = response variable
#x1, x2, ... = any number of explanatory variables
summary(model)
#gives output for the model
plot(model)
#produces multiple plots including a residual plot of residuals versus predicted
values, and a normal q-q plot of the residuals
predict(model)
#gives predicted values for each case
#interval = "confidence" gives confidence intervals for each case
#interval = "prediction" gives prediction intervals for each case

newdata = as.data.frame(cbind(x1 = x1value, x2 = x2value))
#only enter numeric variables this way, for categorical variables, add the values
to newdata with newdata$x3 = "yes" for each categorical variable
predict(model, newdata)
#gives prediction for a new data point
step(model)
#performs stepwise regression
model = glm(y~x1+x2, data=dataname, family="binomial")
#logistic regression
```

9. Subsetting

```
subset(dataname, !is.na(x))
#the data set "data", but only cases for which x is not NA
subset(dataname, x=="levelA")
#data "dataname", but only cases for which x is equal to "levelA"
x[!is.na(x)]
#the variable x, but only cases for which x is not NA
y[!is.na(x)]
#the variable y, but only cases for which x is not NA
x[x < 30]
#the variable x, but only cases for which x is less than 30
x[x != "levelA"]
#the variable x, but only cases for which x does not equal "levelA"
droplevels(x)
#drops empty levels if you have removed all the cases from one level
```