How to make the Tamoxifen graph

First set the working directory, then load the data:

```
library(foreign)
d <- read.dbf('Tamoxifen.dbf')
```

We will be doing calculations separately for each treatment group and therefore define two datasets each containing one of the groups:

```
d1 <- subset(d, Grp==1)
d2 <- subset(d, Grp==2)
```

For convenience I first determine means, SDs, SEs and upper and lower CI bounds. Each of these are saved in a vector (with the same number of elements as there are time points of measurement). One way to do that is to determine - for each group - the means at all time points and glue them together in a vector using `c()`:

```
m1 <- c( mean(d1$c0), mean(d1$c3), mean(d1$c6), mean(d1$c9),
         mean(d1$c18), mean(d1$c24))
m1

## [1]  0.000000  3.962677 15.027769  3.063747  6.165274  7.116543
```

However there is an easier way to determine all these means at once. I use the `sapply()` function that takes as input a data set and the function to be applied to each column of the data set. NB This is advanced - but very convenient.

Here I apply the mean function to each column in d1:

```
m1 <- sapply( d1, mean )
m1

##         X       Grp        c0        c3        c6        c9       c12
## 12.000000  1.000000  0.000000  3.962677 15.027769  3.063747  5.314653
##       c18       c24
##  6.165274  7.116543
```

However we don't want the first two means here (mean of id-variable and group variable) and instead define

```
m1 <- sapply( d1, mean )[3:9]
m1

##        c0        c3        c6        c9       c12       c18       c24
##  0.000000  3.962677 15.027769  3.063747  5.314653  6.165274  7.116543
```

Similar calculations for SD - also for the other group.

```
m2 <- sapply( d2, mean )[3:9]

sd1 <- sapply( d1, sd )[3:9]
sd2 <- sapply( d2, sd )[3:9]
```

Now we can easily determine the SEM and CI for each time point for each group:

```
sem1 <- sd1 / sqrt(21)
sem2 <- sd2 / sqrt(14)



uCI1 <- m1 + 1.96*sem1
lCI1 <- m1 - 1.96*sem1

uCI2 <- m2 + 1.96*sem2
lCI2 <- m2 - 1.96*sem2
```

In this solution I modify the plot arguments several times below, i.e. showing all the commands and the plots generated step by step. Running the commands given in **11.** below produces the plot. In **12.** I further add confidence intervals (this was not part of the exercise).

2.

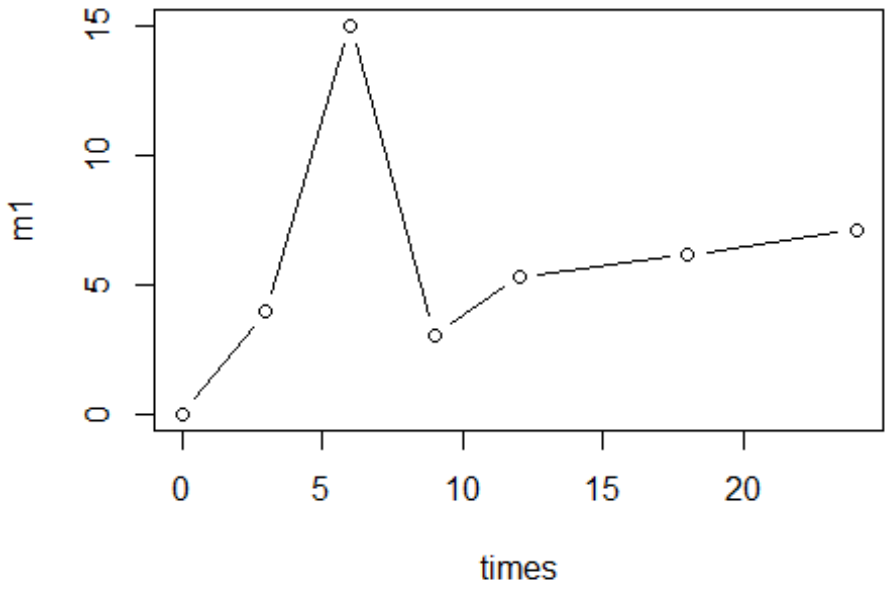
Create a vector with the time points:

```
times <- c(0,3,6,9,12,18,24)
```

3.

Plot the means for the control group, m1, as a function of times using the plot argument `type='b'`.

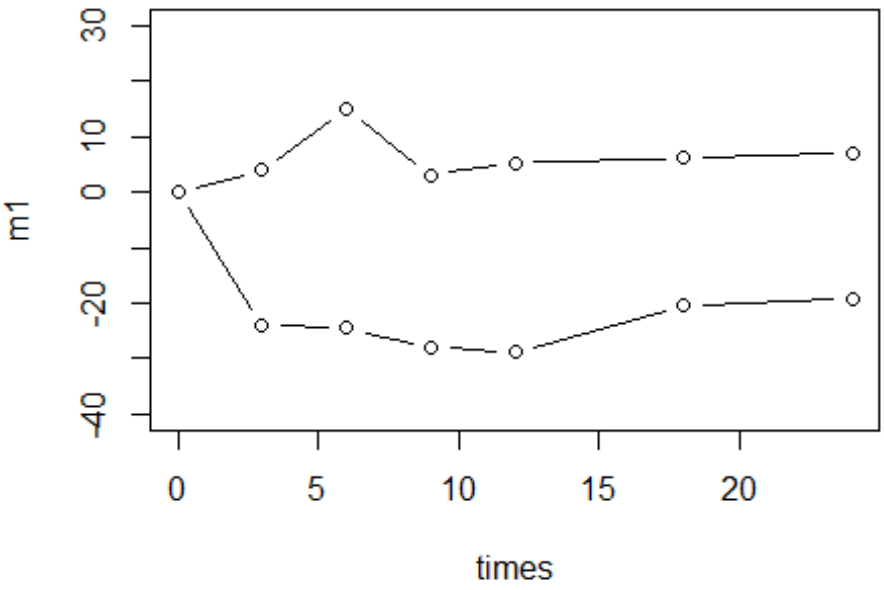
```
plot( times, m1, type='b' )
```



4.

Add a similar curve for the Tamoxifen group to the plot using `lines()`. Notice that the new points are below the y-scale of the plot, so you need to revise the initial plot by setting a suitable `ylim` value.

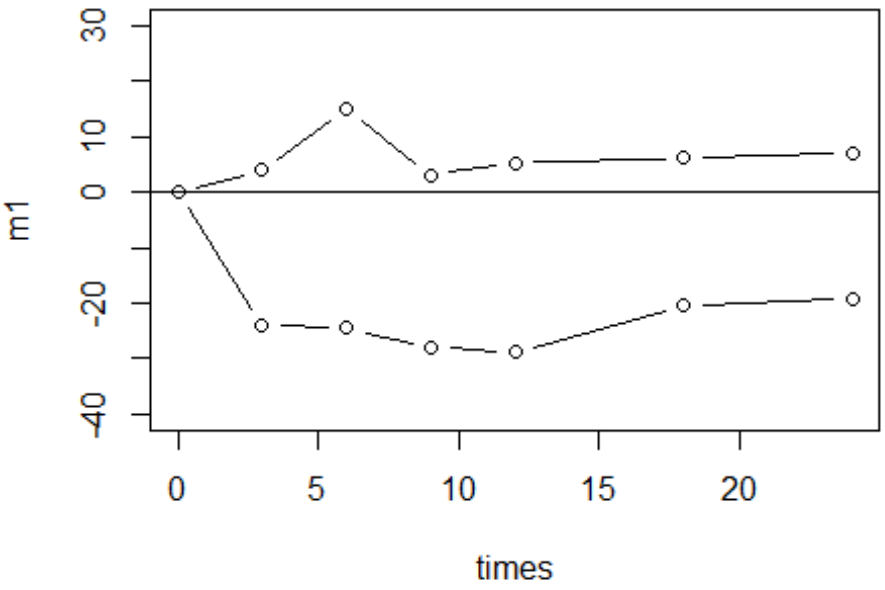
```
plot( times, m1, type='b', ylim=c(-40,30) )
lines( times, m2, type='b' )
```



5.

Add the horizontal line at y = 0 using e.g. `abline()`. abline(a,b) draws a line with intercept specified by a and slope b.

```
plot( times, m1, type='b', ylim=c(-40,30) )
lines( times, m2, type='b' )
abline( 0, 0)
```

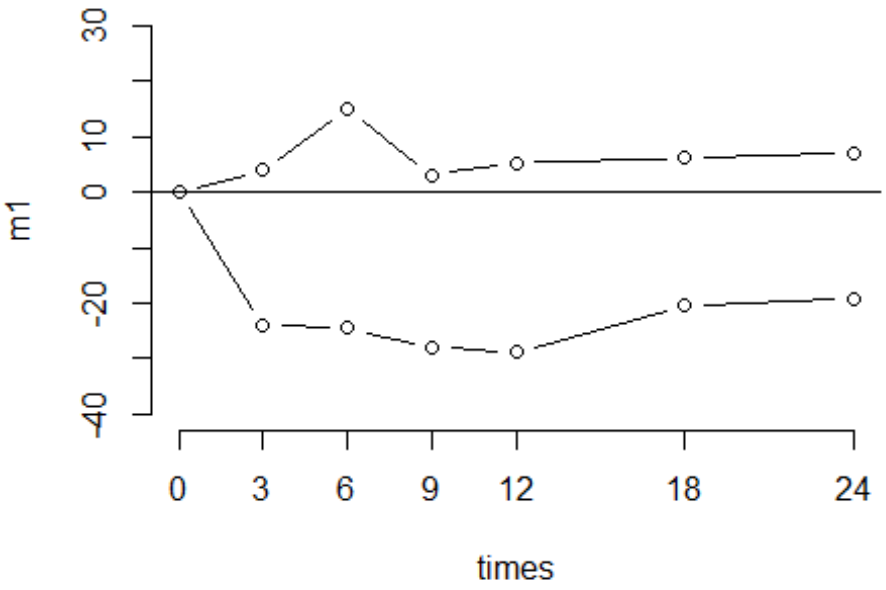


6.

We need a nonstandard x-axis. Modify your initial `plot()` command by adding the argument `axes=F` to tell R not to make the axes. The x-axis is added by afterwards specifying `axis(1, at=c(0,3,6,9,12,18,24) )`

Add the y-axis by simply typing `axis(2)` or add the `at`-argument to control the y-axis also.

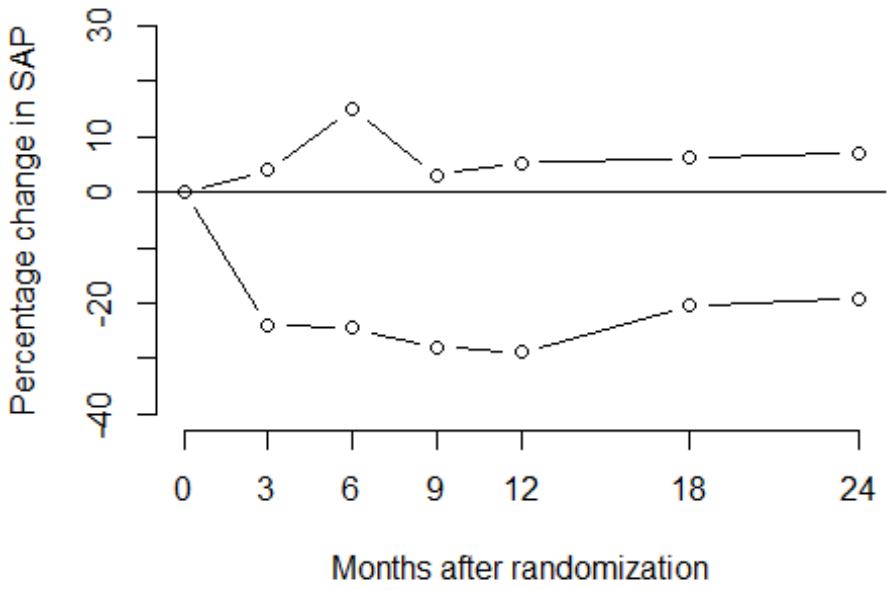
```
plot( times, m1, type='b', ylim=c(-40,30), axes=F )
axis( 1, at=c(0,3,6,9,12,18,24) )
axis( 2 )
lines( times, m2, type='b' )
abline( 0, 0)
```



7.

Use `xlab` and `ylab` arguments in the initial plot call to give better axis labels.

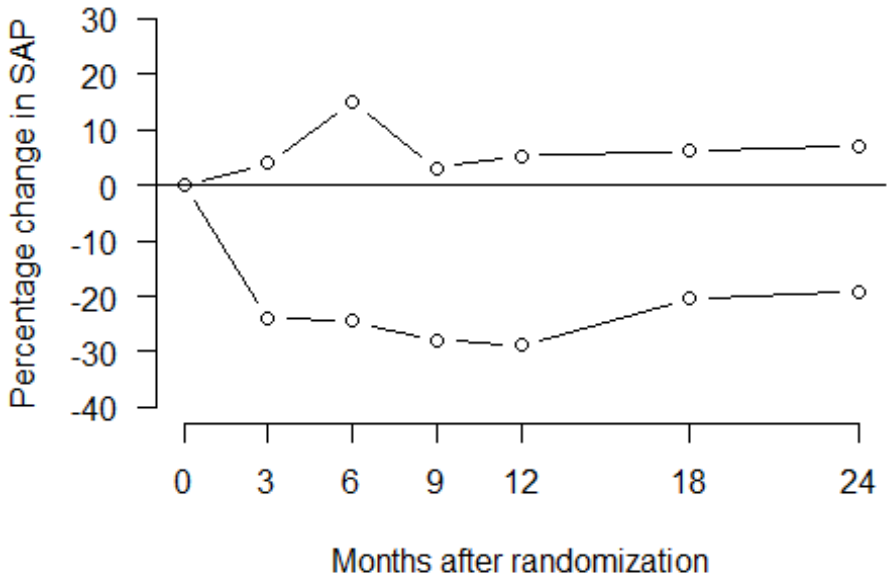
```
plot( times, m1, type='b', ylim=c(-40,30), axes=F,
      xlab='Months after randomization',
      ylab='Percentage change in SAP' )
axis( 1, at=c(0,3,6,9,12,18,24) )
axis( 2 )
lines( times, m2, type='b' )
abline( 0, 0)
```



8.

Rotate the values on the y-axis values by running the command `par(las=1)` before you run your initial `plot()` command.

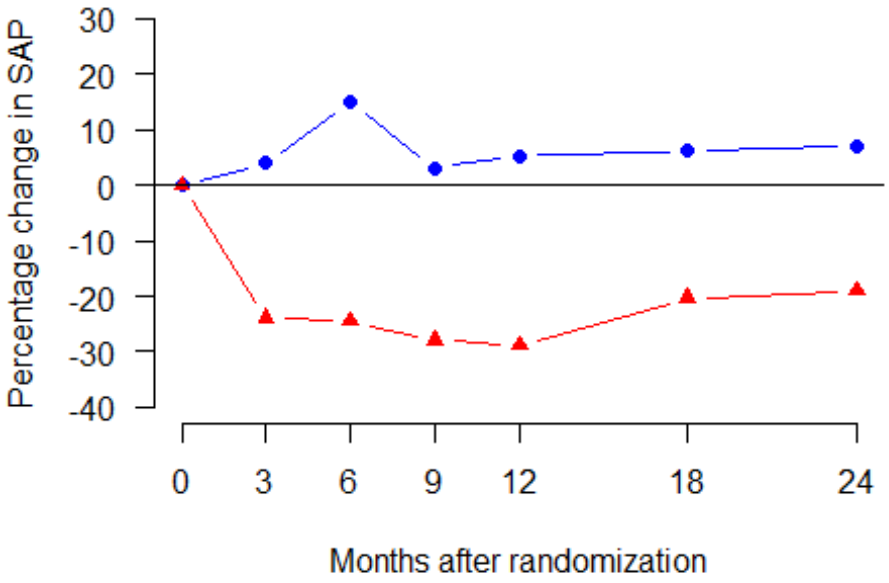
```
par(las=1)
plot( times, m1, type='b', ylim=c(-40,30), axes=F, xlab='Months after
randomization',
     ylab='Percentage change in SAP' )
axis( 1, at=c(0,3,6,9,12,18,24) )
axis( 2 )
lines( times, m2, type='b' )
abline( 0, 0)
```



9.

For each group use different plot symbols, line types, or colors. We here use arguments `col` and `pch`:

```
par(las=1)
plot( times, m1, type='b', ylim=c(-40,30), axes=F,
      xlab='Months after randomization',
      ylab='Percentage change in SAP',
      col='blue', pch=16 )
axis( 1, at=c(0,3,6,9,12,18,24) )
axis( 2 )
lines( times, m2, type='b', col='red', pch=17 )
abline( 0, 0)
```

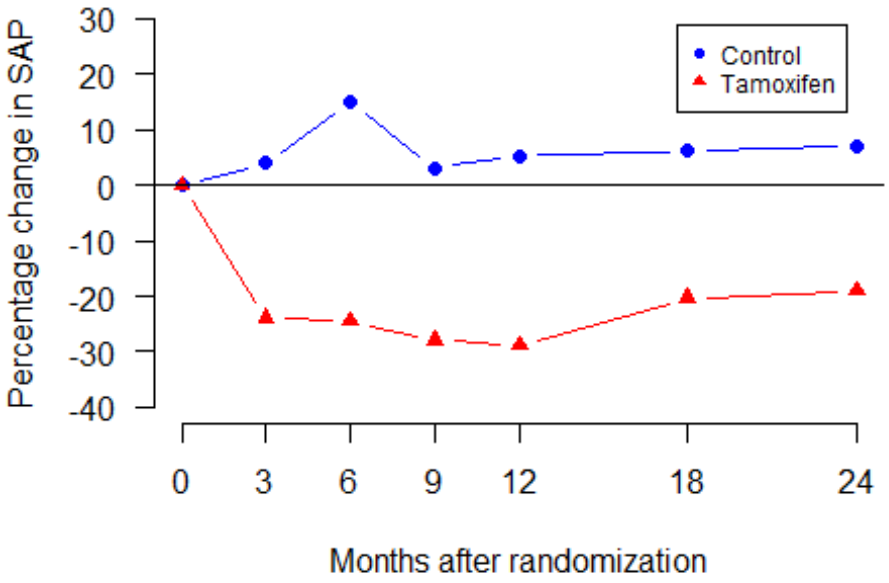


10.

Add a legend, `legend()`, with the plot symbols and group labels. You can use placement 'topright'.

Rstudio sometimes stretches the legend. You can avoid that by using the option `cex=0.5` (or some other number < 1 when using `legend()`). The `cex`-argument controls the size of the text. In the plots generated in this document, `cex=0.75` gives an adequate text size:

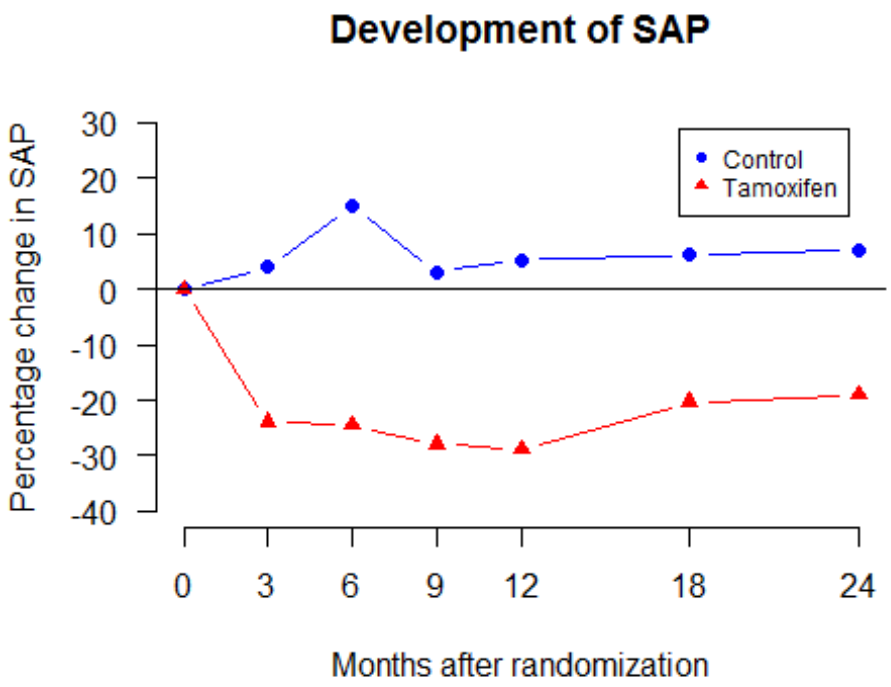
```
par(las=1)
plot( times, m1, type='b', ylim=c(-40,30), axes=F,
      xlab='Months after randomization',
      ylab='Percentage change in SAP',
      col='blue', pch=16)
axis( 1, at=c(0,3,6,9,12,18,24) )
axis( 2 )
lines( times, m2, type='b', col='red', pch=17 )
abline( 0, 0)
legend( 'topright', c('Control','Tamoxifen'), inset=.05,
        pch=c(16,17), col=c('blue','red'), cex=0.75 )
```



11.

Add the title to the plot. We here use the argument `main` in the `plot()`-command:

```
par(las=1)
plot( times, m1, type='b', ylim=c(-40,30), axes=F,
      xlab='Months after randomization',
      ylab='Percentage change in SAP',
      col='blue', pch=16,
      main='Development of SAP' )
axis( 1, at=c(0,3,6,9,12,18,24) )
axis( 2 )
lines( times, m2, type='b', col='red', pch=17 )
abline( 0, 0)
legend( 'topright', c('Control','Tamoxifen'), inset=.05,
        pch=c(16,17), col=c('blue','red'), cex=0.75 )
```

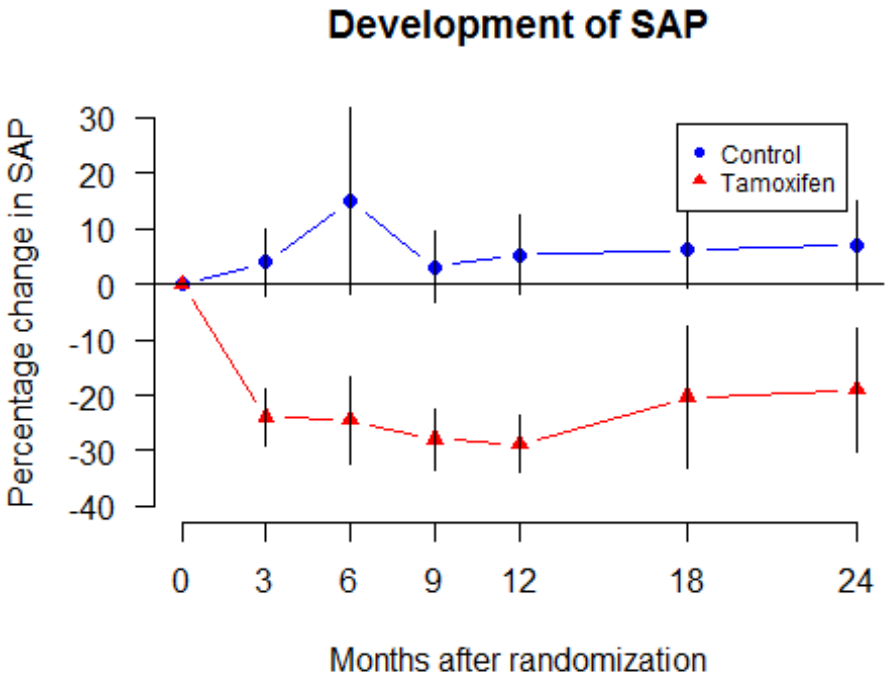


Extra.

This is advanced ...

We can also quite easily - now we have defined the vectors containing the upper and lower CI bounds - add the CIs to the plot using the `arrows()`-command. Use `help(arrows)` to find out how it works.

```
par(las=1)
plot( times, m1, type='b', ylim=c(-40,30), axes=F,
      xlab='Months after randomization',
      ylab='Percentage change in SAP',
      col='blue', pch=16,
      main='Development of SAP' )
axis( 1, at=c(0,3,6,9,12,18,24) )
axis( 2 )
lines( times, m2, type='b', col='red', pch=17 )
abline( 0, 0)
legend( 'topright', c('Control','Tamoxifen'), inset=.05,
        pch=c(16,17), col=c('blue','red'), cex=0.75 )
arrows( times[2:7], lCI1[2:7], times[2:7], uCI1[2:7],
        code=3, length=0.0 )
arrows( times[2:7], lCI2[2:7], times[2:7], uCI2[2:7],
        code=3, length=0.0 )
```


Development of SAP
Percentage change in SAP
30
20
10
0
-10
-20
-30
-40
Control
Tamoxifen
0
3
6
9
12
18
24
Months after randomization