# Illustrating code externalization with knitr

Sarah Gerster (sarah.gerster@isb-sib.ch) and
Frédéric Schütz (frederic.schutz@isb-sib.ch)
Bioinformatics Core Facility
Swiss Institute of Bioinformatics, Lausanne, Switzerland

March 20, 2014

## 1 The concept of code externalization

We discussed thoroughly why it is a good idea of combining R code and reporting/documentation in a single file. The concept of code externalization goes somewhat in the opposite direction. Sometimes, it is easier and/or more convenient to have the main R code in a separate `.R` file.

There are several reasons to consider code externalization. The most important one is probably clarity and manageability of the code: if you have to scroll through too many text parts to find the right place to complete your R code, it can get frustrating. Also, it might be easier to have the main R code in a single file for tuning and testing. Of course, you could always get a `.R` file from the `.lyx` source with a call to the function `purl()`. But then you run into trouble if you start editing this `.R` file instead of the source in the `.lyx` file.

Another important point speaking for code externalization is that it allows to re-use the same R code in different projects. In this sense, it is a bit like sourcing external R scripts. The major difference is that you still organize your code in chunks and benefit from all the chunk options to include/format code/output in your report.

## 2 How code externalization works

The R code is structured in chunks. To do so, you have to annotate the code with comments of the form `## ---- label` or `## @knitr label`. For example, the file `extern.R` contains the following lines to define a chunk `plotdata`:

```
## ---- plotdata ----
dat <- read.table("my_ext_data.csv", header = TRUE, sep = ",")
boxplot(dat)
```

To use the chunk in the report (this file) we first have to read the information in the external file(s) with calls to the function `read_chunk()`:

```
read_chunk("extern.R")
```

We can then evaluate a specific chunk by including a statement of the form

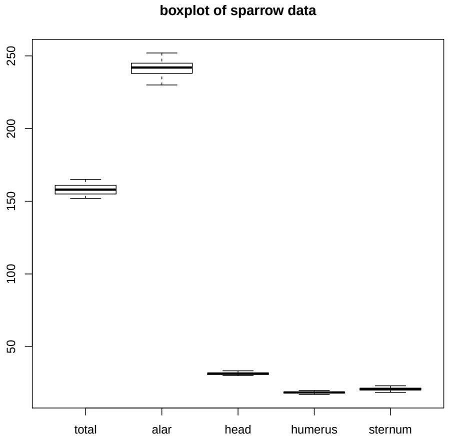
```
<<plotdata, out.width = "0.45\\textwidth">>=
@
```

In other words, for each chunk defined in `extern.R`, we should have a corresponding chunk in the report. You can find more information and examples about code externalization on http://yihui.name/knitr/demo/externalization/

## 3 A brief illustration of the concept of code externalization

All R code for the output below is stored in the file `external.R`. In the explorative analysis below, we simply call the chunks one at a time.

In a first step, we visualize the sparrow data from Bumpus[1] with box plots. The measurements come from 49 birds collected after a severe storm near Brown University in Rhode Island. Birds 1 to 21 survived, while the remainder died (original source Bumpus 1898).



We use principal component analysis (on scaled data) to explore the data set further. We are especially interested in seeing which variable contributes how much to (and with which sign) to the first two principal components.

```
## perform PCA (scale data)
pca_res <- prcomp(sparrow, scale. = TRUE)

## print 'loadings' of first 2 PCs
```

[1] http://www.ndsu.nodak.edu/ndsu/doetkott/introsas/rawdata/bumpus.html

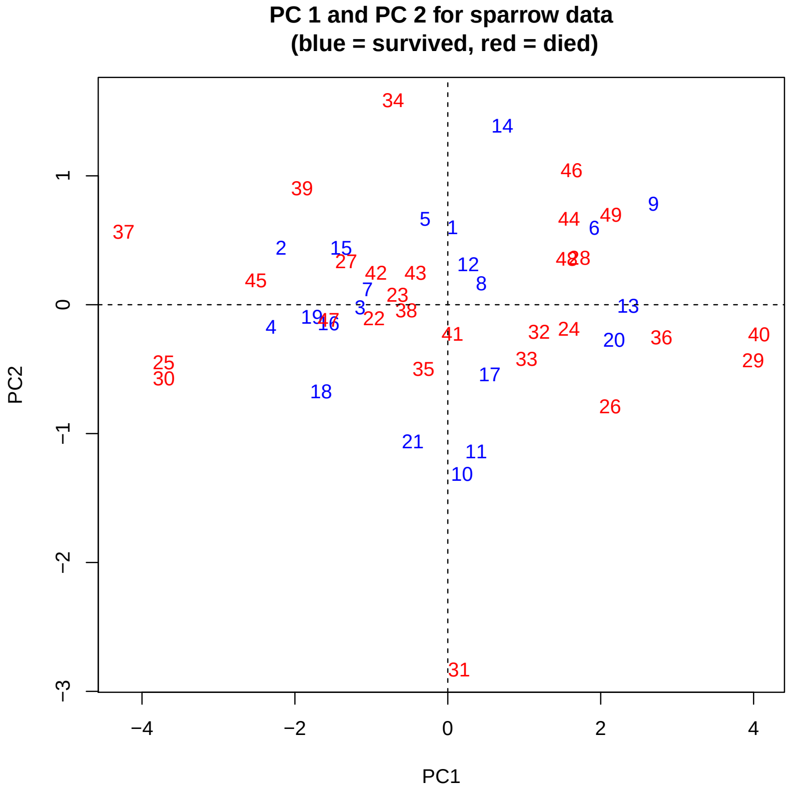
```
require("xtable")

## Loading required package:  xtable
```

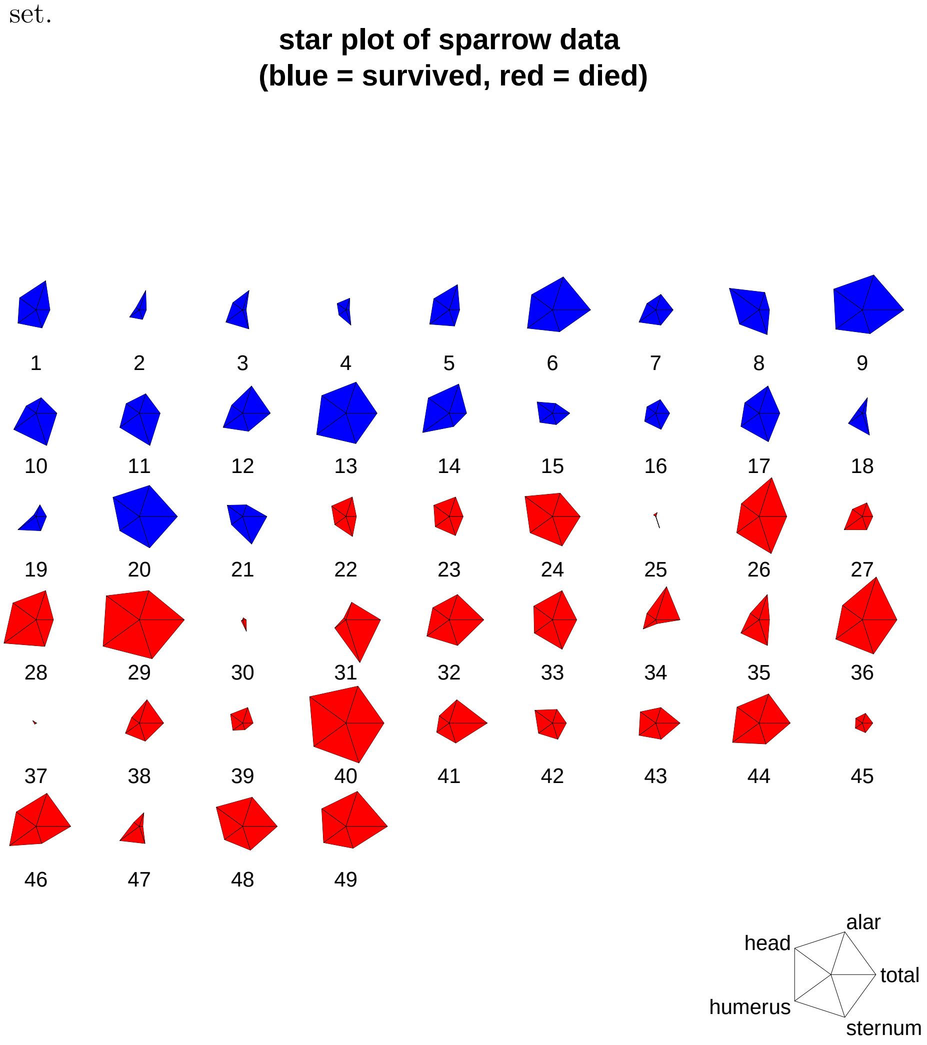
```
print(xtable(pca_res$rotation[, 1:2]), floating = FALSE)
```

| | PC1 | PC2 |
|---|---|---|
| total | 0.45 | -0.05 |
| alar | 0.46 | 0.30 |
| head | 0.45 | 0.32 |
| humerus | 0.47 | 0.18 |
| sternum | 0.40 | -0.88 |

```
## look at biplot, color by dead/alive after storm
plot(pca_res$x[, 1:2], pch = "",
     main = "PC 1 and PC 2 for sparrow data\n (blue = survived, red = died)")
text(pca_res$x[, 1:2], labels = c(1:49),
     col = c(rep("blue", 21), rep("red", 28)))
abline(v = 0, lty = 2)
abline(h = 0, lty = 2)
```



Finally, we also want to look at star plots to complete our brief explorative analysis of the data set.



## R Session information

This document was generated with the following R (packages) version:

- R version 3.0.2 (2013-09-25), `x86_64-pc-linux-gnu`
- Base packages: base, datasets, grDevices, graphics, stats, utils
- Other packages: knitr 1.5, xtable 1.7-1
- Loaded via a namespace (and not attached): evaluate 0.5.1, formatR 0.10, highr 0.3, stringr 0.6.2, tools 3.0.2