R / Bioconductor: A Short Course

James H. Bullard
Sandrine Dudoit
Division of Biostatistics, UC Berkeley
www.stat.berkeley.edu/~bullard
www.stat.berkeley.edu/~sandrine

Cuernevaca, Mexico
January 21-25, 2008

Exploratory Data Analysis with R

EDA: Goals

Broadly, exploratory data analysis sets out to accomplish the following tasks[1]

1. gain insight into a data set
2. find hidden structure
3. find the important variables
4. detect outliers
5. test assumptions
6. begin to develop modeling insights

[1] adapted from: here

EDA

- Exploratory data analysis is at the heart of R programming!
- We need to look at our data both graphically and numerically before we can fit models and make tests.
- What common themes in data analysis can we abstract into functions?
- What type of summary statistics should we compute?

We will investigate three “toy” data sets to get familiar with the functions which R provides. These data sets will embody a lot of what we will see in the future, but will be more manageable (hopefully).

EDA: History

In 1801 William Playfair one of the founders of statistical graphics had this to say:

```
For no study is less alluring or more dry and
tedious than statistics, unless the mind and
imagination are set to work or that the person
studying is particualarly interested in the
subject; which is seldom the case with young
men in any rank of life.
```

Playfair helped introduce graphics as a means of communicating information and did much to make statistics more allurring.

Count Data

We often encounter count or categorical data. Often these data need to be handled very differently than continuous data.

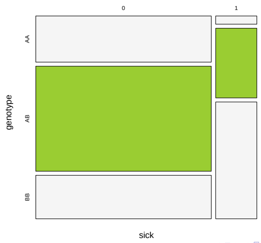
	AA	AB	BB
0	192	437	181
1	8	68	114

Table 1: disease-status and genotype

- What are things we might want to do to count data?

```
> mosaicplot(table(df), main = NULL,
+     col = c("whitesmoke", "yellowgreen"))
```

Count Data



Continuous Data

	viral.load	age	meds	infected
1	1.52	39.56	3TC	3.40
2	2559747.92	63.93	3TC	1.60
3	155.98	47.31	ddI	1.40
4	2.36	48.63	AZT	3.10
5	7.20	52.15	ddI	1.90
6	0.26	33.35	AZT	2.30

Table 2: viral load

- How do we want to plot these data?
- What types of transformations might we make?
- What statistics do we want to do on the columns? on the rows?

Spatial Data

	x	y	intensity
1	1	1	0.58
2	2	1	1.58
3	3	1	0.38
4	4	1	0.50
5	5	1	0.14
6	6	1	1.37

Table 3: intensity by location

- How do we deal with spatial data?
- What types of plots are relevant here?

Summarizing Vectors: Numerical

- stem, summary, quantile
- mean, median, sd, cor, var, mad
- table : can use tables to perform grouping, “GROUP BY”
- cut : cut vectors into buckets basd on cutpoints
- split : partition data sets into groups based on group membership

```
> stem(rexp(10))

  The decimal point is at the |

  0 | 013
  0 | 55666
  1 | 2
  1 | 9
```

Group means: split

Example

We want to compute group level summary statistics for each of the drug categories in our viral-load data set. What statistics should we compute? I believe that certain columns might be correlated how can we assess this? Test the R functions from the previous slides, what do we find? Make a table of the drug categories, are certain drugs over-represented? Write a function which takes two arguments a list of data.frames and computes a summary statistic on this data frame. The *i*th element of the list will be the corresponding data from all subjects who took drug *i* (lapply, levels, and split will be useful).

The R Graphics System(s)

- There are two “graphics systems” in R: Traditional and Grid.
- First, we will focus on the “traditional” system.
  - The traditional system is contained in the package graphics.
  - The traditional system works by calling a high-level plotting function which sets up the plotting window.
  - To add to the plot you call additional functions, like lines, abline, points.

```
> x <- rnorm(100, 10)
> y <- 2 + 3 * x + rnorm(100)
> plot(x, y, pch = 16, cex = 0.3)
> abline(2, 3, col = "red")
```

Devices

- Both R graphics systems are built on top of the grDevices package. Just like the package stats this package is loaded by default and you almost never encounter it directly.
- This package provides functions for printing to other devices besides the screen. In R the screen is almost always the X11 device, except when we are in Windows.
- We have the following: ps, pdf, jpeg, png.
- When we call the dev.off below we turn the active device off. The default device is x11 on Unix and Mac OS X and on windows (or any platform) we can find out by typing: getOption("device").

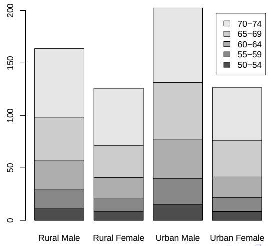
```
> pdf("example%03d.pdf", onefile = FALSE)
> hist(rnorm(1000), breaks = 200)
> dev.off()
```

Plotting Classics

There are a couple of plots like the barplot and pie chart that we see all the time, although there are often significantly "better" plots than these it is important to see them in action
How do we compare the each chunk?

```
> data(VADeaths)
> barplot(VADeaths, legend.text = rownames(VADeaths))
```

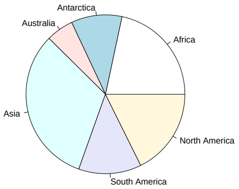
Plotting Classics



Plotting Classics

```
> data(islands)
> pie(islands[c("Africa", "Antarctica",
+     "Australia", "Asia", "South America",
+     "North America")])
```

Plotting Classics



Useful Graphics Functions

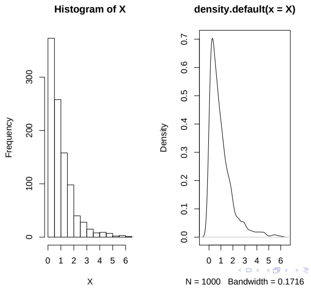
- plot : general plotting function in R
- hist : function for making histograms (important argument: breaks)
- qqplot, qqnorm : plots for comparing quantiles of two distributions
- matplot : function for plotting columns of matrices
- lines, points : draw lines and points on top of the active window
- curve : try: `curve(x2̂)` polygon : a little advanced - see the help
- par : par sets parameters which determine the how things are displayed in the graphics window. Many graphical parameters can be set directly with plotting functions, such as, plot, however, sometimes it is only possible to use par

par

One situation where we need to use par is when we want to put multiple plots on a single page/window

```
> par(mfrow = c(1, 2))
> X <- rexp(1000, 1)
> hist(X)
> plot(density(X))
```

par



piechart and barplot

Example

Make a plot with two “panels” containing a pie chart and a bar plot of the number of subjects in each drug category from the viral load data set. We can also use our drug category summary statistics computed in the previous example to make interesting pie charts and barplots, try to make one of the median viral.load for each drug category. Is there anything interesting to report here? What about a bar plot for our genotype data? Can we make a mosaic plot for our drug data?

plot

- The plot function in `R` can be a whole session of the course, we need to learn just enough to accomplish basic tasks.
- The basic function takes an x and optionally a y as we have already seen `plot(x = x <- seq(-1,1,length = 100), y = dnorm(x))`
- plot is an S3 generic function we can see that plot is defined differently for different objects (remember: `methods(plot)`)

```
> plot(density(rnorm(100)))
> plot(ecdf(rnorm(100)))
```

These plotting functions have been specialized for both the `density`, and `ecdf` classes - what other classes have a specialized plot function?

Working with the viral-load data set

Example

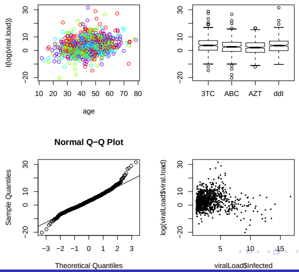
Make some plots of the viral-load data which help understand the relationships between drug type, duration of disease, and viral load (hint: use different plotting symbols, colors)

```
> par(mfrow = c(2, 2))
> cols <- rainbow(4)[as.numeric(viralLoad$meds)]
> plot(I(log(viral.load)) ~ age,
+     data = viralLoad, col = cols)
> boxplot(I(log(viral.load)) ~ meds,
+     data = viralLoad, notch = TRUE)
> qqnorm(log(viralLoad$viral.load))
> qqline(log(viralLoad$viral.load))
```

Working with the viral-load data set

```
> plot(viralLoad$infected, log(viralLoad$viral.load),
+     pch = 16, cex = 0.3)
```

Working with the viral-load data set



- What problems did you encounter presenting the viral load data set?
- What is the distribution of the outcome variable? based on this what type of statistical model might we try to fit?

formulas

We saw in the last example the use of the following construct:

```
> viral.load ~ infected
> class(viral.load ~ infected)
```

- This is a formula object in R, formulas are ways of describing relationships between variables in R. A formula describes a model.matrix, which is essentially a design matrix in addition to the response variable; plot is specialized for the formula object.
- We will see lots more of these in the next section when we talk about statistical models in R.

boxplots

- boxplot : A plotting method for generating Tukey's boxplots.
- Excellent for comparing location shifts of k distributions of varying size.
- Assess skewness and spread of either of one or more distributions.
- Boxplots are often a much better summary of a distribution than are histograms as they do not suffer from either bandwidth choice or the need to have large data sets.

Example

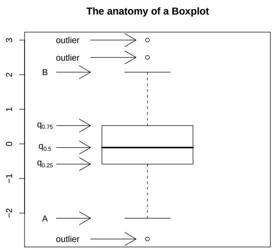
What does skewness look like on a boxplot, spread? can we generate some data to exemplify these things? (hint: remember all of the random number generators which we talked about in the first lecture)

Anatomy of a boxplot

- A, B : lower/upper adjacent values:

$$
\begin{aligned}
r &\triangleq |q_{75} - q_{25}| && (1)\\
A &= \inf\{x_i : x_i > q_{25} - 1.5r\} && (2)\\
B &= \sup\{x_i : x_i < q_{75} + 1.5r\} && (3)
\end{aligned}
$$

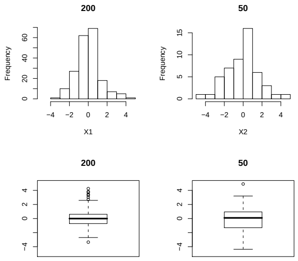
Anatomy of a boxplot



Boxplots Vs. Histograms

Below we have plotted two distributions using both boxplots and histograms. One distribution has 200 points and the other has 50 points - which plotting method makes it easier to determine that they are the same? What kind of plot might even be better than either to compare two distributions?

Boxplots Vs. Histograms



Bimodal Distributions

- Often we have a data a dependent variable y which has a mixture distribution based on some covariate.
- Imagine we have data with the following density:

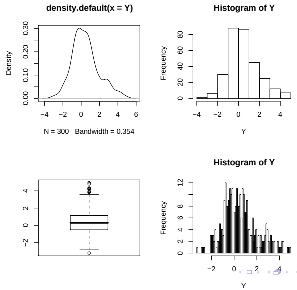
$$y = \pi\phi(\mu_1, \sigma_1) + (1 - \pi)\phi(\mu_2, \sigma_2) \tag{4}$$

- y comes from a mixture of two normals! How can we see this graphically?
- In this case boxplot doesn't help that much, but density estimation and histogram do a better job - be careful though histograms can be misleading based on the number of breakpoints which we choose.

Bimodal Distributions

```
> N <- 300
> Y <- ifelse(rbinom(N, prob = 0.8,
+      size = 1), rnorm(N, 0, 1),
+      rnorm(N, 2.5, 1))
> par(mfrow = c(2, 2))
> plot(density(Y))
> hist(Y, breaks = 10)
> boxplot(Y)
> hist(Y, breaks = 100)
```

Bimodal Distributions



Density Estimation

- We have now seen the density a number of times and we should describe more carefully what the density function does.
- Essentially, kernel density estimation performs a weighted average of points using as a weighting scheme a particular kernel, often the normal kernel.

A kernel density estimator is defined as:

$$f_n(x) = \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{5}$$

K is the kernel and must satisfy (6).

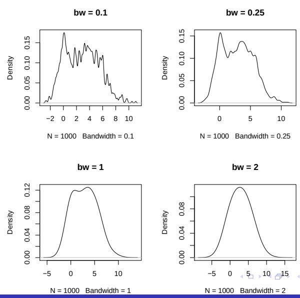
$$\int_{-\infty}^{-\infty} K(x)dx = 1 \tag{6}$$

Density Estimation

h is the bandwidth and determines the smoothness of the estimated function $f_n(x)$. We can visually show the effects of the choice of h.

```
> N <- 1000
> X <- ifelse(rbinom(N, prob = 0.3,
+     size = 1), rnorm(N, 0, 1),
+     rnorm(N, 4, 2))
> par(mfrow = c(2, 2))
> g <- sapply(c(0.1, 0.25, 1, 2),
+     function(bw) {
+         plot(density(X, bw = bw),
+             main = paste("bw =",
+                 bw))
+     })
```

Density Estimation

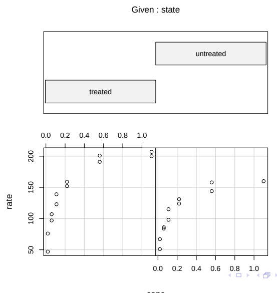


coplots

- We can use the coplot to present the viral load data set - This is a conditional plot popularized by Cleveland
- In addition, this is our first look at plotting functions from the grid graphics system. Although you wouldn't know this just by looking at the plot!
- Can we make a coplot of the viral load data? Put both a coplot and a boxplot on the same plot, what happens?

```
> data(Puromycin)
> coplot(rate ~ conc | state, data = Puromycin)
```

coplots



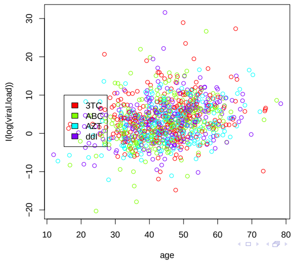
Legends

In order to most clearly present ideas we need to be able to annotate plots, choose colors, and plotting symbols. There are lots of R functions to do this and we'll start off with the basics.

- We use the legend function to add legends to plots

```
> colors <- rainbow(4)[as.numeric(viralLoad$meds)]
> plot(I(log(viral.load)) ~ age,
+     data = viralLoad, col = colors)
> legend(15, 10, legend = levels(viralLoad$meds),
+     fill = rainbow(4))
```

Legends

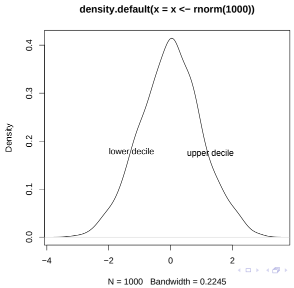


Adding Text

- In order to add text to a plot we use the text.
- In order to add text to the margins of plots we use the function mtext. We will not cover mtext and many more advanced features of the plotting system, the excellent resource "R Graphics" covers these things in great detail.

```
> plot(density(x <- rnorm(1000)))
> qtiles <- quantile(x, prob = seq(0,
+     1, length = 11))
> text(qtiles[2], dnorm(qtiles[2]),
+     "lower decile")
> text(qtiles[10], dnorm(qtiles[10]),
+     "upper decile")
```

Adding Text



Adding Lines/Arrows

- Often we want to add lines or arrows to a plot, we can do this using a number of functions: segments, points, lines, arrows, abline

Example

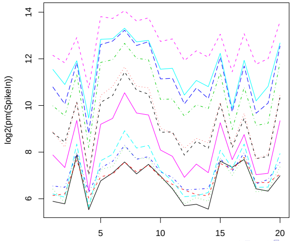
Recreate the normal distribution plot from above, but instead of adding text add arrows pointing to the lower and upper deciles. Label the arrows using the text command. Also, use the segments function to draw vertical lines at the boundaries where we have no data - that is draw vertical lines showing where the kernel density function density has interpolated values.

Plotting Matrices

- We often have matrices where the columns are “repititions” or “trials” and the rows are levels
- A good example is the SpikeIn data set in Affy, this data is data on a single probe set for 12 chips - We can use matplot to simplify the plotting of these data

```
> library(affy)
> data(SpikeIn)
> matplot(log2(pm(SpikeIn)), type = "l")
```

Plotting Matrices

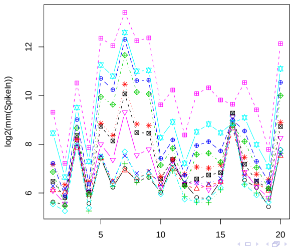


Line Types

- In the previous example we used the “type = l” argument to draw a line instead of points, there are other such “type” specifiers - namely: 'p', 'l', 'b', 'o', 'h', 's'.
- We have the “lty” or line type argument which specifies the line type.
- We have the “pch” argument which represents the plotting symbol.
- Lets try some of the others out!

```
> matplot(log2(mm(SpikeIn)), type = "b",
+     pch = 1:ncol(mm(SpikeIn)))
```

Line Types

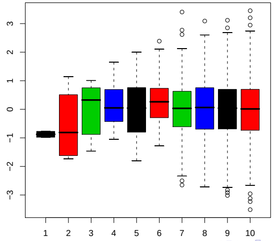


Plotting Arguments

- In the previous example we did `pch = 1:ncol(mm(SpikeIn))`, that is we specified a symbol for each of the `ncol(mm(SpikeIn))` chips, try just `pch = 1:4`
- What happens?
- When we specify plotting arguments we have to realize that they are also recycled. Sometimes this is what you want, other times not!
- In the boxplot below you can use color to indicate a shared covariate between boxplots of the same color.

```
> boxplot(lapply(1:10, function(i) {
+     rnorm(2^i)
+ }), col = 1:4)
```

Plotting Arguments

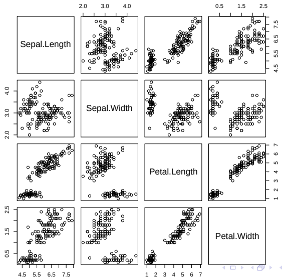


Pairwise Comparisions

- Another important way of looking at "replicates" is via the pairs function.
- This function takes a matrix and constructs a grid of plots making all of the bivariate comparisons.
- Below we have made a pairs plot from the iris data set comparing the sizes of parts of the irises across different species.
  1. Determine what the iris data set contains exactly?
  2. Add colors to the plot below to help see whether or not there are differences between species.

```
> pairs(iris[, 1:4])
```

Pairwise Comparisions

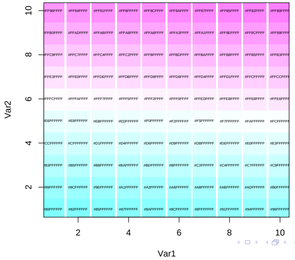


Colors

- R offers a number of tools to work with colors and color scales.
- check out: topo.colors, heat.colors, gray.colors, colors, rainbow, cm.colors

```
> cols <- cm.colors(10^2)[matrix(1:100,
+     nrow = 10)]
> plot(expand.grid(1:10, 1:10), col = cols,
+     pch = 15, cex = 6)
> text(expand.grid(1:10, 1:10), cols,
+     cex = 0.4)
```

Colors



Quick Questions

1. What does expand.grid do?
2. What does `cex = .4` do?
3. If I tell you that there are 25 different plotting symbols from 1 to 25 can you make a similar plot as the one above with the blocks of colors replaced by the plotting symbols?
4. How do we add an x label? a y label? and a title to our plots?

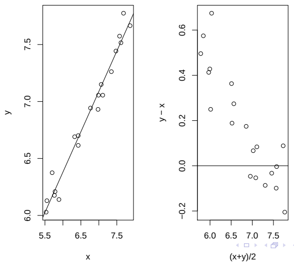
Mean-Difference Plots

- Scatter plots help us determine the relationship between two variables x and y.
- Often however we want to highlight differences.
- After a normalization of two samples we are more interested in highlighting how different two samples x and y are.
- A mean difference plot generally plots the tuple: $((x+y)/2, y-x)$.

Mean-Difference Plots

```
> l2pm <- log2(pm(SpikeIn))
> par(mfrow = c(1, 2))
> plot(l2pm[, 1], l2pm[, 2], xlab = "x",
+     ylab = "y")
> abline(lm(l2pm[, 2] ~ l2pm[, 1]))
> plot((l2pm[, 1] + l2pm[, 2])/2,
+     l2pm[, 2] - l2pm[, 1], xlab = "(x+y)/2",
+     ylab = "y - x")
> abline(0, 0)
```

Mean-Difference Plots

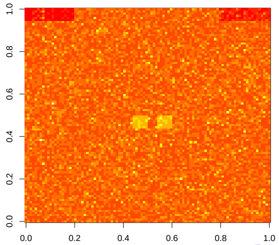


Spatial Data

- We want to return to our spatial data example; this is an example of trivariate data or the tuple (x, y, z) where z represents an intensity and x and y represent points on the grid. The traditional way that these have been presented is using the image function.
- image when you have a discrete space and you are looking for trends in the data or anomolies as we see in the image below they jump right out!

```
> image(matrix(spatial[, 3], nrow = 100),
+     col = heat.colors(10))
```

Spatial Data



More Higher Dimensional Plotting Functions

- In addition to image we have persp, contour

Example

Make a couple of plots using persp and contour of the spatial data. The data is located in “data/spatial.dta” - First, it will be helpful to have a look at the examples and understand what type of objects these functions take.

Smoothing

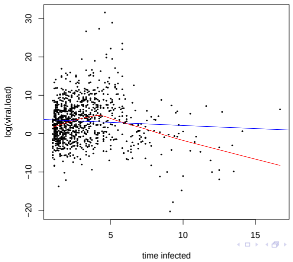
Often it is difficult to see a clear picture because we have too much data, or our eye is drawn to outliers. In these cases it is helpful to look at a “smoothed” versions of the data.

- lowess : lowess, or locally weighted polynomial regression.
- Often lowess helps pick out small trends in data which is not immediately obvious with other smoothed lines.
- Both lowess and loess are functions in R, lowess is older and takes as arguments an x and y, whereas *loess* uses formulas - also the defaults are different which can make a difference in the estimated line.

Smoothing

```
> plot(viralLoad$infected, log(viralLoad$viral.load),
+     pch = 16, cex = 0.3, xlab = "time infected",
+     ylab = "log(viral.load)")
> lines(lowess(viralLoad$infected,
+     log(viralLoad$viral.load)),
+     col = "red")
> abline(lm(I(log(viralLoad$viral.load)) ~
+     viralLoad$infected), col = "blue")
```

Smoothing



Comparing Lowess

Example

In the last plot we saw a slight trend in lowess as we tend towards longer periods of infection. What do we see if we make the mean line? the regression line? the median line? Using the viral load data set do the following:

1. plot the 4 lines in both different line types and different colors.
2. Make the same plots for the each of the different drug categories, is anything surprising?

Other Tools and Techniques

- Hexbin : A Bioconductor function to make two dimensional density plots where we divide the grid into hexagonal bins and use a color or gray scale to portray density in the bin.
- Running Means : A simple technique where nearby values are averaged together to produce a "local" average.
- smooth : To perform different types of running median smoothing.

Literate Programming

- Sweave provides a tool for doing "literate programming" and "reproducible research" in R.
- "Literate programming" is an ideal coined by Donald Knuth - very related to the quote by Knuth: *Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.*
- Sweave was written by Friedrich Leisch, the manual can be found in the resources folder, additionally a paper by Robert Gentleman and Duncan Temple Lang about reproducible research can also be found in the resources folder.

Sweave

- Sweave works by gluing markup (latex) together with code (R) to construct documents - We want the analysis and the document to be one and the same, that way we can easily see that they match up.
- Sweave can generate a latex document or an HTML document.
- In order to work with Sweave we need to know a little `latex` and a little `R`.
- All my lectures, quizes, and homeworks were done using Sweave. All of the example code, answers are embedded in the document. All you have to do to get all the answers is do: `Stangle("slides.Rnw")`.

Nature Medicine (Baggerly 2007)

Recently, Potti et al. published an article in Nature Medicine reporting an approach predicting whether a tumor will respond to chemotherapy. In Microarrays: retracing steps, Baggerly et al. attempt to reproduce their analysis using the same data and code. They find that they are unable to reproduce the results claimed in the paper unless they deviate from the content of the paper. A particular quote is especially telling:

```
We do not believe that any of the errors we found
were intentional. We believe that the paper
demonstrates a breakdown that results from the
complexity of many bioinformatics analyses. This
complexity requires extensive double-checking and
documentation to ensure both data validity and
analysis reproducibility. We believe that this
```

Nature Medicine (Baggerly 2007)

```
situation may be improved by an approach that allows
a complete, auditable trail of data handling and
statistical analysis. We use Sweave, a package that
allows analysts to combine source code (in R) and
documentation (in LaTeX) in the same file.
```

A Simple Example

1. In the "src/Sweave" directory of the course copy the file "simple.Rnw" locally and execute the following commands from within `R`

   ```
   > Sweave("simple.Rnw")
   ```

2. Now from the command line we need to run pdflatex on the generated tex file, this should be as easy as:

   ```
   thales:~ bullard$ pdflatex simple.tex
   ```

3. Now we should be able to open the newly created pdf file.

Look at the directory where we have run things - see how many files have been created! Generally it is a good idea to have a seperate directory for each .Rnw file.