

## Saving Graphics Files

There are several standard formats for storing graphics and pictures. The two I use are

- PostScript, for use in  $\text{T}_E\text{X}$
- JPEG, for use in HTML

R can save graphics files in either of these formats.

One way to do this is make the graph the active window, and then to choose `Save as ...` on the `File` menu.

## Saving Graphics Files

(Almost) everything I do in R is from a script (program) that I write prior to execution.

It is annoying to have to point and click on menus.

In the `grDevices` package there is a useful function for saving plots on `windows` devices (i.e., Microsoft Windows).

```
savePlot('ex77110', type="eps")
```

I don't know of such a function for other systems.

## Saving and Using Files in R

Each time you reference an external file in an R session of course you can use the full path name, which is tiresome and requires multiple changes to a saved R script.

R provides help in keeping your work organized into directories.

One way is to set a working directory and then all external files come out of or go into that directory until you change it again.

```
setwd('c:/transfers/csi771/09fall')
```

After this command, the command

```
savePlot('ex77110', type="eps")
```

puts the file `ex77110.eps` in `c:\transfers\csi771\09fall`

## Saving and Using Files in R

If your R program interacts with external files in various directories, setting and resetting the directory is annoying.

An alternative to `setwd` is to define one or more character strings and then use `paste`.

```
datadir <- 'c:/transfers/csi771/data'  
graphdir <- 'c:/transfers/csi771/09fall'
```

Then

```
x <- read.Table(filename=paste(datadir, 'rats.dat', sep=""))  
...  
savePlot(filename=paste(graphdir, 'ex77110', sep=""), type="eps")
```

## Saving and Using Graphics

Let's create a dataset, make a graph, save it, and then use it.

```
setwd('c:/transfers/csi771/09fall')
set.seed(555)
x      <- matrix(rnorm(300),ncol=3)
Sigma <- matrix(c(12,  0, 5,
                  0, 13, 0,
                  5,  0, 1),ncol=3)

x <- x%*%Sigma
plot(x[,1],x[,2])
savePlot('ex77110a',type="eps")
savePlot('ex77110a',type="jpg")
```

## Saving and Using Graphics

Now in a  $\text{T}_{\text{E}}\text{X}$  file in that directory, I use

```
\begin{center}  
\includegraphics [width=5in,height=5in] {ex77110a.eps}  
\end{center}
```

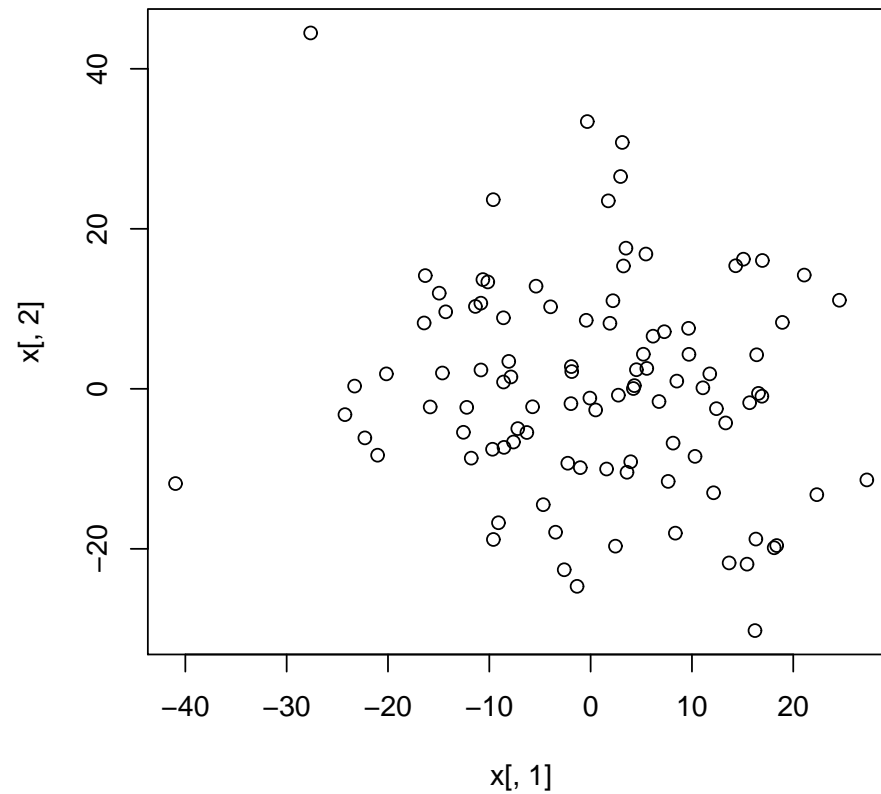
and in an HTML file in that directory, I use

```

```

# Saving and Using Graphics

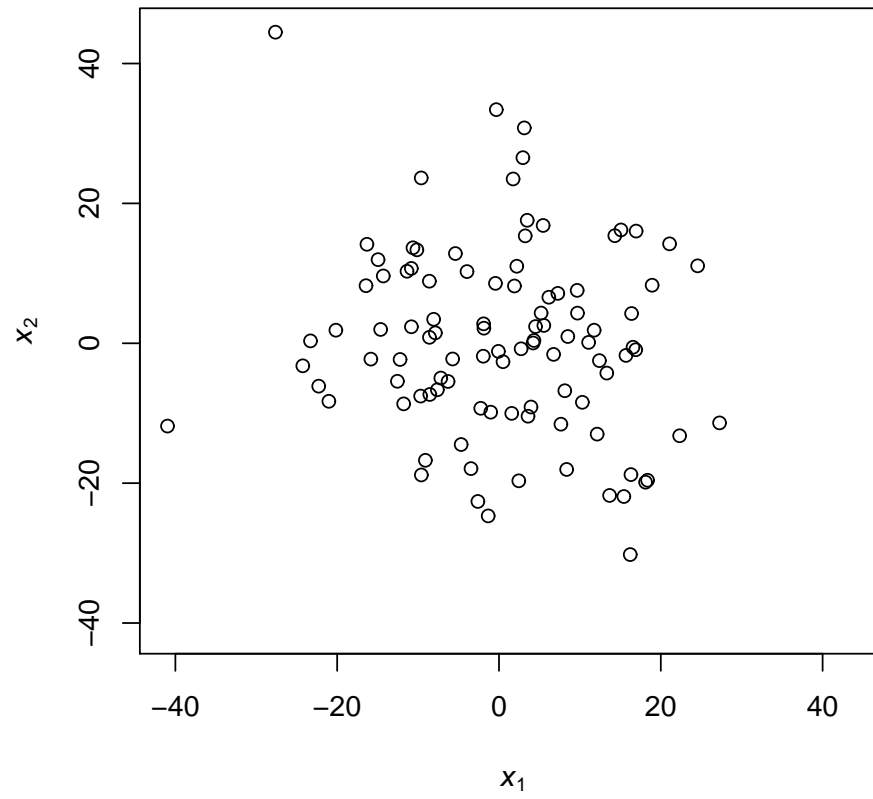
Example. In the  $\text{\LaTeX}$  file that is producing what you're seeing, the next 3 executable lines, which will be invisible, are the three lines on the previous slide.



Let's make the plot as before (but a little better):

```
plot(x[,1],x[,2],xlim=c(min(x[,1],x[,2]),max(x[,1],x[,2])),  
     ylim=c(min(x[,1],x[,2]),max(x[,1],x[,2])),  
     xlab=expression(italic(x)[1]),ylab=expression(italic(x)[2]))  
savePlot('ex77110',type="eps")
```

This yields



For examples, let's use the little dataset created earlier from a  $N_2(0, \Sigma)$  distribution. Recall

$$\textit{Sigma} = \begin{bmatrix} 12 & 5 \\ 5 & 13 \end{bmatrix}$$

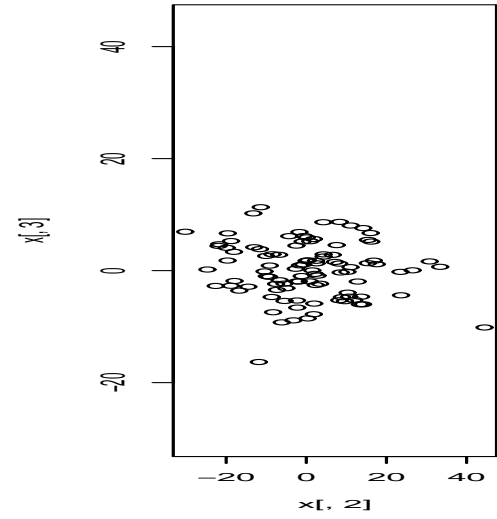
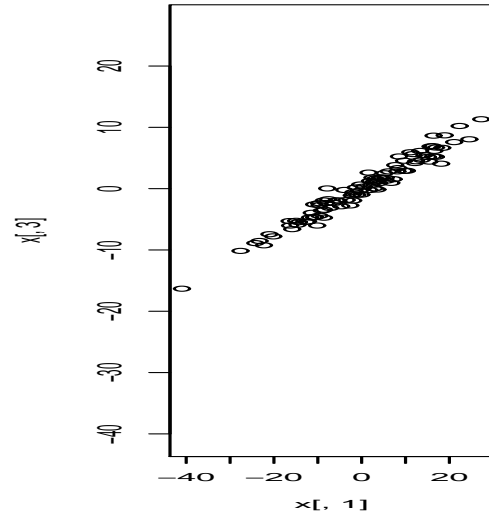
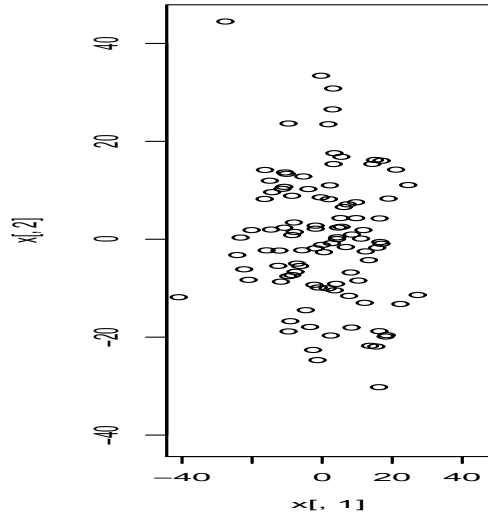
In R, we have

```
round(cov(x))
      [,1] [,2] [,3]
[1,]  163  -24   62
[2,]  -24  178  -8
[3,]   62   -8   25
```

Let's do a "pairs" plot.

```
reset <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(x[,1],x[,2],xlim=c(min(x[,1],x[,2]),max(x[,1],x[,2])),
      ylim=c(min(x[,1],x[,2]),max(x[,1],x[,2])))
plot(x[,1],x[,3],xlim=c(min(x[,1],x[,3]),max(x[,1],x[,3])),
      ylim=c(min(x[,1],x[,3]),max(x[,1],x[,3])))
plot(x[,2],x[,3],xlim=c(min(x[,2],x[,3]),max(x[,2],x[,3])),
      ylim=c(min(x[,2],x[,3]),max(x[,2],x[,3])))
savePlot('ex77120',type="eps")
par(reset)
```

This yields



Another way.

```
pairs(x)  
savePlot('ex77125', type="eps")
```

This yields

