Introduction to R
Due September 9

1. Obtain the windows version of R

R is free software for statistical analysis and computing. Much information can be found at http://www.r-project.org

An introduction to R by R development team is provided at
http://cran.r-project.org/doc/manuals/R-intro.pdf

Windows version of R can be downloaded from
http://cran.r-project.org/bin/windows/base/

In addition, a small software Tinn-R can be used to edit code and Run it in R.

2. Get help

R has a build-in help function. To get more information on any specific named function, for example solve and mean, the command is

```
help(solve)
help(mean)
```

An alternative is

```
?solve
?mean
```

For most R installations, help is available in HTML format by running
help.start()
You will browse R help in internet browser

Technically R is an expression language with a very simple syntax. It is case sensitive, so A and a are different symbols and would refer to different variables. The set of symbols which can be used in R names depends on the operating system and country within which R is being run (technically on the locale in use). Normally all alphanumeric symbols are allowed (and in some countries this includes accented letters) plus '.' and '_', with the restriction that a name must start with '.' or a letter, and if it starts with '.' the second character must not be a digit.

Commands are separated either by a semi-colon (';'), or by a newline. Elementary commands can be grouped together into one compound expression by braces ('{' and '}'). Comments can be put almost anywhere, starting with a hashmark ('#'), everything to the end of the line is a comment.

If a command is not complete at the end of a line, R will give a different prompt, by default
+

on second and subsequent lines and continue to read input until the command is syntactically complete. This prompt may be changed by the user.

3. Arithmetic operation

Some examples are

```
>5+2*3.5
[1] 12
>(5+2)*3.5
[1] 24.5
>5/4
[1] 1.25
>1/0
[1] Inf
```

Inf means infinite

4. Objects

In R, everything is treated as an object. To store a result to an object, use '=' or '<-' symbols.

```
> a <- 3+10
> b <- a*3
>a
[1]  9
```

To see all objects, use ls() or objects ()

```
ls()
[1] "a" "b"
```

Names of objects must start with a character or '.' Symbol. Because name is CASE SENSITIVE, you need to be cautious when you define the name for an object.

5. Vectors

R operates on named data structures. The simplest structure is the numeric vector, which is a single entity consisting of an ordered collection of numbers. One way to construct a vector is using c() function

```
> x<-c(3,4,5,5, 3,7, 12,5)
> x
[1]   3   4   5   5   3   7   12   5
```

Alternatively, sequential numbers can be generated. For instance

```
> y<-1:10
> y
[1] 1   2   3   4   5   6   7   8   9   10
```

Or a sequence function can be used to generate a sequence

```
> s<-seq(10, 100, 10)
> s
 [1]   10   20   30   40   50   60   70   80   90   100
```

Vector operation in R

```
> v<-(s+y)/10
> v
 [1]   1.1   2.2   3.3   4.4   5.5   6.6   7.7   8.8   9.9 11.0
```

A vector can be an input for functions. For instance,

```
> mean(y)
[1] 5.5

> c(var(y), sum(y), max(y), min(y), length(y))
[1] 9.166667 55.000000 10.000000   1.000000 10.000000
> summary (y)
    Min.  1st Qu.  Median    Mean  3rd Qu.     Max.
    1.00     3.25     5.50     5.50     7.75    10.00
```

Vectors can be used in arithmetic expressions, in which case the operations are performed element by element. Vectors occurring in the same expression need not be of the same length. If they are not, the value of the expression is a vector with the same length as the longest vector which occurs in the expression. Shorter vectors in the expression are recycled as often as need be (perhaps fractionally) until they match the length of the longest vector.

Common operations include

*max(x)* # the largest elements of a vector x
*min(x)* # the smallest elements of a vector x.
*range(x)* # a vector of length two, namely c(min(x), max(x)).
*length(x)* # the number of elements in x,
*sum(x)* # the total number of elements in x.
*mean(x)* # sample mean.
*var(x)* #sample variance.
*sd(x)* # standard deviation

5.1 Reading and writing vectors with a file

A vector can be read from and written to a file. To save a vector in a file, type

```
> write (v, file = “datafile.txt’, ncolumns =1)
```

Command rm() can be used to delete an object. For example,

```
> ls()
[1] "a" "b" "s" "v" "x" "y"
> rm(v)
> rm(a,b)
> ls()
[1] "s" "x" "y"
```

To read a vector from a file, function scan() can be used.

```
v<-scan("datafile.txt")
Read 10 items
> v
 [1]  1.1  2.2  3.3  4.4  5.5  6.6  7.7  8.8  9.9 11.0
```

5.2 Subset of a vector

To subset a vector, use [index] symbols. T or F can be used to extract a subset of a vector

```
> x<-10:20
> x[1:6]
[1] 10 11 12 13 14 15
> x[c(2,4,5,10)]
[1] 11 13 14 19
> x[c(T,F,F,T,T,T,F,F,F,T)]
[1] 10 13 14 15 19 20
> b<-x<15
> b
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
FALSE
>x[b]
[1] 10 11 12 13 14
```

6. List

A list is an object consisting of ordered collection of objects known as components

```
> Lst<-list("x"=1:10, "y"=seq(10,1,-1),"names"=c("loc1", "loc2"))
> Lst
$x
 [1]  1  2  3  4  5  6  7  8  9 10

$y
 [1] 10  9  8  7  6  5  4  3  2  1

$names
[1] "loc1" "loc2"
```

Each component of a list object can be accessed using [[]] or $

```
> Lst[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10
> Lst[["y"]]
 [1] 10  9  8  7  6  5  4  3  2  1
> Lst$names
[1] "loc1" "loc2"
> Lst[["names"]]
[1] "loc1" "loc2"
> Lst[["names"]][1]
[1] "loc1"
```

7. Data frame

A data frame is a list with class “data.frame”. Component in a data frame should be vector. The form of a data frame can be compatible with a table because it is composed of rows and columns. To make a data frame, use “data.frame()” function

```
> x<-1:10
> y<-rnorm(10,0,1)
> z<-exp(y)
> ex.df<-data.frame(id=x, rn=y, exp.rn=z)
> ex.df
   id         rn       exp.rn
1   1  0.7599951  2.1382657
2   2  1.0036588  2.7282458
3   3  1.0751746  2.9305046
4   4  0.1018214  1.1071857
5   5 -2.0992703  0.1225458
6   6  0.2792984  1.3222018
7   7 -0.4963828  0.6087286
8   8 -0.1589320  0.8530543
9   9  2.4945482 12.1162580
10 10 -1.1056537  0.3309944
```

Individual element of data frame can be accessed using $, or []

```
> ex.df[1:3,]
  id       rn      exp.rn
1  1 0.759995 2.138266
2  2 1.003659 2.728246
3  3 1.075175 2.930505
> ex.df[1:3, "rn"]
[1] 0.759995 1.003659 1.075175
> ex.df$rn[1:3]
[1] 0.759995 1.003659 1.075175
> ex.df[1:3,2:3]
         rn      exp.rn
```

```
1 0.759995 2.138266
2 1.003659 2.728246
3 1.075175 2.930505
```

```
>ex.df[,c(1,3)]

      id        exp.rn
1      1    2.1382657
2      2    2.7282458
3      3    2.9305046
4      4    1.1071857
5      5    0.1225458
6      6    1.3222018
7      7    0.6087286
8      8    0.8530543
9      9 12.1162580
10 10    0.3309944
```

Reading and writing a data frame

```
>write.table(ex.df, file="f:/UA/Geog574/mydf.csv",sep=",")
```

If you want to know where the file is stored, you can get current working directory using getwd(). If necessary, you can change working directory with setwd() function

```
> getwd()
[1] "C:/Program Files/R/R-2.2.0/bin"
> setwd("f:/UA/Geog574/")
> getwd()
[1] "f:/UA/Geog574"
```

To read a table from a text file, use read.table() command.

```
> ex2.df<-read.table("mydf.csv", header=T, sep=",")
> ex2.df
      id          rn        exp.rn
1      1    0.7599951    2.1382657
2      2    1.0036588    2.7282458
3      3    1.0751746    2.9305046
4      4    0.1018214    1.1071857
5      5 -2.0992703    0.1225458
6      6    0.2792984    1.3222018
7      7 -0.4963828    0.6087286
8      8 -0.1589320    0.8530543
9      9    2.4945482 12.1162580
10 10 -1.1056537    0.3309944
```

8. Plot data

R provides functionalities to plot data such as histogram and scatterplot. Here, state.x77 dataset that is provided with R will be used for exercise.

```
> class(state.x77)
> state77<-as.data.frame(state.x77)
> colnames(state77)
[1] "Population" "Income" "Illiteracy" "Life Exp" "Murder" "HS Grad" "Frost" "Area"

>hist(state77$Income, nclass=10, main="Income", col="light blue")
>hist(state77$"HS Grad", main="High School", col="light blue")
>boxplot(state77$Income)
>plot (state77$"HS Grad", state77$Income, xlab="HS Grad", ylab="Income")
>plot(state77[,c(2,35)])
>qqnorm(state77$Income)
>qqline(state77$Income)
```

9. Fitting linear regression

We will use lm() function to fit a linear regression model for the above Income and HS Grad dataset

$$Y_i = \alpha + \beta X_i + \varepsilon_i \qquad \varepsilon_i \sim N(0,1)$$

```
>state77.lm<-lm(state77$Income~state77$"HS Grad")
```

To summarize the result of the regression:

```
>summary(state77.lm)
Call:
lm(formula = state77$Income ~ state77$"HS Grad")

Residuals:
     Min       1Q   Median       3Q      Max
-1083.13  -277.41   -34.15   241.46  1238.17

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         1931.105    462.739   4.173 0.000125 ***
state77$"HS Grad"     47.162      8.616   5.474 1.58e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 487.1 on 48 degrees of freedom
```

Multiple R-Squared: 0.3843, Adjusted R-squared: 0.3715
F-statistic: 29.96 on 1 and 48 DF, p-value: 1.579e-06
According to the result, the regression model is

$$Income_i = 1931.105 + 47.162\,HS\ Grad_i + \varepsilon_i \qquad \varepsilon_i \sim N(0,1)$$

To investigate the results visually, get the residuals and fitted values. Then generate plots with the values.

```
>resids<-resid(state77.lm)
>fits<-fitted(state77.lm)
>par(mfrow=c(2,2))

#HS Grad vs. Income with fitted line
>plot (state77$”HS Grad”, state77$Income, xlab=”HS Grad”, ylab=”Income”)
>lines(state77$”HS Grad”, fits)

#HS Grad vs. residuals
>plot(state77$"HS Grad", resids, ylab="Residuals")
>abline(h=0)

#Histogram and Q-Q plot
>hist(resids, xlab="Residuals")
>qqnorm(resids, ylab="Residuals")
>qqline(resids)
>par(mfrow=c(1,1))
```

Assignment

1. Read the crime dataset “crime.csv” in the lab data folder
2. create a histogram, boxplot, and QQ plot for each of following variables: CRIME, HOVAL, and INC. Discuss the normality of the three variables
3. Create a scatterplot matrix for CRIME, HOVAL, and INC variables. Discuss the relationship of the variables
4. Fit a regression model with CRIME as dependent variables and HOVAL & INC as independent variables. Provide an equation for the fitted regression model
5. From the result of the regression model, generate fitted plot, residual plot, histogram of the residuals, and QQ plot of the residuals. Discuss if the assumption of linear regression is reasonably valid or not.
6. Attach all R codes that you developed for this assignment at the end of your exercise.