R: An Introductory Session

The following session is intended to introduce to you some features of the R environment: the best way to learn R is by using it. Many features of the system will be unfamiliar and puzzling at first, but this will soon disappear. Also, although you may not yet understand all the statistical details, the plan is to type the commands and see what happens as a result.

The left column gives commands; the right column gives brief explanations and suggestions.

Login to R by clicking on the `intro.RData` file after you have downloaded it from MyUni.

`ls()`	Obtain a listing of all objects currently available to your session.
`library(MASS)`	This command makes the datasets from Venables and Ripley available.
`help.start()`	Start the on-line help facility. Alternatively, use the `help` drop down window as described in the Computing Guide.
`help(help)` `?help`	This is another way to get help. Read the help page about how to use help.
`cystfibr`	Typing the name of an object causes it to be displayed. (This is not a good idea for very large datasets.) The `cystfibr` data frame has 25 rows and 10 columns; it contains lung function data for cystic fibrosis patients 7–23 years old.

`is.data.frame(cystfibr)`
`length(cystfibr)`
`attributes(cystfibr)`
	We determine the `type` of an object and its attributes (names of variables or components, etc) by using commands like this.

`cystfibr$fev1`	The $ symbol allows us to refer to individual variables without 'attaching' the dataframe. `fev1` is a numeric vector of measurements on forced expiratory volume.

Some graphical features in R:

`x <- rnorm(1000)` `y <- rnorm(1000)`	Generate 1000 pairs of normal variates.
`truehist(c(x, y+3), nbins=25)`	Histogram of a mixture of normal distributions. Experiment with the number of bins (25) and the shift (3) of the second component.
`?truehist`	Read about the optional arguments.
`contour(dd <- kde2d(x,y))`	A two-dimensional density plot.

`image(dd)`	A pseudo-colour plot.
`hull <- chull(x, y)`	Find the convex hull of x and y in the plane.
`plot(x, y)` `polygon(x[hull], y[hull], dens=15)`	Plot the points in the plane, and mark in their convex hull.

`data(hills)`	This dataset is from the MASS library and contains data on record times of Scottish hill races against distance and total height climbed.
`hills`	List the data.
`pairs(hills)`	Show a matrix of pairwise scatterplots.
`objects()`	See which R objects you have created.
`rm(x,y,hull)`	Remove objects no longer needed (i.e., cleanup).

Some basic data manipulation in R:

`> x <- 1`
`> y <- 2`
`> x+y`
`> log(x/(1+3*y))`
	Arithmetic expressions typed into R are evaluated and printed out. Try these examples (the R prompt is included here for clarity).

`> w <- x+y`
`> z <- log(x/(1+3*y))`
`> w`
`> z`
	To save the result of a calculation as a new R object, use the assignment operator <-. Try these examples.

`small.data`	This object is a list containing four components, each of which is a matrix. This structure is typical of microarray data and these are data from 3 slides (each slide represents a different experiment) for 10 genes. Examine the attributes etc of `small.data`. The data objects in an arithmetic expression can be a combination of scalars, vectors or matrices. If they are all matrices of the same dimension, then the expression is evaluated for each component.
`small.data$R` `small.data$Rb`	These are matrices of red foreground and background intensities.
`Red <- small.data$R - small.data$Rb`	We subtract the backgrounds to obtain the background-adjusted intensities.
`Green <- small.data$G - small.data$Gb`	Obtain the green background-adjusted intensities and find the log base 2 intensity ratio M.

```
M <- log(Red/Green, base=2)
M
```

As an exercise, calculate the matrix A of average log intensities

$$A = (\log_2 \mathrm{Red} + \log_2 \mathrm{Green})/2.$$

We will next look more closely at the cystic fibrosis data.

`attach(cystfibr)`	This makes the columns (i.e., variables) available by name.

`par(mex=0.5)`
`pairs(cystfibr, gap=0,`
`  cex.labels=0.9)`
	Obtain pairwise scatterplots between all the variables in the dataset. The arguments `gap` and `cex.labels` control the visual appearance by removing the space between subplots and decreasing the font size; `mex` reduces the interline distance in the margins. `plot(cystfibr)` gives a similar plot.
	Which variables appear to be closely related?

`fm <- lm(pemax~age)` `summary(fm)`	Fit a simple linear regression of `pemax` on `age` (in years) and look at the analysis; `pemax` is a numeric vector of maximum expiratory pressure.
`plot(age, pemax,` `  col="blue")`	Make a standard scatterplot and colour the points blue.
`title("Max expiratory` `  pressure vs age")`	Add a title to the plot.
`abline(fm, lty=3,` `  col="red")`	Add in the red regression line using a different line type; `abline()` is able to extract the information it needs from the fitted regression object.
`lines(lowess(age,pemax),` `col="green")`	Fit a smooth regression curve using a modern regression function, and plot the smoothed line. Use the help facility to learn more about `lowess`.

`plot(fitted(fm),`
`  resid(fm),`
`  xlab="Fitted values",`
`  ylab="Residuals",`
`  main= "Residuals vs Fitted")`
	A standard regression diagnostic plot to check for heteroscedasticity (i.e., unequal variance). Is there any evidence of heterscedasticity here? Guess what all the arguments do.

`qqnorm(resid(fm), main=`
`  "Q-Q Plot")`
`qqline(resid(fm), col="green")`
	A normal scores plot to check for skewness, kurtosis and outliers.

	At any time you can make a hardcopy of the graphics window by clicking on the appropriate box of the window and selecting the `Print` option.
`detach(cystfibr)`	Remove the `cystfibr` data frame from the search list.
`rm(fm)`	Clean up again.

Next we will look at using R for weighted regression with an exercise using simulated data.

`x <- seq(1, 20, 0.5)` `x`	Make $x = (1, 1.5, \ldots, 19.5, 20)$ and list it.
`w <- 1 + x/2` `y <- x + w*rnorm(x)`	`w` will be used as a 'weight' vector and to give the standard deviations of the errors.

`dum <- data.frame(x,y,w)`
`dum`
`rm(x,y,w)`
	Make a data frame of three columns named `x`, `y` and `w`, and look at it. Remove the original `x`, `y` and `w`.

`fm <- lm(y~x, data=dum)` `summary(fm)`	Fit a simple linear regression of `y` on `x` and look at the analysis.
`fm1 <- lm(y~x, data=dum, weight=1/w^2)` `summary(fm1)`	Since we know the standard deviations, we can do a weighted regression.
`library(modreg)`	This is a library of nonparametric smoothing functions, including splines.
`lrf <- loess(y~x, dum)`	Fit a smooth (nonparametric) regression curve. This is like fitting many regressions over the x- range and 'linking' them together. `loess` is a newer function than `lowess`.
`attach(dum)` `plot(x,y)`	
`lines(spline(x, fitted(lrf)), col=2)`	Add in the local regression curve using a spline interplotation between the calculated points.
`abline(0,1,lty=3,col=3)`	Add in the true regression line (intercept 0, slope 1) with a different line type and colour.
`abline(fm, col=4)`	
`abline(fm1, lty=4,col=5)`	Finally add in the weighted regression line. This one should be the most accurate estimate, but may not be of course.
`detach(dum)`	Clean up again.
	An an exercise, obtain the standard diagnostic plots as before.
`q()`	Quit R.

Patty Solomon

July 2004