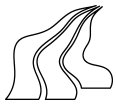
DEPARTMENT OF MATHEMATICAL SCIENCES

AALBORG UNIVERSITY



FREDRIK BAJERS VEJ 7 G ▪ DK - 9220 AALBORG ØST ▪ DENMARK
URL: www.math.auc.dk
E-mail: jm@math.aau.dk

Phone: +45 96 35 88 63
Fax: +45 98 15 81 29

Exercise: An introduction to R (Part A)

The following is a modified version of Appendix A in "An Introduction to R". This is meant as a way of getting to know R better — hence you are not expected to understand everything in detail.

The following session is intended to introduce you to some features of the R environment by using them. Many features of the system will be unfamiliar and puzzling at first, but this puzzlement will soon disappear.

The following is written for the UNIX user (differences are minor when using Windows).

`$R` Start R as appropriate for your platform.

The R program begins, with a banner.

(Within R, the prompt on the left hand side will not be shown to avoid confusion.)

```
help.start()
```

Start the HTML interface to on-line help (using a web browser available at your machine). You should briefly explore the features of this facility with the mouse.

Iconify the help window and move on to the next part.

```
x <- rnorm(200)
y <- rnorm(x)
```

Generate two pseudo-random standard normal vectors of x- and y-coordinates of the same length (here 200).

```
hist(x)
hist(y)
```

Produce histrograms for x and y.

```
plot(x, y)
```

Plot the points in the plane. A graphics window will appear automatically.

`ls()` See which R objects are now in the R workspace.

`rm(x, y)`
Remove objects no longer needed. (Clean up).

```
x <- 1:20
x
```
Make $x = (1, 2, \ldots, 20)$.

```
w <- 1 + sqrt(x)/2
w
```
A 'weight' vector.

```
z<-rnorm(x)*w
```
`z` is the vector obtained by multiplying the vectors `rnorm(x)` and `w` coordinatewise.

```
dummy <- data.frame(x=x, y= x + z)
dummy
```
Make a *data frame* of two columns, x and y, and look at it.

```
fm <- lm(y ~  x, data=dummy)
summary(fm)
```
Fit a simple linear regression of y on x and look at the analysis. The R notaion `y ~ x` implies that the mean of y given x is of the form $a_0 + a_1 x$ (note that the intercept a_0 is default). Writing `y ~ x + x^2 -1` implies that the mean of y given x is of the form $a_1 x + a_2 x^2$.

```
plot(dummy$x, dummy$y)
```
Standard point plot.

```
lines(dummy$x, dummy$y)
```
Connect the points with lines.

```
abline(0, 1, lty=3)
```
The true regression line: (intercept 0, slope 1).

```
abline(coef(fm))
```
Estimated regression line.

```
plot(fitted(fm), resid(fm),
  xlab="Fitted values",
  ylab="Residuals",
  main="Residuals vs Fitted")
```
A standard regression diagnostic plot to check for heteroscedasticity. Can you see it?

```
qqnorm(resid(fm), main="Residuals Rankit Plot")
```
A normal scores plot to check for skewness, kurtosis and outliers. (Not very useful here.)

```
rm(fm, x, dummy)
```
Clean up again.

Exercise: An introduction to R (Part B)

The next section will look at data from the classical experiment of Michaelson and Morley to measure the speed of light.

```
data(morley)
morley
```

Read in the Michaelson and Morley data as a data frame, and look at it. There are five experiments (column `Expt`) and each has 20 runs (column `Run`) and `Speed` is the recorded speed of light, suitably coded. You can use `names(morley)` as a way to get the names of different columns.

```
morley$Expt <- factor(morley$Expt)
morley$Run <- factor(morley$Run)
```

Change `Expt` and `Run` into factors.

```
plot(morley$Expt, morley$Speed, main="Speed of Light Data", xlab="Experi
```

Compare the five experiments with simple boxplots.

```
fm <- aov(Speed ~ Run + Expt, data=morley)
summary(fm)
```

Two-way analysis of variance, with 'runs' and 'experiments' as factors.

```
rm(fm)
```

Clean up before moving on.

We now look at some more graphical features: contour and image plots.

```
x <- seq(-pi, pi, len=50)
y <- x
```

x is a vector of 50 equally spaced values in $-\pi \leq x \leq \pi$. y is the same.

```
f <- outer(x, y, function(x, y){cos(y)/(1 + x^2)})
```

f is a square matrix, with rows and columns indexed by x and y respectively, of values of the function $\cos(y)/(1 + x^2), \cos(y)/(1 + x^2)$.

```
oldpar <- par(no.readonly = TRUE)
par(pty="s")
```

Save the plotting parameters and set the plotting region to "square".

```
contour(x, y, f, nlevels=15)
```

Make a contour map of f.

```
dev.copy2eps()
```

This will produce a postscript file named `Rplot.eps` (default name) containing a copy of the current active output device (in this case the contour map of f).

```
par(oldpar)
```
Restore the old graphics parameters.

```
image(x, y, f)
image(x, y, fa)
```
Make some high density image plots. Currently, this may not work on our SUN machines...

```
objects(); rm(x, y, f, fa)
```
...and clean up before moving on.

R can do complex arithmetic, also.

```
th <- seq(-pi, pi, len=100)
z <- exp(1i*th)
```
`1i` is used for the complex number i.

```
par(pty="s")
plot(z, type="l")
```
Plotting complex arguments means plot imaginary versus real parts. This should be a circle.

```
w <- rnorm(100) + rnorm(100)*1i
```
Suppose we want to sample points within the unit circle. One method would be to take complex numbers with standard normal real and imaginary parts ...

```
w <- ifelse(Mod(w) > 1, 1/w, w)
```
...and to map any outside the circle onto their reciprocal.

```
plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+",xlab="x", ylab="y")
lines(z)
```
All points are inside the unit circle, but the distribution is not uniform.

```
w <- sqrt(runif(100))*exp(2*pi*runif(100)*1i)
plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+", xlab="x", ylab="y")
lines(z)
```
The second method uses the uniform distribution. The points should now look more evenly spaced over the disc.

```
rm(th, w, z)
```
Clean up again.

In some situation it may be appropriate to make a density estimate based on a scatter plot and display this as a contour plot.

`library(MASS)`
: Include a standard R library with functions needed in the following.

`par(mfrow=c(1,2))`
: Split the output device in two.

```
x <- rnorm(1000)
y <- rnorm(x,mean=x)
plot(x,y,pch=3)
```
: Generate vector x of 1000 standard normals and vector y so that y_i is normal distributed with mean x_i and variance one.

`dd=kde2d(x,y)`
: Generates a smoothed estimate of the density for the iid $(x_1, y_1), \ldots (x_{1000}, y_{1000})$.

`contour(dd,add=T)`
: Adds contours to the image

`contour(dd)`
: Generates a seperate contourplot

`q()` Quit the R program. You will be asked if you want to save the R workspace, and for an exploratory session like this, you probably do not want to save it.