STAT3401 Week 1: Lab exercises concerning multivariate eda

Paul Hewson

28th September 2007

1 Multivariate eda

Graphical exploration of data is essential. There are a wide range of techniques for use with multivariate data, this week we examine some of the more common ones. The aim of this week is to introduce you to some of these techniques, and provide pointers for more advanced ones. You should save the code you use to generate these graphs in a short script file for future reference!

2 A correlation paradox

Here's the correlation paradox we referred to. Can you explain what's going on here?

```
> u <- rnorm(100)
> v <- rnorm(100)
> w <- rnorm(100)
> x <- u + v
> y <- u + w
> z <- v - w
> X <- cbind(x,y,z)
> cor(X)
> pairs(X)
```

3 Key eda techniques

First, we need to load some data and obtain the most important summary statistics. We will load data on USArrests as well as irises. As you go through the lab, if appropriate do try to produce graphs for the other dataset.

```
> data(USArrests)
> ?USArrests
> cov(USArrests)
> cor(USArrests)
> colMeans(USArrests)
> data(iris)
> ?iris
```

There are then a few ways of producing the "standard" pairwise scatterplots.

```
> ## The base version
> pairs(USArrests)
> ## The trellis version
> library(lattice)
> splom(USArrests)
```

As well as examining the relationships between variables, we may be interested in the relationship between individuals.

```
> ## Chernoff's faces
> #library(TeachingDemos)
> #faces(USArrests)
> ## a heatmap (more useful if you standardise the variables, try it
> ## on the iris data
> heatmap(as.matrix(USArrests))
> ## a star plot
> stars(USArrests, col.segments = c("red", "green", "blue", "pink"),
+   draw.segments = TRUE, full = FALSE)
> ## parallel co-ordinates plot
> require(MASS)
> parcoord(iris[,-5], col = as.numeric(iris[,5]))
> ## make sure you take put andrews.R in your workspace
> source("andrews.R")
> andrews.curves(iris[,-5], cls = iris[,5])
```

You can see for example that `parcoord` and `andrews.curves` take a data matrix as the first argument (in this case we specify the first four columns of the iris data), and a grouping factor as the second argument (with slightly different syntax).

4 Interactive Graphics

You've been given a short piece of code in the portal which helps draw 3d scatterplots on the `rgl` device. If you put this code in your working directory and "source" it, you should be able to create a simple dynamic 3d scatterplot. Check out `?rgl` if you want any help on the rgl device, in particular to see how you might go about saving an image!

```
> source("rglellipsoid.R")
> scatter3d(iris$Sepal.Length, iris$Sepal.Width, iris$Petal.Width,
+           ellipsoid=TRUE, surface=FALSE,
+           groups=as.factor(iris$Species))
```

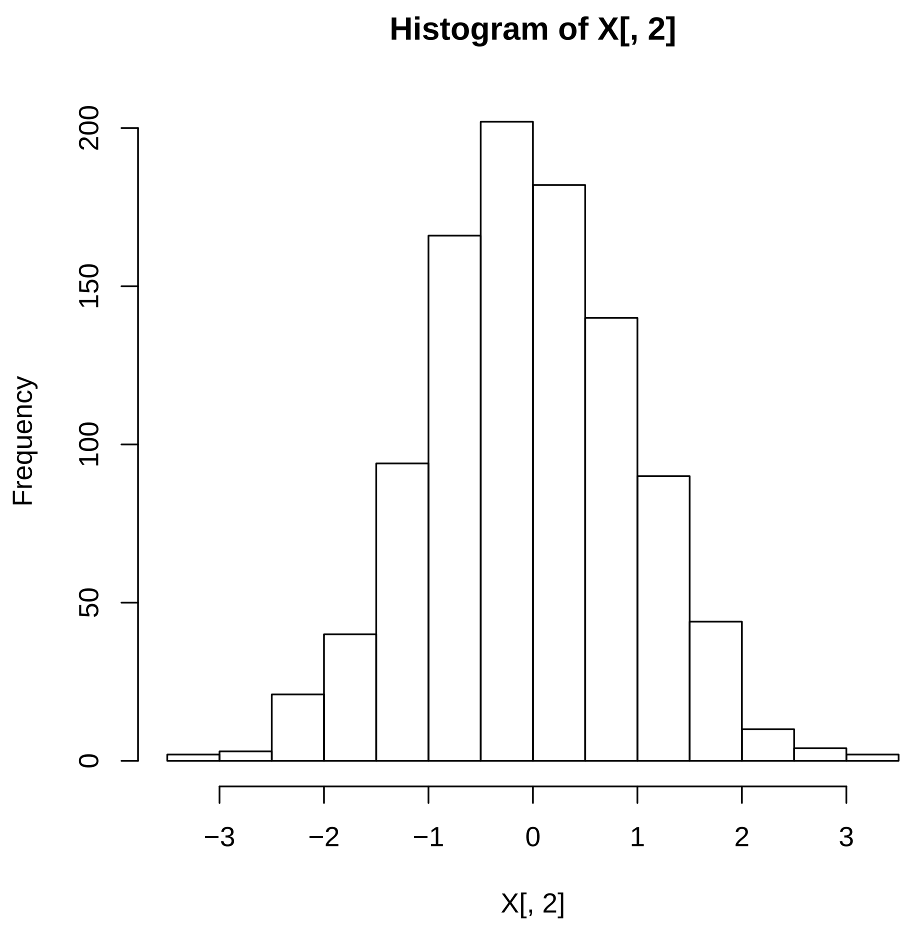
5 Simulating multivariate normal data and linear combinations

First, load the `MASS` library which contains a function `mvrnorm` which simulates multivariate normal data. The first argument to `mvrnorm` is n, the number of variate rows to generate, the second argument is the mean vector to use. Here you see that we want to generate three observations, each with mean zero. Finally, we need to specify the covariance matrix, we have done this on the previous line and stored the results in `covmat`.

```
> library(MASS)
> covmat <- matrix(c(1,0.7,0.7, 0.7,1, 0.7, 0.7, 0.7, 1),3,3)
> X <- mvrnorm(1000, c(0,0,0), covmat )
```

Having simulated some data (stored in the matrix `X`), we are going to check the univariate normality of each margin. The code below produces histograms and plots for the second column, you should check all three!

```
> hist(X[,2])
> qqnorm(X[,2])
> qqline(X[,2], col = "red")
```



Finally, to start thinking about linear combinations, try generating a few linear combinations e.g. $z_i = 2x_{i1} + 0.2x_{i2} + 0.15x_{i3}$ for all $i = 1, \dots, n$.

```
> Z <- 2 * X[,1] + 0.2 * X[,2] + 0.15 * X[,3]
> hist(Z) ## etc.
```

Try different values for the coefficients (i.e replace $2, 0.2 and 0.15$ with other values) and see whether z can be considered to be univariate normal.

6 On your own

Load the following libraries, and check out the following functions and data files:

- Libraries: `library(gclus)`, `library(Flury)`
- Functions: `?cpairs`, `?cparcoord`

- Data: `data(wines)`, `data(turtles)` and `data(flea.beetles)` (in other words, load the data and look at the helpfiles in the same way you check the helpfile for a function)

Do also check out the correlation demo:

```
library(TeachingDemos)
run.cor.examp()
```