R for Math/Stat 571 A – part 1
Sandy Dall'erba

Based on:
Wright, D. B. & London, K. (2009). *Modern regression techniques using R: A practical guide for students and researchers*. London: Sage.
Marques de Sá J.P. (2007), *Applied Statistics Using SPSS, STATISTICA, MATLAB and R*, Springer; 2nd edition

1) Brief introduction to R

Download R on your machine from the following website:
http://cran.r-project.org/

This will give you both a very powerful statistics language and statistics package. It will allow you to do most of the statistics you want but not all. Additional packages (with additional codes) can be obtained online. I recommend a package called **foreign** that allows you to import and export data between R and other statistical packages and do most of the statistical analysis we need.
Type:
install.packages("foreign", dependencies=TRUE)

And chose the server close to where you live. It will upload the package that you can now access even if you are not connected to the internet. The package needs to be activated as follows:
library("foreign")
this will give you access to a large number of functions that are useful to us here.

If you do not add "dependencies" in the first line of code, you may get the following message
Loading required package: xxxxxxxxxxxx
It obliges you to install the xxxxxxxxxx package by **install.packages(" ")** and to open it with **library(" ")**

Useful link to get access to these packages:
http://cran.r-project.org/web/packages/

Keep also in mind that some workbooks have been automatically downloaded with R. These are useful for introductory steps and basic functions.

R works by applying functions to objects. For the sake of practicing, we will use the function mean, but first we have to create a variable. A variable is a set of similar objects. We can create a variable by assigning a list of values to a variable name. We do so by using the following characters: **<-** or **->** and the letter c which stands for concatenate (telling that these four numbers are a set)

Example:
scores <-c(5,6,7,8) or c(5,6,7,8) -> scores

and then type:
scores

it returns:
[1] 5 6 7 8

scores is a numeric variable, which is a type of object. Be careful, R is case sensitive !

mean(scores)
[1] 6.5

The [1] in front of the results is because some functions produce several pieces of information so their parts are labeled. You can use functions in creating new variables. For example, let's create a variable which is the difference between each value and the mean of the variable:

residscores <-scores-mean(scores)
residscores
[1] -1.5 -0.5 0.5 1.5

Note that you can insert comments (that are not commands) by typing #. Such as:
```
#  Chapter 1
```

2) Concrete example:

Step 1: convert the xls. data in txt file Tab delimited. This creates a "eu.txt" file that will be called in the next function. Under excel, click on save as, new document, text tab delimited.

data <- read.delim ("C:\\Documents and Settings\\Sandy Dall'erba.SBS-SBS1679\\Desktop\\Math 571A - Fall 2012\\R\\eu.txt", header=T) *** read.delim is necessary for tab delimited/ comma delimited files***header=T means the first line in the data are the variable names.
attach(data) ***this is to keep the data always available to the program and it overwrites any data that may have the same name**
names(data) ***this is to verify the name of the headers***

[1] "NAME" "CODE" "POLYID" "core" "periphery" "NATION"
[7] "CNTRYNAME" "GDP89" "GDP90" "GDP91" "GROWTH" "MANUF80"
[13] "MANUF89" "AGRI" "XCOO" "YCOO" "INV"
summary(data) ****it reveals the min, max values as well as quartile values***
data[10] or data$GDP91 ***it displays the name and all the values of the data in column 9, i.e. the data of the variable GDP91****

GDP91

1 10.080428

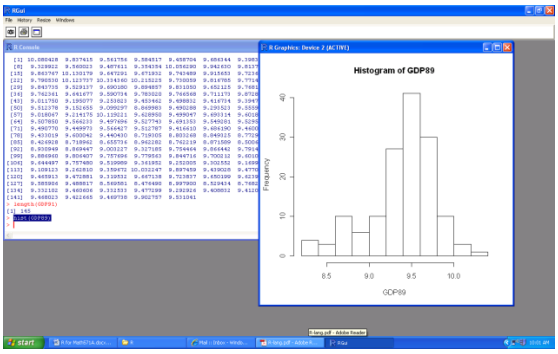
2 9.837415

3 9.561756, etc…

var(GDP89) ****variance***
[1] 0.156465
sd(GDP89) ***standard deviation***
[1] 0.3955566

length(GDP91) ***allows you to see the length of the data ***
[1] 145
NCOL(GDP89) ***number of columns***
[1] 1
NROW(GDP89) ***number of rows***
[1] 145
dim(GDP89) ***dimension : always NULL for a vector***
NULL
dim(data)
[1] 145 17

hist(GDP89) ***will give the histogram of this variable****



Try also hist(GDP89, breaks=18)

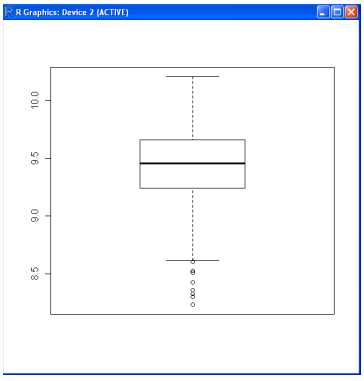
Use the "print screen option" of your keyboard to copy/paste images + the crop option if necessary.

Identifying the presence of outliers can be done with a boxplot:
boxplot(GDP89, range=1.5)
*** this will draw a boxplot corresponding to the following formula***

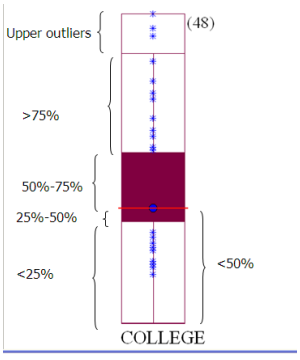
It shows the first and the fourth quartile of the distribution as well as outliers. An outlier is an observation of which value is more than a given multiple (1.5 here) of the interquartile range (the difference in value between the 25% and 75% observation) above or below respectively the value for the 75th percentile and 25th percentile :

$Q_1 - 1.5*(Q_3 - Q_1)$ > *outlier* *outlier* > $Q_3 + 1.5*(Q_3 - Q_1)$

The thick line in the middle is the median
The (lower) thin line is the hinge that indicates which observations are outliers.



Other example



boxplot.stats(GDP89)
$stats
[1] 8.614118 9.240431 9.455740 9.659238 10.205263

$n

[1] 145

$conf
[1] 9.400788 9.510692

$out
[1] 8.604271 8.359129 8.528616 8.328944 8.305051 8.233559 8.430288 8.510933

Explore the meaning of this function with:
help("boxplot.stats") or call the Help menu on the top

You can compare these values with:
summary(GDP89)

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
8.234	9.240	9.456	9.386	9.659	10.210

3) Basic regression

The basic regression function is lm, which stands for linear model. The function to use for a regression is written as: lm(Y~X) for the following regression: $Y_i = \beta_1 + \beta_2 X_i + e_i$, or, in the case of growth across European regions: $(growth)_i = \beta_1 + \beta_2 (initial\ income)_i + e_i$

We need to define our variables first:
y <- GROWTH
x <- GDP89

Running a regression will create a lm.object. Let us run 2 regressions first:
reg1<-lm(y~x)

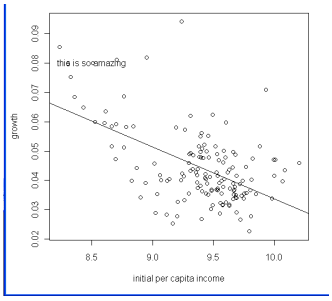
If you want the results associated to the regression 1:
summary(reg1)

the output will be:

```
Call:
lm(formula = y ~ x)

Residuals:
      Min        1Q    Median        3Q       Max
-0.023239 -0.006352 -0.001562  0.005764  0.046928

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.210570   0.021505   9.792  < 2e-16 ***
x           -0.017679   0.002289  -7.723 1.81e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01087 on 143 degrees of freedom
Multiple R-squared: 0.2943,     Adjusted R-squared: 0.2894
F-statistic: 59.64 on 1 and 143 DF,  p-value: 1.811e-12
```

How do you interpret the value of the slope coefficient?
How do you interpret the R squared?

Next we will plot the data.
plot(x,y,xlab="initial per capita income", ylab="growth")
note that you enter the name of the value on the x-axis first, y-axis second
it will name the labels
abline(reg1) ***this places the regression line into the scatterplot. It is only for straight lines****



It is sometimes useful to add text to a graph. We use the function text:
text(8.5, 0.08, "this is so amazing")
add some text in the graph and locate it according to the x and y coordinates*

4) Normality tests

The first step consists in calculating the residuals of the previous regression. We will then compare their distribution with the distribution of normally distributed residuals.

4.1. Find the residuals
$\hat{Y}_i = \hat{\beta}_1 + \hat{\beta}_2 X_i$
ypredihat<-0.21057-0.017679*x

$Y_i = \hat{Y}_i + \hat{u}_i$

uhat<-y-ypredihat

4.2 Create normally distributed residuals

x2<- rnorm (mean=10, sd=10, n=145)

****we create a random variable x2 with n=145 cases (just like our number of observations, but it is NOT necessary), 10 as a population mean and 10 as the standard deviation

Now we can define y:

set.seed = 121 ***this allows us to specify a seed, i.e. a single value, interpreted as an integer. This is to make sure that the random variables we all generate are not too far apart****

y2<-x2+rnorm(mean=0, sd=10, n=145) ***this is to generate some random y variables***

reg2<-lm(y2~x2)

summary(reg2)

since the intercept is not significant, we use

uhatnorm<- y2-0.95501*x2

4.3 Skewness and kurtosis

Now, before we look at the skewness and kurtosis of the residuals, we need to load another package. The two commonly used are fBasics and e1071.

install.packages("fBasics")

or

install.packages("e1071")

Let's use the first one which obliges us to select the location of a server. We will use the function skewness which provides the value of the sample residuals skewness, not the estimate of the population residual skewness.

library(fBasics)

skewness(uhat) 0.9066632	skewness(uhatnorm) -0.02025387
kurtosis(uhat) 2.218644	kurtosis(uhatnorm) -0.4718078

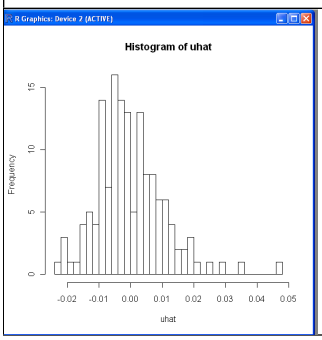
Reminder:

A normal distribution has a skew of 0 and a kurtosis of 3.

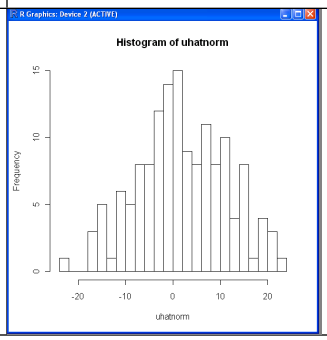
- The positive skew of uhat means that the tail on the right side of the distribution is longer than the left side and the bulk of the values (including the median) lies on the left side of the mean.
- Here, the "kurtosis" of uhat is below 3 (the normal distribution has a kurtosis of 3) which means uhat has a lower, wider peak around the mean and shorter tails

4.4. histogram of residuals

hist(uhat, breaks=28) ***for 29 intervals***

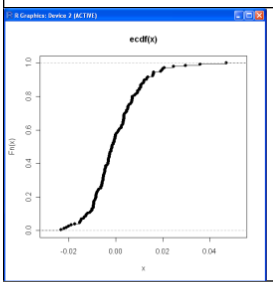


hist(uhatnorm, breaks=28)

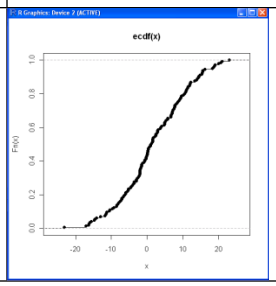


4.5. Normal probability plot (ecdf:empirical cumulative distribution function)

plot.ecdf(uhat)



plot.ecdf(uhatnorm)



This plot shows the shape of the probability density function of a random variable. The x-axis shows the value of uhat. The y-axis shows the cdf *F(x), i.e.* the proportion of X values that are less than or equal to *X*. If a variable were normally distributed, the normal probability plot would be approximately a straight line.

If you want to see several figures at once, choose the following and then type your codes for the plots

par(mfrow=c(1,2)) ***it means 1 row, 2 columns***

4.6. Run the JB test

install.packages("tseries")

library(tseries)

jarque.bera.test(uhat)	jarque.bera.test(uhatnorm)
Jarque Bera Test data: uhat X-squared = 52.0039, df = 2, p-value = 5.099e-12	Jarque Bera Test data: uhatnorm X-squared = 1.1616, df = 2, p-value = 0.5594

The JB test is not significant for the normally distributed residuals, so H0 that JB=0 cannot be rejected.

Note: there are other tests for normality:

- Kolmogorov-Smirnov goodness of fit test
- Lilliefors test for normality
- Shapiro-Wilk test for normality