GU4205/5205-Linear Regression Models-Introduction to R Basics, Base R

Authors: Dr. Banu Baydil, Dr. Ronald Neath, Dr. Gabriel J. Young

Fall 2022-Lab1a

Section 1: Getting Started with RMarkdown

In RMarkdown, "R code" is written in a **code chunk** and "regular text" is written outside of the **code chunk**. This whole document is structured in this format. Also see the RStudio template for RMarkdown by opening a new .rmd (or markdown) document. In this document text written within two double starts appears in boldface such as **Text** in the .doc word (or .pdf) documents created from this document by selecting Knit Word option (or the Knit PDF option) in the Knit menu at the top of the Markdown document panel of RStudio. Note that Knit PDF might not work in some of the computers on campus labs. However, once created, you can save the word document as .pdf document from within the Word program.

To begin a **code chunk**, write three apostrophes followed by {r}. To end a code chunk, write three apostrophes. The following example displays the mathematical constant pi rounded to 5 decimal places. If you select the code in a chunk, the green 'Run' arrow on the top-right (above the markdown file or above the chunk) allows you to run the code in its respective chunk. Also by selecting **Insert Chunk** in the Chunks menu top-right above (the markdown file) and you can insert a code chunk automatically. The three apostrophes followed by {r} at the beginning of the code chunk, and the three apostrophes at the end of the code chunk, do not show in the .doc or .pdf documents created from this markdown document.

```
round(pi,5)
```

```
## [1] 3.14159
```

Section 2: Assigning Variables and Commenting

Use the symbol **<-** or **=** to assign a variable. Once assigned, the variable will exist in the **R workspace** as seen in the upper right-hand corner of RStudio. To comment, use the symbol #. A comment tells R that no code will run on the right side of the pound sign. See the example below.

```
# This is a comment

# Next we assign a variable
a <- c(1,2,6,8)
```

The concatenation function **c()** used above is one way to build a vector in R. You can also use the colon notation or the function **seq()**, i.e.,

```
# These are the same vectors
b <- seq(10,15,by=1)
c <- 10:15
d <- c(10,11,12,13,14,15)
b
```

```
## [1] 10 11 12 13 14 15
```

```
c
```

```
## [1] 10 11 12 13 14 15
```

```
d
```

```
## [1] 10 11 12 13 14 15
```

Section 3: Read in .csv Files and Dataframes

In this section we will run the UN11.csv dataset from STAT GU4205/5205 course textbook. You can get the dataset from the course textbook website.

First! Before running the code chunk below, make sure the csv file **UN11.csv** is in the SAME folder as this markdown file. Consequently R will know how to retrieve the .csv file from your computer.

The Base R function **read.csv()** is used to read in a csv file. Note that the resulting object, assigned as **UN11_data**, is a **dataframe.** In Base R, a **dataframe** is the most common way for R to organize and store a dataset. A dataframe allows for each column to have a different variable type and is organized in a rectangular format. The rows of a dataframe are its observations and columns of a dataframe are its variables. Please see the below example which uploads the UN11 dataset.

Before you upload your data file from Excel, you will need to remove the rows (observations) that have missing or incorrect values in your excel dataset (i.e. manually 'clean' the data set), and then if it is not already in .csv format, you will need to save your dataset as a .csv file in Excel. Afterwards, you can upload your dataset by modifying the below command accordingly. Note that all commands are case sensitive, so you will need to be careful when typing them.

```
UN11_data <- read.csv(file="UN11.csv",as.is=T,header=T)
```

A few basic operations follow. The function **dim()** shows the dimension of the dataframe.
The function **head()** shows the first 6 cases of the dataframe.
The function **names()** gives the variable names of the dataframe.

```
head(UN11_data)
```

```
##             X     region  group fertility   ppgdp lifeExpF pctUrban
## 1 Afghanistan       Asia  other     5.968   499.0    49.49       23
## 2     Albania     Europe  other     1.525  3677.2    80.40       53
## 3     Algeria     Africa africa     2.142  4473.0    75.00       67
## 4      Angola     Africa africa     5.135  4321.9    53.17       59
## 5    Anguilla  Caribbean  other     2.000 13750.1    81.10      100
## 6   Argentina Latin Amer  other     2.172  9162.1    79.89       93
```

```
dim(UN11_data)
```

```
## [1] 199   7
```

```
names(UN11_data)
```

```
## [1] "X"         "region"    "group"     "fertility" "ppgdp"     "lifeExpF"
## [7] "pctUrban"
```

Extracting a single column from a dataframe

To extract a column from a dataframe, we use the notation **dataframe$variable**. See the below example which extracts per person gdp **ppgdp** and **fertility** and stores them as the variables **UN11.ppgdp** and **UN11.fertility**. We also look at the first 10 cases of each variable using the **head()** function.

```
UN11.ppgdp <- UN11_data$ppgdp
head(UN11.ppgdp,10)
```

```
##  [1]   499.0  3677.2  4473.0  4321.9 13750.1  9162.1  3030.7 22851.5 57118.9
## [10] 45158.8
```

```
UN11.fertility <- UN11_data$fertility
head(UN11.fertility,10)
```

```
##  [1] 5.968 1.525 2.142 5.135 2.000 2.172 1.735 1.671 1.949 1.346
```

Section 4: Descriptive Statistics on Numeric Variables

Consider the variable **UN11.fertility**. Some common R functions used for descriptive statistics follow below:

```
# Sample mean
mean(UN11.fertility)
```

```
## [1] 2.761383
```

```
# Sample median
median(UN11.fertility)
```

```
## [1] 2.262
```

```
# Six number summary
summary(UN11.fertility)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.134   1.754   2.262   2.761   3.545   6.925
```

```
# Sample variance
var(UN11.fertility)
```

```
## [1] 1.7945
```

```
# Sample standard deviation
sd(UN11.fertility)
```

```
## [1] 1.339589
```

```
# Sample range
range(UN11.fertility)
```

```
## [1] 1.134 6.925
```

```
# Percentiles (25th and 75th)
quantile(UN11.fertility, probs=c(.25,.75))
```

```
##    25%    75%
## 1.7535 3.5445
```

Section 5: Factor Variables, Categorical Variables

R prefers to store categorical variables as **factor** variables. To do this, convert categorical variables into factors using the function **factor()**. Note that the **read.csv()** function used earlier will default to converting all categorical variables into factors if we do not use the argument **as.is=T**.

```
UN11.region <- factor(UN11_data$region)
```

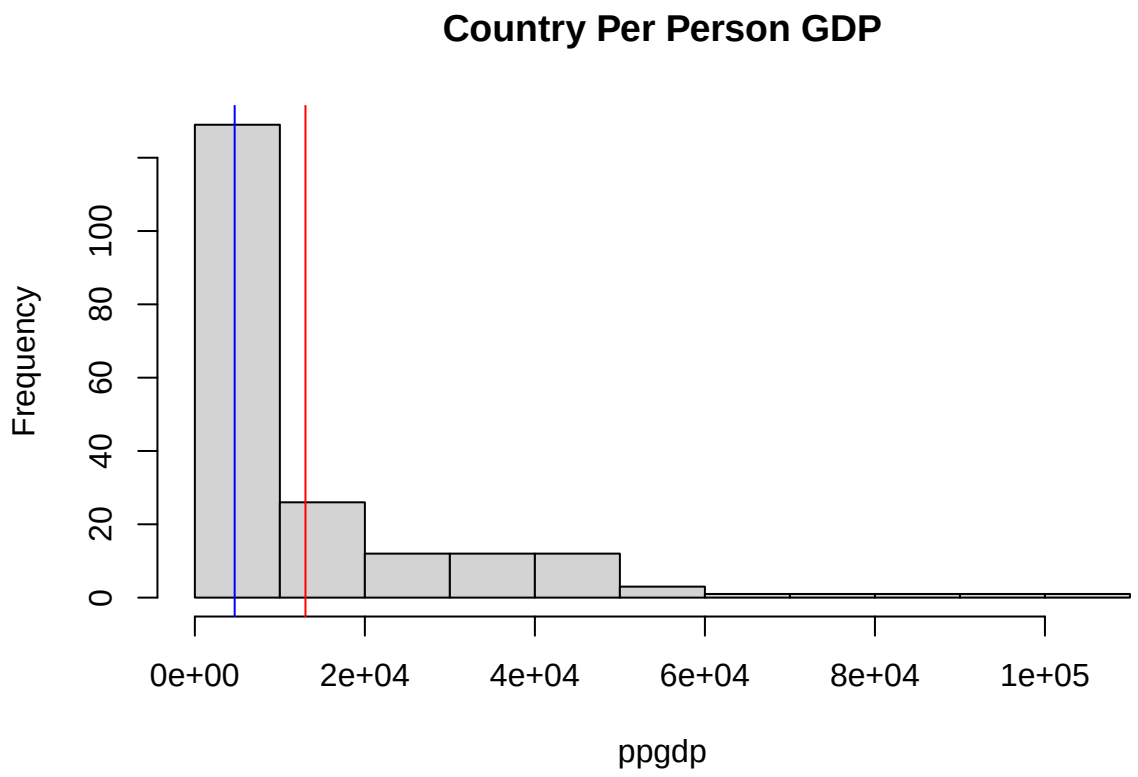
Section 6: Basic R Graphics

Histograms

A histogram is the most common graphic used to display a quantitative variable. The code below shows a histogram of the variable **UN11.ppgdp**. Note that the title and x-axis have been updated using the options **xlab** and **main**.

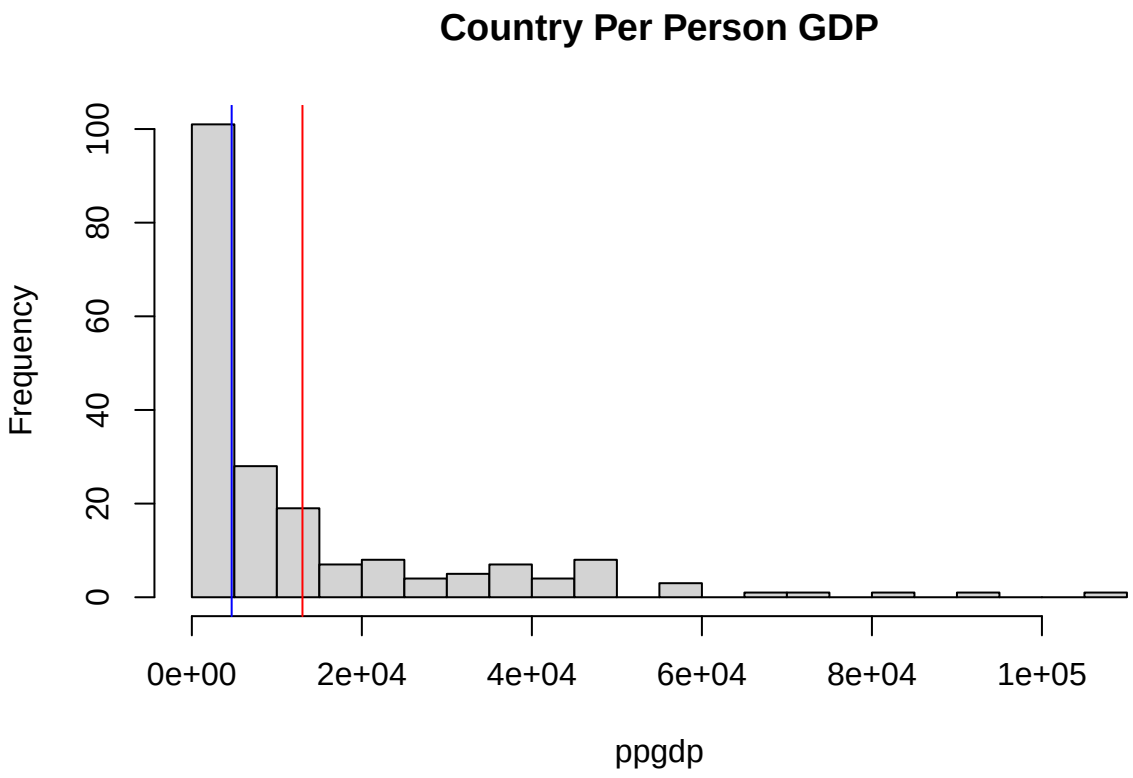
We can also draw vertical lines denoting the mean (red) and median (blue) per person gdp using the abline() function.

```
hist(UN11.ppgdp,main="Country Per Person GDP",xlab="ppgdp")
abline(v=mean(UN11.ppgdp),col="red")
abline(v=median(UN11.ppgdp),col="blue")
```



We can also change the amount of detail in the histogram using the **breaks** option as shown below. Note that the number you set as breaks is not exactly equal to the number of bins in the histogram. However, the smaller this number is the less detailed the histogram is, and the larger this number is the more detailed the histogram is.

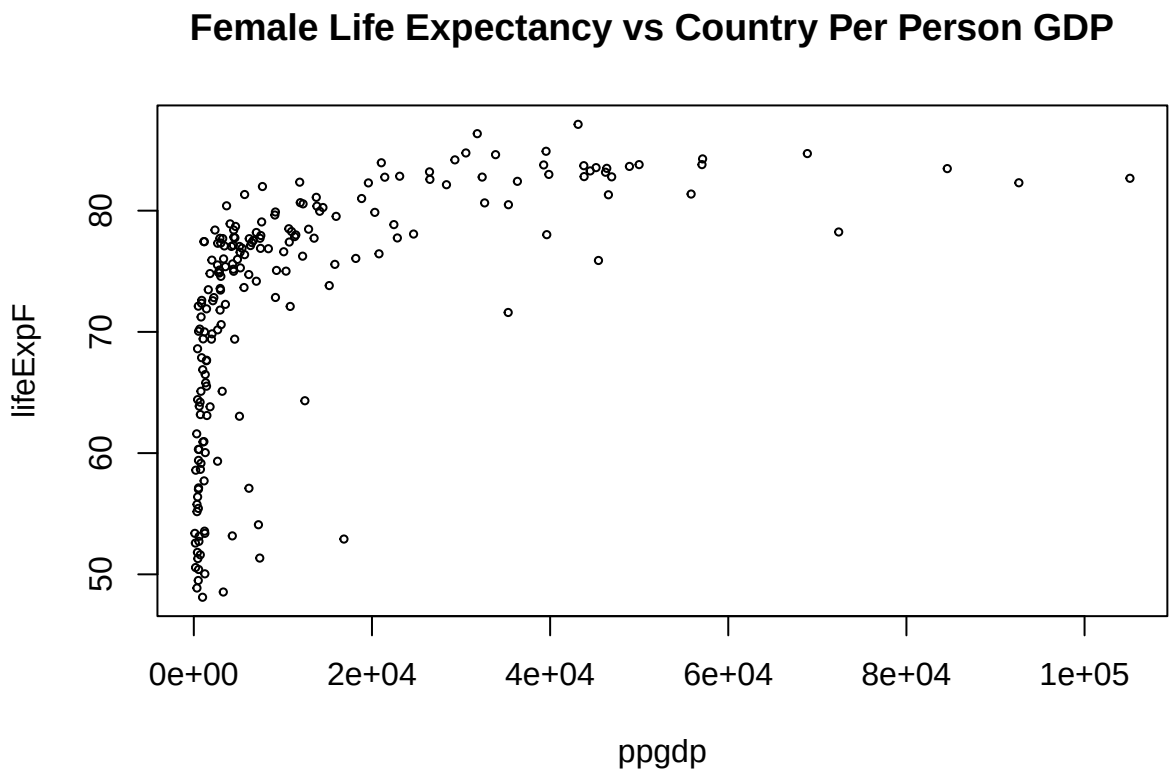
```
hist(UN11.ppgdp,main="Country Per Person GDP",xlab="ppgdp", breaks=15)
abline(v=mean(UN11.ppgdp),col="red")
abline(v=median(UN11.ppgdp),col="blue")
```



Scatter plots: Plotting two quantitative variables

To display the relationship between two quantitative variables, we use a scatter plot. The plot displayed below shows life expectancy of females vs per person gdp. Here, we have extracted the lifeExpF variable directly from the **UN11_data** using the **$** notation. The point size is also changed to 1/2 its original size using the option **cex=.5**.

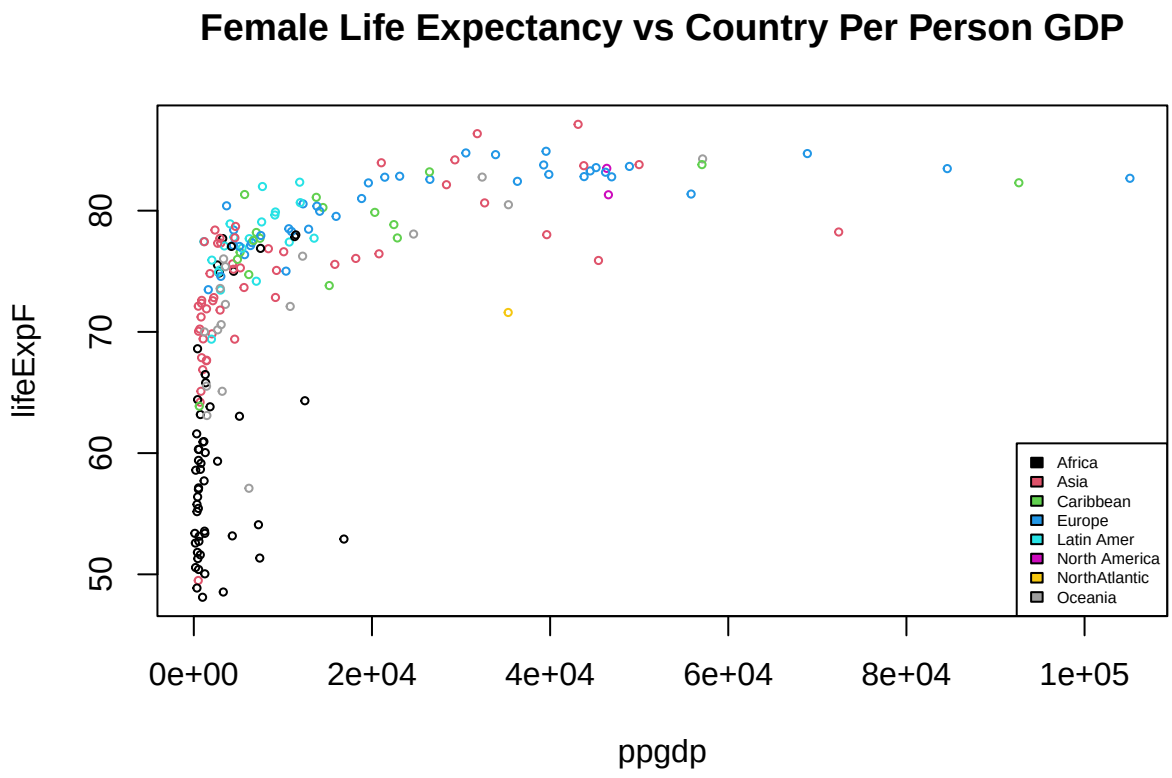
```
plot(UN11.ppgdp,UN11_data$lifeExpF,cex=.5,main="Female Life Expectancy vs Country Per Person GDP  ",
     xlab="ppgdp",ylab="lifeExpF")
```



Color-coded scatterplots

A scatter plot can also be split by a categorical (or factor) variable. In the following example, we split the female life expectancy values up by the region. A legend is also included in this plot.

```
plot(UN11.ppgdp, UN11_data$lifeExpF,
     col=UN11.region,
     cex=.5,main="Female Life Expectancy vs Country Per Person GDP",
     xlab="ppgdp",ylab="lifeExpF")
legend("bottomright",
       legend=levels(UN11.region),
       fill=1:8,
       cex=.5)
```



Section 7: R Basic Operations

Simple arithmetic

```
2+3
```

```
## [1] 5
```

```
4^2 - 3*2
```

```
## [1] 10
```

```
4 ^2-3 * 2
```

```
## [1] 10
```

```
(4^2) - (3*2)
```

```
## [1] 10
```

```
4^(2-3) * 2
```

```
## [1] 0.5
```

```
1 - 6 + 4
```

```
## [1] -1
```

```
1 - (6 + 4)
```

```
## [1] -9
```

```r
4 + 3^2
```

```
## [1] 13
```

```r
(4+3)^2
```

```
## [1] 49
```

Logarithms

```r
log(100)
```

```
## [1] 4.60517
```

```r
log(100, base=10)
```

```
## [1] 2
```

```r
log10(100)
```

```
## [1] 2
```

```r
exp(1)
```

```
## [1] 2.718282
```

```r
log(4)
```

```
## [1] 1.386294
```

```r
log(4, 2)
```

```
## [1] 2
```

Getting help and examples

```r
help(log)
```

```
## starting httpd help server ... done
```

```r
help.start()
```

```
## If nothing happens, you should open
## 'http://127.0.0.1:23663/doc/html/index.html' yourself
```

```r
example("log")
```

```
##
## log> log(exp(3))
## [1] 3
##
## log> log10(1e7) # = 7
## [1] 7
##
## log> x <- 10^-(1+2*1:9)
##
## log> cbind(x, log(1+x), log1p(x), exp(x)-1, expm1(x))
##           x
##  [1,] 1e-03 9.995003e-04 9.995003e-04 1.000500e-03 1.000500e-03
##  [2,] 1e-05 9.999950e-06 9.999950e-06 1.000005e-05 1.000005e-05
```

```
##  [3,] 1e-07 1.000000e-07 1.000000e-07 1.000000e-07 1.000000e-07
##  [4,] 1e-09 1.000000e-09 1.000000e-09 1.000000e-09 1.000000e-09
##  [5,] 1e-11 1.000000e-11 1.000000e-11 1.000000e-11 1.000000e-11
##  [6,] 1e-13 9.992007e-14 1.000000e-13 9.992007e-14 1.000000e-13
##  [7,] 1e-15 1.110223e-15 1.000000e-15 1.110223e-15 1.000000e-15
##  [8,] 1e-17 0.000000e+00 1.000000e-17 0.000000e+00 1.000000e-17
##  [9,] 1e-19 0.000000e+00 1.000000e-19 0.000000e+00 1.000000e-19
```

```
args(log)
```

```
## function (x, base = exp(1))
## NULL
```

Scientific notation

```
1e7
```

```
## [1] 1e+07
```

```
1e3
```

```
## [1] 1000
```

```
4.07e4
```

```
## [1] 40700
```

Vectors and operations on vectors

```
c(1,2,3,4)
```

```
## [1] 1 2 3 4
```

```
c(1, 2, 3, 4)
```

```
## [1] 1 2 3 4
```

```
1:4
```

```
## [1] 1 2 3 4
```

```
4:1
```

```
## [1] 4 3 2 1
```

```
seq(1,4)
```

```
## [1] 1 2 3 4
```

```
c(1, 2, 3, 4) / 4
```

```
## [1] 0.25 0.50 0.75 1.00
```

```
c(1, 2, 3, 4) / c(4,3,2,1)
```

```
## [1] 0.2500000 0.6666667 1.5000000 4.0000000
```

```
log( c(0.1, 1, 10, 100), 10)
```

```
## [1] -1  0  1  2
```

```
c(1, 2, 3, 4) + 4
```

```
## [1] 5 6 7 8
```

```
c(1, 2, 3, 4) + c(4, 3)
```

```
## [1] 5 5 7 7
```

```
c(1, 2, 3, 4) + c(4, 3, 2)
```

```
## Warning in c(1, 2, 3, 4) + c(4, 3, 2): longer object length is not a multiple of
## shorter object length
```

```
## [1] 5 5 5 8
```

How to get 2 4 6 8 ?

```
seq(2, 8, by=2)
```

```
## [1] 2 4 6 8
```

```
seq(2, 8, length=4)
```

```
## [1] 2 4 6 8
```

```
seq(0, 1, by=0.1)
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
seq(0, 1, length=11)
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Creating objects

```
x <- c(1, 2, 3, 4)
x
```

```
## [1] 1 2 3 4
```

```
x/2
```

```
## [1] 0.5 1.0 1.5 2.0
```

```
sqrt(x)
```

```
## [1] 1.000000 1.414214 1.732051 2.000000
```

```
y <- sqrt(x)
y
```

```
## [1] 1.000000 1.414214 1.732051 2.000000
```

```
( y <- sqrt(x) )
```

```
## [1] 1.000000 1.414214 1.732051 2.000000
```

```
y = sqrt(x)
y
```

```
## [1] 1.000000 1.414214 1.732051 2.000000
```

Generate a vector of 100 numbers from N(0,1)

```
x <- rnorm(100)
x
```

```
##   [1]  0.390715682 -0.596899055  0.086179641  1.590581290  0.256665096
##   [6]  0.336111593  2.452677988 -0.054953062  0.217563480 -1.525111613
##  [11] -0.556267541  0.244818740  0.486125220  0.183154364  0.395191303
##  [16] -0.674070342  0.474369426 -0.974772488 -1.314119519  1.628611817
##  [21]  0.063539459 -0.751150122  1.039792552  1.110870273 -0.173148911
##  [26] -0.033175020 -0.332726141  0.278567530 -0.376568774 -0.937024744
##  [31] -0.515563586 -1.570074489  0.664038682 -0.923927146  0.254097469
##  [36] -1.012059126  1.254023949 -0.539984187 -0.983154368 -0.270535876
##  [41]  1.216640936 -0.996334564  0.231178460 -0.897019508  0.521632278
##  [46]  0.344399026 -0.292821183  0.585750130  0.304154379 -1.342340642
##  [51] -2.093442081  0.211148185 -1.502158900  0.259440177 -0.121626573
##  [56] -0.255388341 -3.241080941 -0.405020174  0.011010555 -1.004485005
##  [61]  0.494921570  1.894659425  0.129307173  0.598247081 -1.149961037
##  [66] -1.568552488 -0.253031146  1.046463934  0.518301717  0.148436441
##  [71]  1.018864367 -0.268632116  1.091481639  0.367955326  0.212614396
##  [76] -0.908255646 -0.455985182  0.711939342  0.048171013 -0.094953492
##  [81]  1.149958544  0.471534469 -0.617673847  0.018065123 -1.203462853
##  [86] -0.734627531 -0.764707670 -0.113688832  0.273015657  0.009428547
##  [91] -0.683631656 -0.227620565 -0.696458267  0.877309160  0.101643873
##  [96] -0.243546127 -0.009890428  1.402689059 -0.450568993 -1.558822067
```

```
summary(x)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -3.241081 -0.676461 -0.000231 -0.085930  0.391835  2.452678
```

Character data

```
( words <- c("Welcome", "to", "Linear Regression Models", "Course") )
```

```
## [1] "Welcome"                   "to"
## [3] "Linear Regression Models" "Course"
```

```
paste(words, collapse=" ")
```

```
## [1] "Welcome to Linear Regression Models Course"
```

Logical data

```
(vals <- c(TRUE, TRUE, FALSE, TRUE) )
```

```
## [1]  TRUE  TRUE FALSE  TRUE
```

```
!vals
```

```
## [1] FALSE FALSE  TRUE FALSE
```

```
y
```

```
## [1] 1.000000 1.414214 1.732051 2.000000
```

```
sum(y)
```

```
## [1] 6.146264
```

```
vals
```

```
## [1]  TRUE  TRUE FALSE  TRUE
```

```
sum(vals)
```

```
## [1] 3
```

```
sum(!vals)
```

```
## [1] 1
```

Indexing vectors

What's in the 12th position of x?

```
x <- rnorm(50)
x  #Read off from the 12th location in the full vector
```

```
##  [1] -0.72283655 -1.03099485 -0.27392307  0.57842104  0.38816203  1.70145388
##  [7] -1.54081563 -0.82250341  1.08853256  1.28310729 -0.11797639 -0.47420923
## [13] -0.04379007 -0.07645576 -1.05108123 -0.09295174 -0.78729981  0.08029124
## [19] -0.79579437 -0.18122048  0.20408441 -0.34457962 -0.67410384  0.81866690
## [25]  1.49485722  1.22032614  0.78131921 -1.28599497  1.24470019  0.92962346
## [31] -0.78763464  1.37420198 -0.18951107  0.37756916 -1.65454035 -1.64375474
## [37]  1.67750723 -0.51154399 -0.89200067  1.23202194 -1.55564481 -0.01959716
## [43]  2.51758304  2.44855405 -1.46214059  1.00884205 -0.09835549 -0.69025721
## [49]  1.28769916 -0.45003535
```

```
x[12] #or simply extract the value in the 12th location
```

```
## [1] -0.4742092
```

```
x[6:15]
```

```
##  [1]  1.70145388 -1.54081563 -0.82250341  1.08853256  1.28310729 -0.11797639
##  [7] -0.47420923 -0.04379007 -0.07645576 -1.05108123
```

```
x[1:10]
```

```
##  [1] -0.7228366 -1.0309949 -0.2739231  0.5784210  0.3881620  1.7014539
##  [7] -1.5408156 -0.8225034  1.0885326  1.2831073
```

```
x[-(11:50)]
```

```
##  [1] -0.7228366 -1.0309949 -0.2739231  0.5784210  0.3881620  1.7014539
##  [7] -1.5408156 -0.8225034  1.0885326  1.2831073
```

```
words[2]
```

```
## [1] "to"
```

```
vals[3]
```

```
## [1] FALSE
```

User defined functions

```
x
```

```
##  [1] -0.72283655 -1.03099485 -0.27392307  0.57842104  0.38816203  1.70145388
##  [7] -1.54081563 -0.82250341  1.08853256  1.28310729 -0.11797639 -0.47420923
## [13] -0.04379007 -0.07645576 -1.05108123 -0.09295174 -0.78729981  0.08029124
## [19] -0.79579437 -0.18122048  0.20408441 -0.34457962 -0.67410384  0.81866690
## [25]  1.49485722  1.22032614  0.78131921 -1.28599497  1.24470019  0.92962346
## [31] -0.78763464  1.37420198 -0.18951107  0.37756916 -1.65454035 -1.64375474
## [37]  1.67750723 -0.51154399 -0.89200067  1.23202194 -1.55564481 -0.01959716
## [43]  2.51758304  2.44855405 -1.46214059  1.00884205 -0.09835549 -0.69025721
## [49]  1.28769916 -0.45003535
```

```
mean(x)
```

```
## [1] 0.06931954
```

```
sum(x)
```

```
## [1] 3.465977
```

```
length(x)
```

```
## [1] 50
```

```
sum(x) / length(x)
```

```
## [1] 0.06931954
```

```
myMean <- function(x){ sum(x) / length(x) }
myMean(x)
```

```
## [1] 0.06931954
```

```
mean(y)
```

```
## [1] 1.536566
```

```
myMean(y)
```

```
## [1] 1.536566
```

```
mean(1:100)
```

```
## [1] 50.5
```

```
myMean(1:100)
```

```
## [1] 50.5
```

Command line editing: Left/Right/Up arrows, Backspace/Delete keys

Note: In the command window, Up arrow scrolls through previous commands.

Cleaning up

```
objects()
```

```
##  [1] "a"              "b"              "c"              "d"
##  [5] "myMean"         "UN11.fertility" "UN11.ppgdp"     "UN11.region"
##  [9] "UN11_data"      "vals"           "words"          "x"
## [13] "y"
```

```
remove(x, y, vals, words)
objects()
```

```
## [1] "a"              "b"              "c"              "d"
## [5] "myMean"         "UN11.fertility" "UN11.ppgdp"     "UN11.region"
## [9] "UN11_data"
```

Working with logical variables and operations

```
1 == 2
```

```
## [1] FALSE
```

```
1 != 2
```

```
## [1] TRUE
```

```
1 < 2
```

```
## [1] TRUE
```

```
1 <= 2
```

```
## [1] TRUE
```

```
1 < 1:3
```

```
## [1] FALSE  TRUE  TRUE
```

```
3:1 > 1:3
```

```
## [1]  TRUE FALSE FALSE
```

```
TRUE & c(TRUE, FALSE)
```

```
## [1]  TRUE FALSE
```

```
TRUE | c(TRUE, FALSE)
```

```
## [1] TRUE TRUE
```

```
c(TRUE, TRUE, TRUE) | c(TRUE, TRUE, FALSE)
```

```
## [1] TRUE TRUE TRUE
```

```
x<-rnorm(100)
(z <- x[1:10] )
```

```
##  [1] -0.4978438  2.5433511 -0.1731713 -0.2542464  1.1931300  1.9248239
##  [7]  0.3090743 -0.3325560  0.1314518 -0.6924591
```

```
z < -0.5
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
z > 0.5
```

```
##  [1] FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

```
z < -0.5 | z > 0.5
```

```
##  [1] FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE
```

```
abs(z) > .5
```

```
##  [1] FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE
```

```
z[ abs(z) > 0.5 ]
```

```
## [1]  2.5433511  1.1931300  1.9248239 -0.6924591
```

```
z[ !(abs(z) > 0.5) ]
```

```
## [1] -0.4978438 -0.1731713 -0.2542464  0.3090743 -0.3325560  0.1314518
```