Introduction to R

1. Import/Export Data in R.

scan: Basic function to input data from a text file, **scan** returns a vector of numbers, strings , or logical.

To read from the console use: scan()

To read from a file use: scan(filepath, what = " ")

Example:

```
> z <- scan()
1: 1 2
3: 4 6 7
6: 10 11
8:
Read 7 items
> z
[1] 1 2 4 6 7 10 11
```

read.table: To read data from a text file in the form of a data table and produce a data frame in R.

read.table(file, sep=" ",head=F)

read.csv: To read data from a "," delimited file in the form of a data table and produce a data frame in R.

read.csv(file, sep=" ",head=F)

library(foreign)

read.xport:	Read a SAS XPORT Format Library
read.dbf:	Read a DBF File
write.dbf:	Write a DBF File

write.table: To output an R data frame or an R matrix into a text file data table

Example:

```
pima = read.table("PIMA.txt",sep=",",head=T)
write.table(pima,file="c:\\Documents and Settings\\a\\My Documents\\587\\PIMA1.txt",row.names=F)
```

save and load:

```
save(x,y,z, file="C:/saved.txt", compress=F)

save.image(file="C:/.RData", compress=F)
```

```
load( file="C:/saved.txt" )

dump()

source()
```

2. MORE DATA MANIPULATION

Sorting:

```
    sort.list(pima$PEDI)

    pima[sort.list(pima$PEDI),][1:20,]
```

Transformations:

```
    par(mfrow=c(3,3))

    apply(pima[,-9],2,hist)

    lPRG = log(pima$PRG)

    lPRG = log(1+pima$PRG)

    hist(lPRG)

    pima$lPRG = lPRG
```

LOGICAL OPERATORS: > , >= , < , <= , == , != , & , | or ! Not

```
  x = rnorm(10)
 u = x > 0
 u
 x[u]
 x
```

CONDITIONAL

```
 x <- rnorm(10)
 u <- sum(x)
 if( u < 0) { x <- -x }
```

LOOPS

```
 for loop:

z <- matrix(rnorm(200),20,10)
 mean.samp <- NULL
 for(i in 1:10) {
 mean.samp[i] <- mean(z[,i])
 }
 stem(mean.samp)
```

```
while loop:

while(condition is true) { do this }

repeat  { do this ; if (this) break  }

apply( x, 1, fun)

sapply( x, fun)

tapply( x, y, fun)

It is very important to vectorize the code.
```

3. OBJECT ORIENTED PROGRAMING

Classes and Methods

```
> print
function (x, ...)
UseMethod("print")
<environment: namespace:base>
> methods(print)
  [1] print.acf*
  [2] print.anova
  [3] print.aov*
  [4] print.aovlist*
  [5] print.ar*
  [6] print.Arima*
  [7] print.arima0*
  [8] print.AsIs
  [9] print.Bibtex*
 [10] print.by
 [11] print.check_code_usage_in_package*
 [12] print.check_demo_index*
 [13] print.check_make_vars*
>  class(x)
```

4. FUNCTIONS

```
fourmom <- function(x) {
 m1 <- c(mean(x))
 m2 <- mean(x^2)
 m3 <- mean(x^3)
 m4 <- mean(x^4)
 list(m1=m1,m2=m2,m3=m3,m4=m4)
}
x <- rnorm(10)
fourmom(x)
```

Homework:

Write your own function that takes the median of the rows of a matrix without using loops or apply.

Exercise:

Take the following function. Can you tell me what it does?

```
cmean <- function(x) {

n <- nrow(x)
p <- ncol(x)
nax = is.na(x)
nna <- rep(1,n) %*% nax
i <- nna > 0
result = rep(NA,p)
result[!i] <- rep(1,n) %*% x[,!i] / n
result
}
```

Modify the function so it will allow you to remove missing values.

Answer:

```
cmean <- function(x, na.rm=T) {
n <- nrow(x)
if( na.rm) {
nax = is.na(x)
nna <- rep(1,n) %*% (!nax)
x[nax] <- 0
return( c(rep(1,n) %*% x / nna))
}
else return(c( rep(1,n) %*% x /n))
}
```

5. PLOTS:

5.1 High level:

barplot() to draw bar-plots

```
data(VADeaths, package = "base")
barplot(VADeaths, plot = FALSE)
barplot(VADeaths, plot = FALSE, beside = TRUE)
mp <- barplot(VADeaths) # default
tot <- colMeans(VADeaths)
text(mp, tot + 3, format(tot), xpd = TRUE, col = "blue")
```

```
#
barplot(VADeaths, beside = TRUE,
col = c("lightblue", "mistyrose", "lightcyan",
"lavender", "cornsilk"),
legend = rownames(VADeaths), ylim = c(0, 100))
title(main = "Death Rates in Virginia", font.main = 4)
#
hh <- t(VADeaths)[, 5:1]
mybarcol <- "gray20"
mp <- barplot(hh, beside = TRUE,
col = c("lightblue", "mistyrose",
"lightcyan", "lavender"),
legend = colnames(VADeaths), ylim= c(0,100),
main = "Death Rates in Virginia", font.main = 4,
sub = "Faked upper 2*sigma error bars", col.sub = mybarcol,
cex.names = 1.5)
segments(mp, hh, mp, hh + 2*sqrt(1000*hh/100), col = mybarcol, lwd = 1.5)
stopifnot(dim(mp) == dim(hh))# corresponding matrices
mtext(side = 1, at = colMeans(mp), line = -2,
text = paste("Mean", formatC(colMeans(hh))), col = "red")
```

plot() to draw scatter-plots

```
x <- rnorm(30)

y <- rcauchy(30)
abline(0,1)
plot(x,y, pch=16, cex=0.7, xlab=”from Normal”,ylab=”from
Cauchy”,main=”Random Data”)
abline(0,1)
```

5.2 LOW LEVEL COMMANDS

```
 par(mfrow=c(2,2))

mm <- array(c(1, 2), dim = 2)
nf <- layout(mm, 4, c(5, 3), TRUE)
plot(x,y)
hist(x)
par(fig=0,0.5,0,1)

par( mar=c(2,2,1,1))
```

Pairwise scatter plots:

pairs(state.x77)

3D Plots: Are sometimes useful but may need animation

5.3 Conditional plots

```
(In R) data(state)

attach(data.frame(state.x77))#> don't need `data' arg. below

coplot(Life.Exp ~ Income | Illiteracy * state.region, number = 3,

panel = function(x, y, ...) panel.smooth(x, y, span = .8, ...))

detach() # data.frame(state.x77)
```

Parallel Plot: Graphof a multivariate dataset where the observations are represented by lines.

Objectives:

1. To visualize comparisons between multivariate data groups.
2. Help asses the quality of classification tools
3. To find data clusters and outliers.

```
parallel( ~ state.x77 | state.region )
```

Using the Crime dataset:parallel(~X[,1:4])

```
hist.inv <- function(x,side=1) {
gg <- hist(x,plot=F)
con <- max(gg$counts)*1.05
if( side==1) {
 plot(range(gg$breaks),c(0,con),type="n", axes=F,xlab="",ylab="")
 axis(2,con -( i <- pretty(gg$counts)),i)
 for(i in 1:length(gg$counts)) rect(gg$breaks[i],con-
gg$counts[i],gg$breaks[i+1],con)
}
if(side==2) {
 plot(c(0,con),range(gg$breaks),type="n", axes=F,xlab="",ylab="")
 axis(1,con -( i <- pretty(gg$counts)),i)
 for(i in 1:length(gg$counts)) rect(con-
gg$counts[i],gg$breaks[i],con,gg$breaks[i+1])
 }
}

superplot <- function(x,y) {
par(mar=c(1.1,1.1,1,1))
nf <- layout(array(c(2,4,1,3), dim=c(2,2)), c(1,3), c(3, 1), TRUE)
plot(x,y)
hist.inv(y,2)
hist.inv(x) ; frame()
}
x = pima[,2]; y= pima[,3]; superplot(x,y)

##Another way of doing it

superplot2 = function(x,y) {
xhist <- hist(x,  plot=FALSE)
yhist <- hist(y, plot=FALSE)
top <- max(c(xhist$counts, yhist$counts))
xrange <- range(x)
yrange <- range(y)
nf <- layout(matrix(c(2,1,0,3),2,2,byrow=TRUE), c(1,3), c(3,1), TRUE)
layout.show(nf)
par(mar=c(1,1,1,1))
plot(x, y, xlim=xrange, ylim=yrange, xlab="", ylab="")
par(mar=c(1,1,1,1))
barplot(yhist$counts, xlim=c(top, 0), space=0, horiz=TRUE) # !
par(mar=c(1,1,1,1))
barplot(xhist$counts, ylim=c(top,0), space=0) #!ylim!
invisible()
}

x <- pmin(3, pmax(-3, rnorm(50)))
y <- pmin(3, pmax(-3, rnorm(50)))
```

6. LIBRARIES.

mva- Multivariate Analysis. Principal Components

```
library()
library(cluster)
## the variances of the variables in the
## USArrests data vary by orders of magnitude
data(USArrests)
(pc.cr <- princomp(USArrests))
princomp(USArrests, cor = TRUE)
princomp(scale(USArrests, scale = TRUE, center = TRUE), cor = FALSE)

summary(pc.cr <- princomp(USArrests))
loadings(pc.cr)
plot(pc.cr) # does a screeplot.
biplot(pc.cr)
```

7. STATISTICAL MODELS:

ANOVA: Annette Dobson (1990) "An Introduction to Generalized Linear Models". Plant Weight Data.

```
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
anova(lm.D9 <- lm(weight ~ group))
summary(lm.D90 <- lm(weight ~ group - 1))# omitting intercept
summary(resid(lm.D9) - resid(lm.D90)) #- residuals almost identical
opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))
plot(lm.D9, las = 1) # Residuals, Fitted, ...
par(opar)
```

Smoothing Splines:

```
 library(modreg)
  data(cars)
  attach(cars)
  plot(speed, dist, main = "data(cars) & smoothing splines")
  cars.spl <- smooth.spline(speed, dist)
  (cars.spl)
  all(cars.spl $ w == table(speed)) # TRUE (weights = multiplicities)

 lines(cars.spl, col = "blue")
 lines(smooth.spline(speed, dist, df=10), lty=2, col = "red")
 legend(5,120,c(paste("default [C.V.] => df =",round(cars.spl$df,1)),
 "s( * , df = 10)"), col = c("blue","red"), lty = 1:2,
 bg='bisque')
 detach()
```

8. WRITING R – packages:

These should work for not very old versions of windows like Windows 7 32 bit and 64 bit, Windows Vista, Windows XP with regular updates. It may not work for most Windows 2000 versions and older but I would try it anyway.

1. Create a package skeleton in R, using the command package.skeleton(name="roots",list=c('superplot','superplot2','hist.inv'))
   Then edit the DESCRIPTION file and .Rd help files appropriately. You will need to follow the instructions of the manual (Help =>Manuals=> Writing R Extensions

2. Download and install the following software:
   1. Mike TeX: http://www.miktex.org
   2. Rtools: http://www.murdoch-sutherland.com/Rtools/.
      Rtools will install: Vanilla Perl, Tool set, MinGW, Cygwin, TCL/TK

3. Check PATH VARIABLE: (Start->Settings->)Control Panel->System->Advanced. Click on the

Environment Variables button, which should be in the middle. The previous software in Rtools should be in the path. If your version of windows is not very old the previous installation on step 2. should make the appropriate modifications to the path. In my windows 7 - 64 bits my path like this at the beginning of the path:
c:\Rtools\bin;c:\Rtools\perl\bin; C:\R\bin;c:\Rtools\MinGW64\bin;C:\Program Files\R\R-2.15.1\bin\x64;C:\Program Files (x86)\MiKTeX 2.9\miktex\bin; and here continues with other addresses for other non R software. For a 32 bit windows is not identical either, and there will be modifications related to your installation folder addresses.
You can check in the CMD window if R is already in the path by just typing R.
You can also check if MikeTeX is in the path by typing PATH in the CMD window.

4. Download the newest version of the R batch files http://cran.r-project.org/contrib/extra/batchfiles/
Extract the files to the folder `C:\Rbatch`. And pasted `C:\Rbatch; to the PATH as before.`

5. OPEN cmd window with administrator permission. Then go to the folder root to where the package was created
```
Cd ….
```

6. RUN the command to create the R-package.
```
Rcmd build  “packagelocation”
```

7. RUN the command to install the R-package.
```
Rcmd INSTALL  “packagelocation”
```

8. If there are any error messages that you can’t resolve please bring them to class and we will deal with them.

Simple exercise:
Take a look at the superplot2 function on the notes above. The bottom left box of the resulting plot is empty. Please fill it up with a summary of the two variables including means, medians, standard deviations and correlation coefficient. If you have room add also the MAD's. (Median absolute deviation)
Homework:
1. Fix *superplot* and *superplot2*. There is a problem with the scales on the side. The histogram scale does not exactly correspond to the scales of the axes of the scatter plot. There is a small displacement please try to fix this is you can but only after you succeed with part 2.
2. Write an R-package with these functions. *superplot*, *superplot2*,***`hist.inv.`***