A short course in Longitudinal Data Analysis
ESRC Research Methods Short Course

Lab 1 - June 2011

1 Getting started

Create a folder `LDA` in `My Documents`, where you will store all the files for the course. Download the files

```
statmod-1.4.10.zip
joineR.zip
```

from the web-site `www.lancs.ac.uk/staff/diggle/lda`. These files contain `R` packages needed for the lab sessions.

Open `R`, click on "packages," select "Install package(s) from local zip files ..." and install the `statmod` and `joineR` packages.

Now load the package `joineR`, using the `R` command.

```
> library(joineR)
```

All functions in `R`, and in any loaded package, have a help page, which can be accessed with the command `?function.name`, for example:

```
> ?data
Data Sets
Description:
     Loads specified data sets, or list the available data sets.
> ?plot
plot                package:graphics                R Documentation
Generic X-Y Plotting
Description:
     Generic function for plotting of R objects.  For more details
     about the graphical parameter arguments, see 'par'.
...
```

2 Introduction to R

Sections 2.1 to 2.3 are intended for those of you who are unfamiliar with the R software. You can copy and paste commands from this file (`lab1.pdf`) on the web-page `www.lancs.ac.uk/staff/diggle`. Note that the `>` symbol at the beginning of each line is the `R` prompt. Experienced `R` users can begin at section 3.

2.1 Assignments

To assign the value 2 to the variable x, enter

```
> x <- 2
```

The two characters `<` and `-` should be read as one symbol, referrred to as the *assignment operator*. You can also use the `=` sign rather than `<-`, but I prefer not to do so, to rmeind myself that assignment is not the same as logical equality. The logical equality operator in `R` is `==`. Othjer logcial operators include `<`, `>`, `<=` (less than or equal to) and `!` (not). For example:

```
> x<-5
> y<-7
> x==y
[1] FALSE
> x<y
[1] TRUE
> !(x<y)
[1] FALSE
> (2*x)<y
[1] FALSE
```

In the `R` console, notice the effect of typing

```
> x
```

To calculate $3x^4 - 8$, enter

```
> 3*x^4 - 8
```

2.2 Vectorised arithmetic

The construct c(...) is used to define vectors. Enter the following.

```
> celsius <- c(18,24,22,23,27,25)
> celsius
[1] 18 24 22 23 27 25
> fahrenheit <- celsius*9/5 + 32
> fahrenheit
[1] 64.4 75.2 71.6 73.4 80.6 77.0
```

Notice that the operation is carried out element-wise. The function `sum()` adds together the elements of the vector, `length` reports the number of elements in the vector. Hence, one way to calculate the mean temperature in Celsius is the following.

```
> sum(celsius)
[1] 139
> length(celsius)
[1] 6
> xbar <- sum(celsius)/length(celsius)
> xbar
[1] 23.16667
```

However (and as one might expect), a sample mean function is already built into the program

```
> mean(celsius)
[1] 23.16667
```

To calculate the standard deviation

```
> sd(celsius)
[1] 3.060501
```

Note that names are case sensitive, 'x' and 'X' do not refer to the same variable.

2.3 Importing data and data frames

The package `joineR` contains the two data sets we will be analysing during the course of the lab sessions. The first of these, `mental`, is a set of data from a placebo-controlled trialof drug therapies for schizophrenia. To read-in the data:

```
> data(mental)
```

In this case, importing the data set is rendered a trivial exericse given that the data is contained within the 'joineR' package. Of course, generally this will not be the case. Typically the user will be required to import the data from a directory using the `read.table` function to create a data frame. To import the schizophrenia trial data from a plain text file, download the file `mental.txt` from the web-site `www.lancs.ac.uk/staff/diggle` and enter the following:

```
> mental<-read.table(file="mental.txt",header=T)
```

The first argument specifies the name of the file where the data are stored (assuming in this example that this is the same directoy as the one from which you have invoked `R`). Setting 'header=T' specifies that the names in the first row of the file `mental.txt` are to be used as the column names of the data-frame `mental`.

The command 'dim', an abbreviation for dimension, reports the number of rows and columns in the table.

```
> dim(mental)
[1] 150  11
```

Look at the first 6 rows of data. Each row corresponds to a single patient in the study.

```
> mental[1:6,]
  id Y.t0 Y.t1 Y.t2 Y.t4 Y.t6 Y.t8 treat n.obs surv.time cens.ind
1  1   55   NA   NA   NA   NA   NA     2     1     0.704        0
2  2   44   NA   NA   NA   NA   NA     1     1     0.740        0
3  3   65   67   NA   NA   NA   NA     2     2     1.121        1
4  4   64   56   NA   NA   NA   NA     2     2     1.224        1
5  5   47   48   NA   NA   NA   NA     2     2     1.303        0
6  6   73   81   NA   NA   NA   NA     2     2     1.541        1
```

The row arguments go to the left of the comma, column arguments to the right. The above command extracts the data in the first 6 rows. By leaving the column argument blank, the data across all columns are extracted.

Extract observations at time zero for the first 6 individuals.

```
> mental$Y.t0[1:6]
[1]  55 44 65 64 47 73
```

or

```
> mental[1:6,2]
[1] 55 44 65 64 47 73
```

The corresponding treatments for these 6 patients are extracted by

```
> mental$treat[1:6]
[1]  2 1 2 2 2 2
```

or

```
> mental[1:6,8]
[1]  2 1 2 2 2 2
```

To view the respones at all times for the first 6 individuals

```
> mental[1:6,2:7]
 Y.t0 Y.t1 Y.t2 Y.t4 Y.t6 Y.t8
1   55   NA   NA   NA   NA   NA
2   44   NA   NA   NA   NA   NA
3   65   67   NA   NA   NA   NA
4   64   56   NA   NA   NA   NA
5   47   48   NA   NA   NA   NA
6   73   81   NA   NA   NA   NA
```

3 Schizophrenia trial data

The `mental` data set was obtained from a clinical trial comparing three different therapies in the treatment of chronic schizophrenia. Data are available from 150 patients, randomly allocated to each treatment group, treatment 1, treatment 2 and treatment 3. Treatment 3 is considered the new treatment while treatment 1 is the standard therapy, and treatment 2 is placebo. The primary response variable (Y) was the total score obtained on a symptom rating scale, a measure of psychiatric disorder.

The study design measured patients at weeks 0, 1, 2, 4, 6 and 8, where time 0 is the baseline week. A reduction of 20% in the mean score is considered a worthwhile clinical improvement.

Drop-out from the study for causes related with the longitudinal profile is the event of interest. Of 150 patients, 68 completed the study, and from those that dropped-out 63 are known to have dropped-out owing to inadequate response. This is assumed to be informative.

3.1 Schizophrenia trial data set

The command to use to load data sets of the package is `data()`, with the name of the data set as its argument. This produces a data frame where elements from row `n.row` and column `n.col` are called with notation `object[n.row,n.col]`.

If you have not done so already, load the 'joineR' package and read-in the `mental` data set

```
> library(joineR)
> data(mental)
> mental[1:6,]
  id Y.t0 Y.t1 Y.t2 Y.t4 Y.t6 Y.t8 treat n.obs surv.time cens.ind
1  1   55   NA   NA   NA   NA   NA     2     1     0.704        0
2  2   44   NA   NA   NA   NA   NA     1     1     0.740        0
3  3   65   67   NA   NA   NA   NA     2     2     1.121        1
4  4   64   56   NA   NA   NA   NA     2     2     1.224        1
5  5   47   48   NA   NA   NA   NA     2     2     1.303        0
6  6   73   81   NA   NA   NA   NA     2     2     1.541        1
```

The elements labelled `NA` correspond to data that are `Not Available` for specific individuals at specific time points. The names of a data frame (`names()`) are the column names of the object. For the schizophrenia data set, the first column `indv` is the individual identification, the following six columns are the measurements of the response variable `Y` at times 0,1,2,4,6 and 8, respectively. Columns 8 to 11 correspond to 'treatment', 'censoring indicator', 'number of observations' and 'survival time', respectively.

```
> names(mental)
 [1] "id"        "Y.t0"      "Y.t1"      "Y.t2"      "Y.t4"      "Y.t6"
 [7] "Y.t8"      "treat"     "n.obs"     "surv.time" "cens.ind"
```

The number of individuals that completed the study is equivalent to the number of individuals that have 6 observations, i.e. `n.obs` equals 6,

```
> length(mental$id[mental$n.obs == 6])
[1] 68
```

whereas the number of subjects that drop-out of the study for an informative reason have censoring indicator 1 (failure),

```
> length(mental$id[mental$cens.ind == 1])
[1] 63
```

The notation $ is used to call a column in a data frame, or an element in a list. There are 50 subjects in each treatment group as

```
> length(mental$id[mental$treat == 1])
[1] 50
```

and similarly for treatments 2 and 3 (try this yourself).

3.2 Reformatting the data

The data is currently formatted as one row per individual. However, for the analysis to follow, we re-format the data to one row per observation, with columns identifying the individual and the time of the measurement. The function `to.unbalanced` performs the change in format. The arguments of any function are obtained with `args( )`.

```
> args(to.unbalanced)
function (data, id.col, times, Y.col, other.col = NA)
NULL
> mental <- to.unbalanced(data = mental, id.col = 1, times = c(0,
     1, 2, 4, 6, 8), Y.col = 2:7, other.col = 8:11)
> mental[1:10, ]
   id time Y.t0 treat n.obs surv.time cens.ind
1   1    0   55     2     1     0.704        0
2   1    1   NA     2     1     0.704        0
3   1    2   NA     2     1     0.704        0
4   1    4   NA     2     1     0.704        0
5   1    6   NA     2     1     0.704        0
6   1    8   NA     2     1     0.704        0
7   2    0   44     1     1     0.740        0
8   2    1   NA     1     1     0.740        0
9   2    2   NA     1     1     0.740        0
10  2    4   NA     1     1     0.740        0
> names(mental)
[1] "id"        "time"      "Y.t0"      "treat"     "n.obs"     "surv.time"
[7] "cens.ind"
> dim(mental)
[1] 900   7
```

Note that for the reformatted data, the information contained in the final four columns ('treat', 'n.obs', 'cens.ind', 'surv.time') is repeated for the 6 (including missing values) records on each subject. Also, the `Y.t0` name for the third column is a bit confusing, since it now applies to all follow-up times t, not just $t = 0$. If you wish you can re-name it as follows.

```
names(mental)[3]<-"Y"
```

3.3 The lm( ) function

Consider a scatterplot of response versus time across all individuals, superimposing a simple regression line.

```
> plot(mental$time,mental$Y,pch=19,cex=0.5,xlab="time",ylab="reponse",
main="mental data response vs time")
fit<-lm(mental$Y~mental$time)
abline(fit)
```

A summary of the fitted regression model is obtained as follows – but note that some of the informaiton given is untrustworthy, because the `lm()` function assumes that repeated measurements on the same subject are uncorrelated.

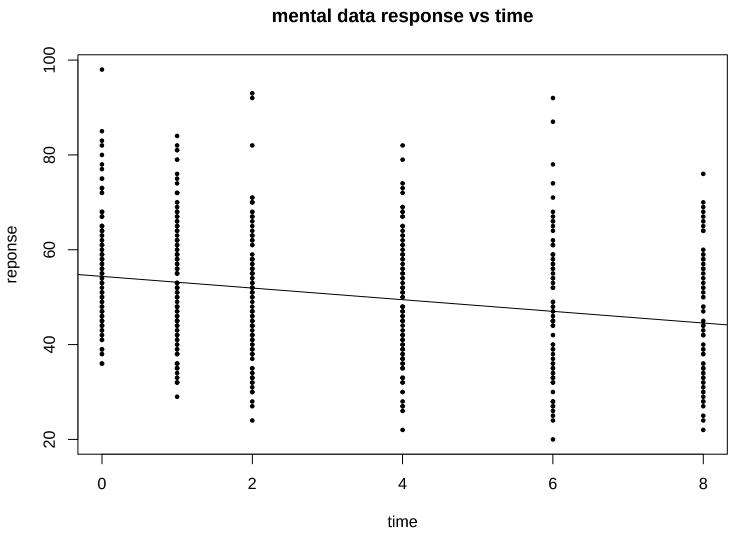

Figure 1: Scatterplot of response versus time across all subjects with superimposed ordinary least squares regression line

```
> summary(fit)
Call:
lm(formula = mental$Y ~ mental$time)

Residuals:
    Min      1Q  Median      3Q     Max
-27.921  -9.921  -0.378   8.993  44.993

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  54.3785     0.7176  75.781  < 2e-16 ***
mental$time  -1.2286     0.1901  -6.463 1.95e-10 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 12.88 on 683 degrees of freedom
  (215 observations deleted due to missingness)
Multiple R-squared: 0.05763,    Adjusted R-squared: 0.05625
F-statistic: 41.77 on 1 and 683 DF,  p-value: 1.953e-10
```

To exit from `R`, type `q()`, but **remember to save your workspace** for the later practical classes.

PJD, 25.05.2011