An Introduction to R

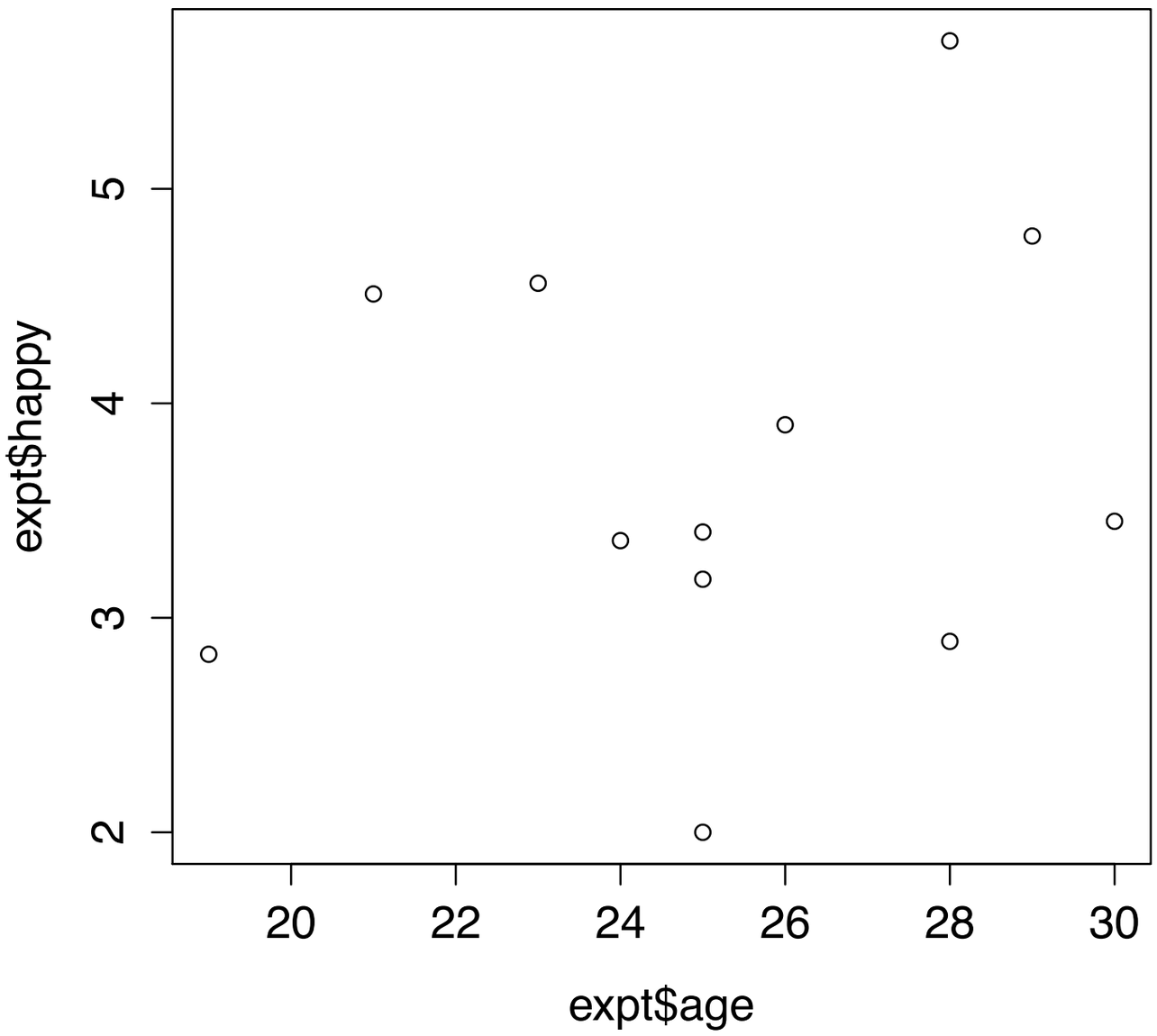
2.2 Statistical graphics

Dan Navarro (daniel.navarro@adelaide.edu.au)
School of Psychology, University of Adelaide
ua.edu.au/ccs/people/dan
DSTO R Workshop, 29-Apr-2015

Scatter plots

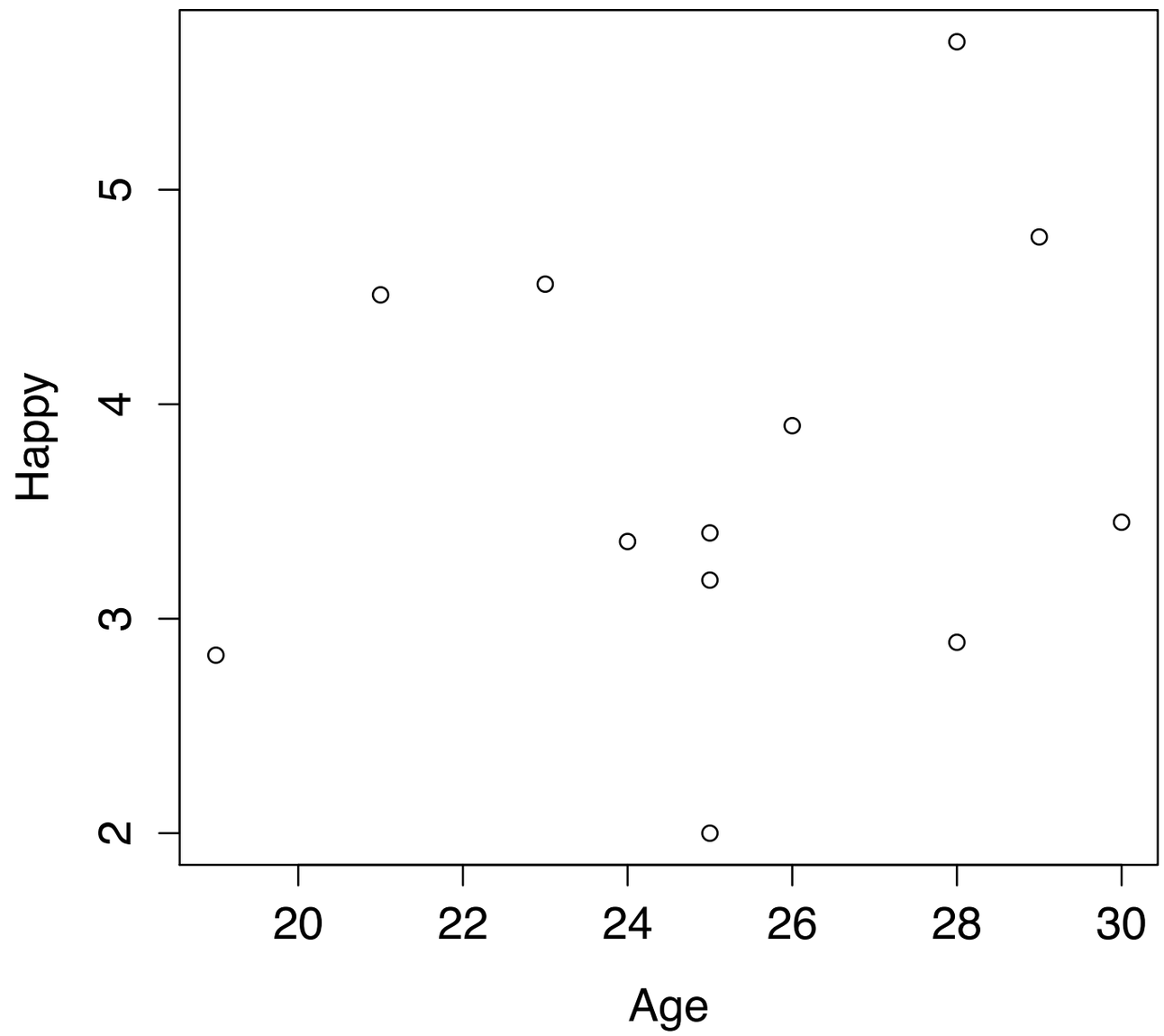
Scatter plots

```
plot( x=expt$age,
      y=expt$happy )
```



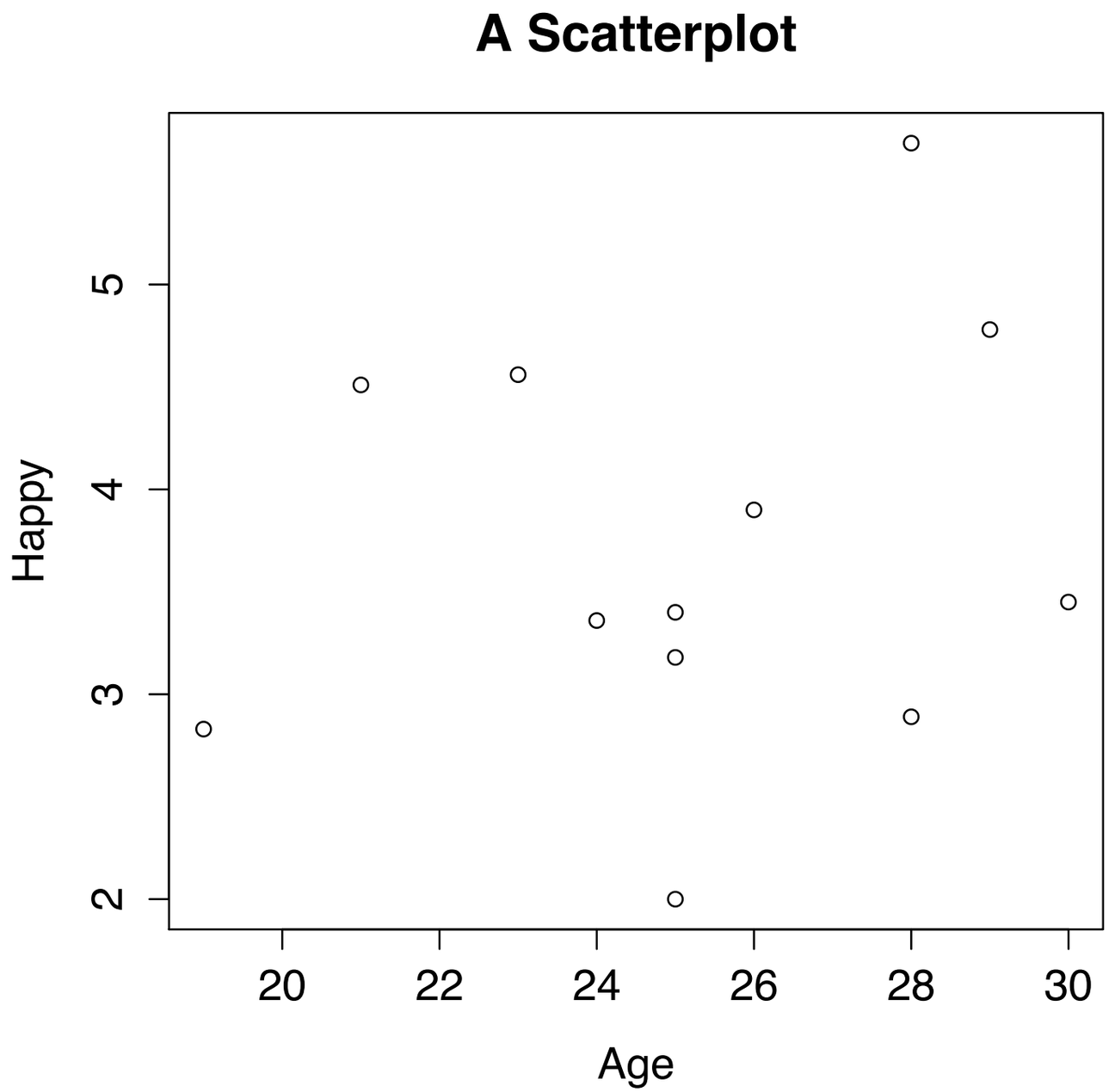
Scatter plots

```
plot( x=expt$age,
      y=expt$happy,
      xlab="Age",
      ylab="Happy" )
```



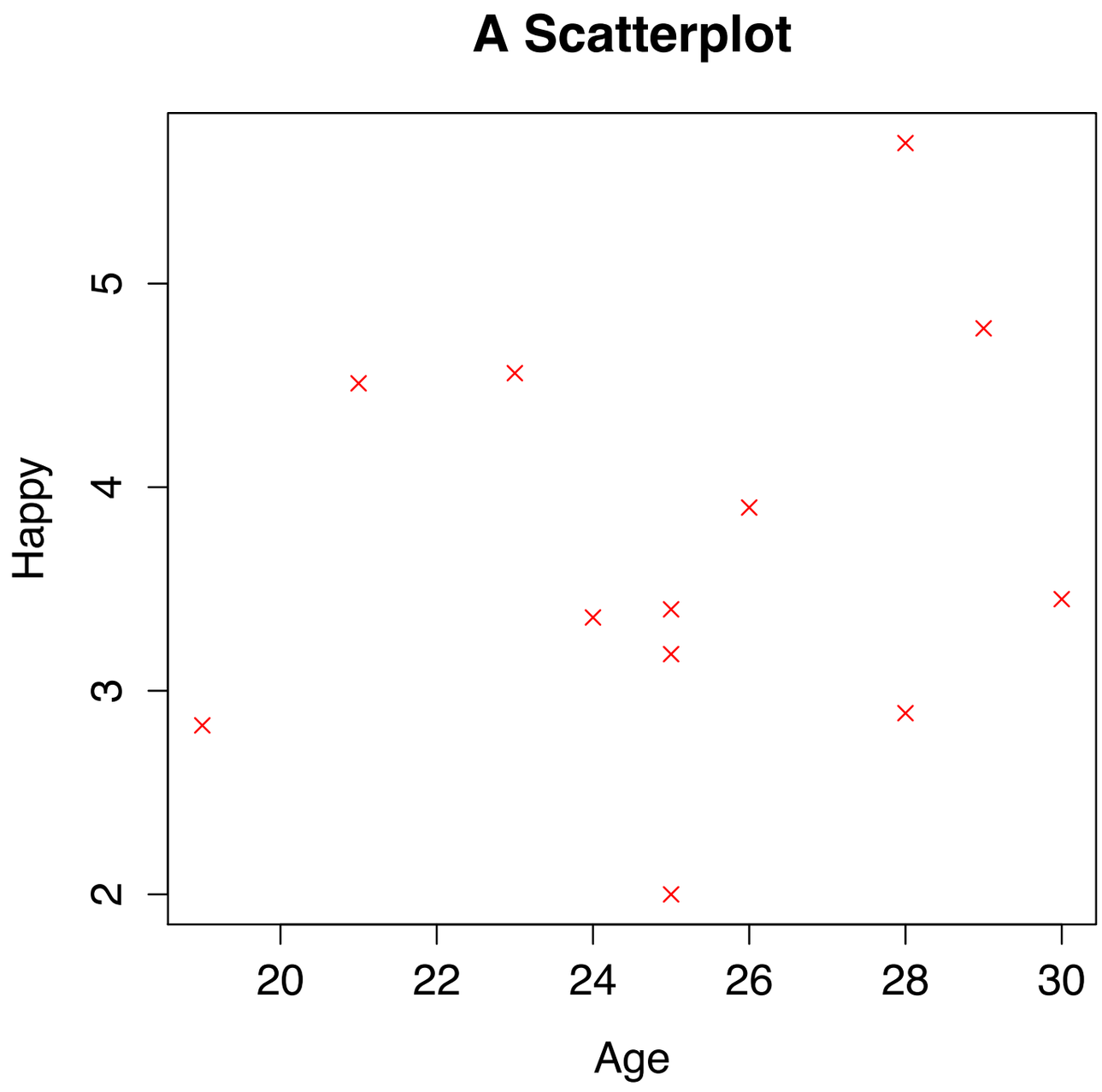
Scatter plots

```
plot( x=expt$age,
      y=expt$happy,
      xlab="Age",
      ylab="Happy",
      main="A Scatterplot"
 )
```

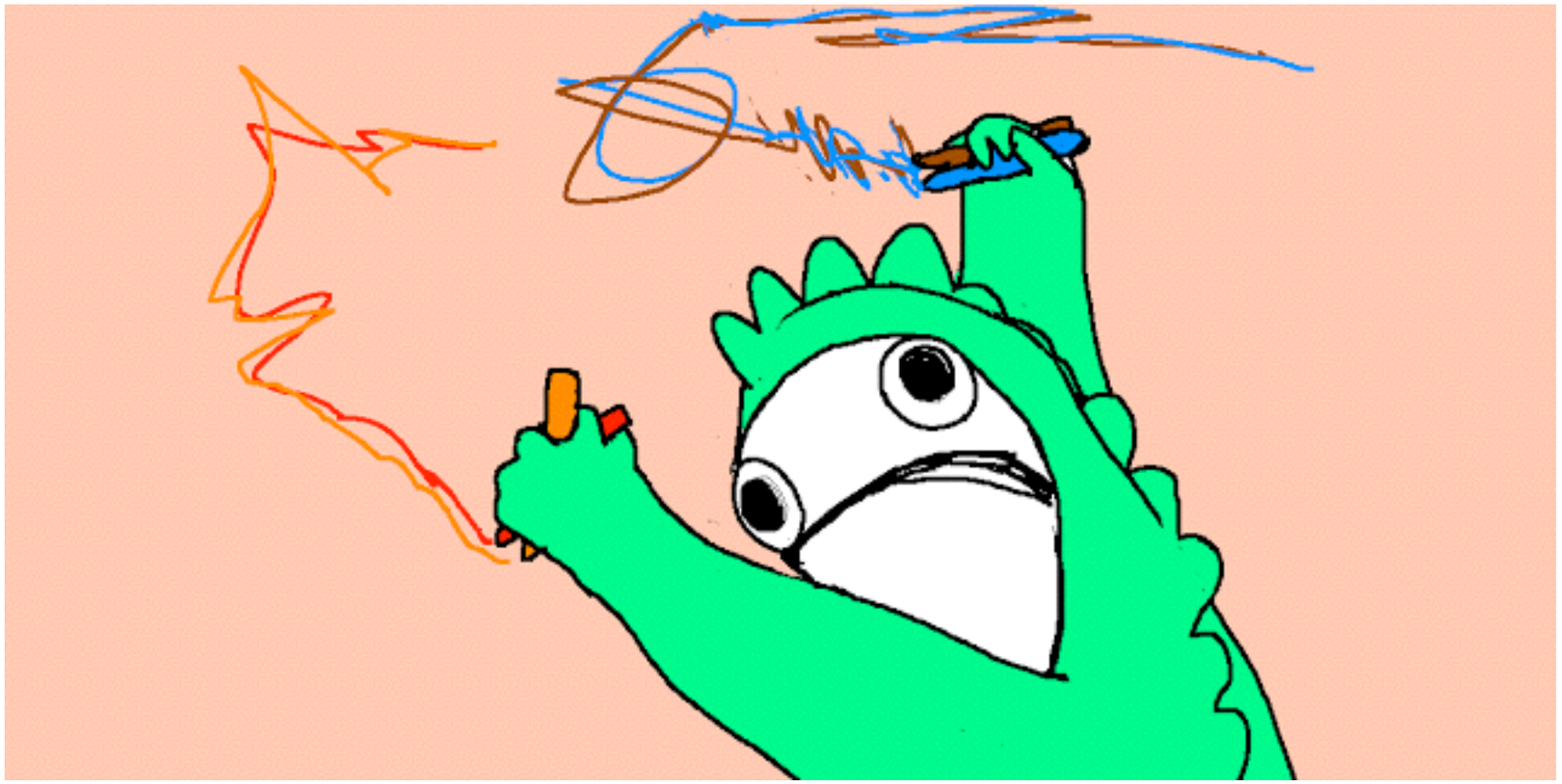


Scatter plots

```
plot( x=expt$age,
      y=expt$happy,
      xlab="Age",
      ylab="Happy",
      main="A Scatterplot",
      pch=4,
      col="red" )
```

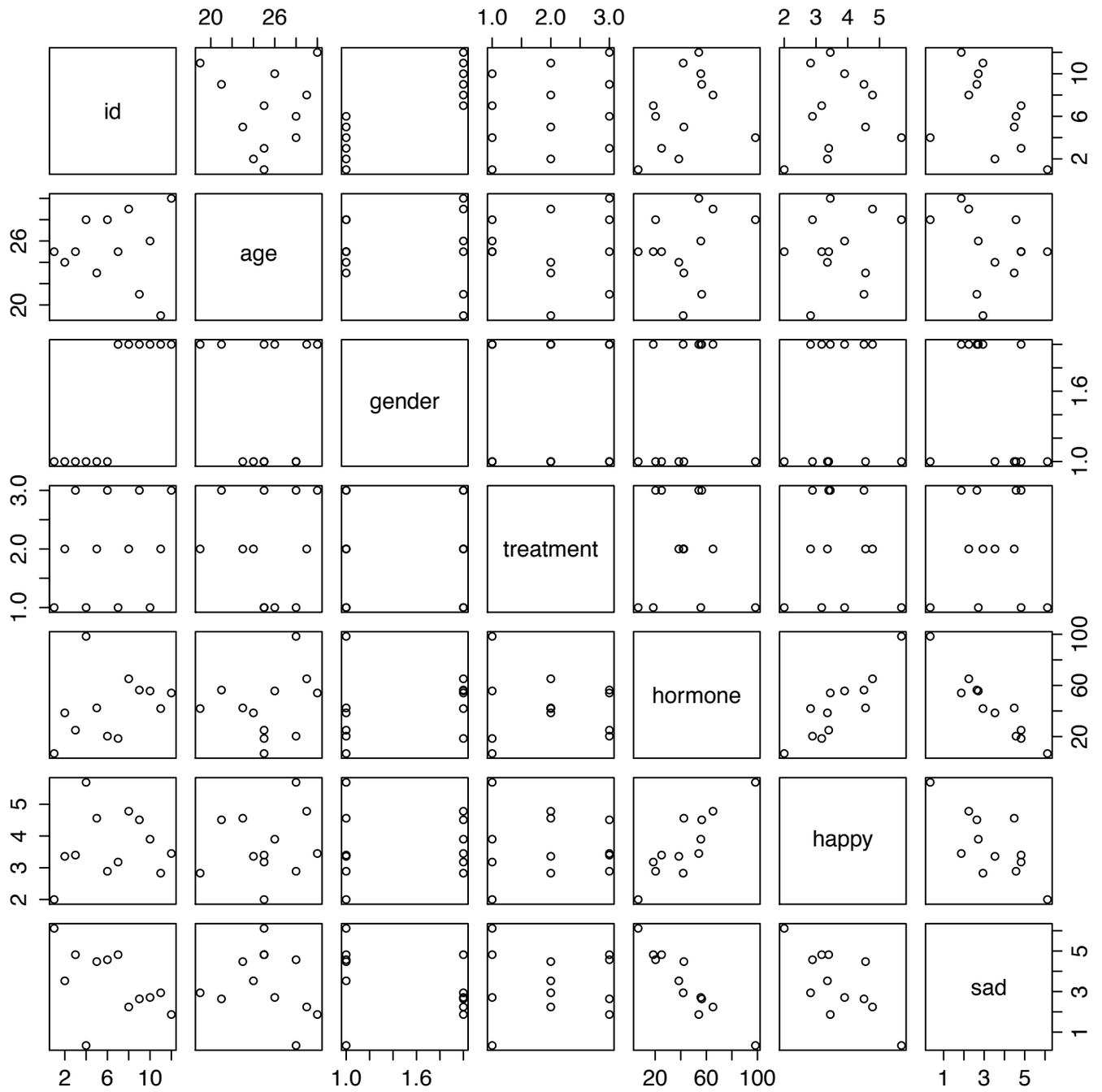


Try it yourself (Exercise 2.2.1)



Scatter plot matrix:

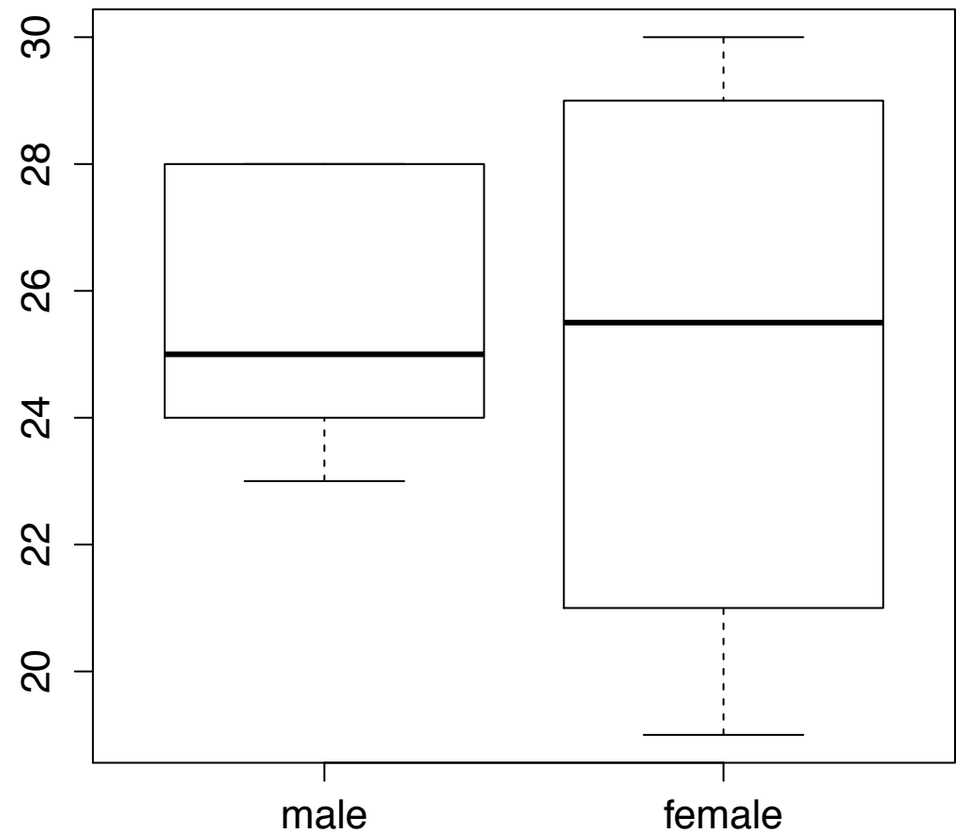
```
pairs( expt )
```



Box plots

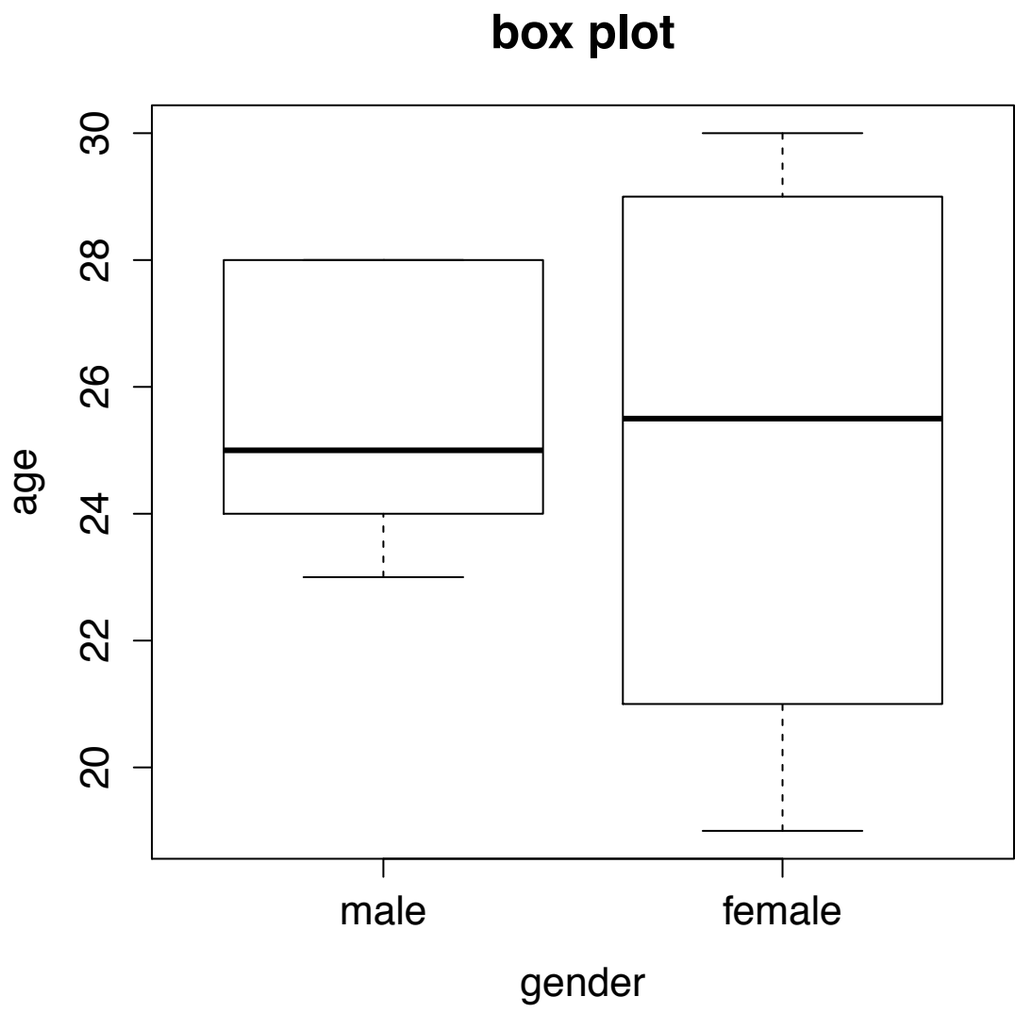
Box plots

```
boxplot(
  formula = age ~ gender,
  data = expt
)
```

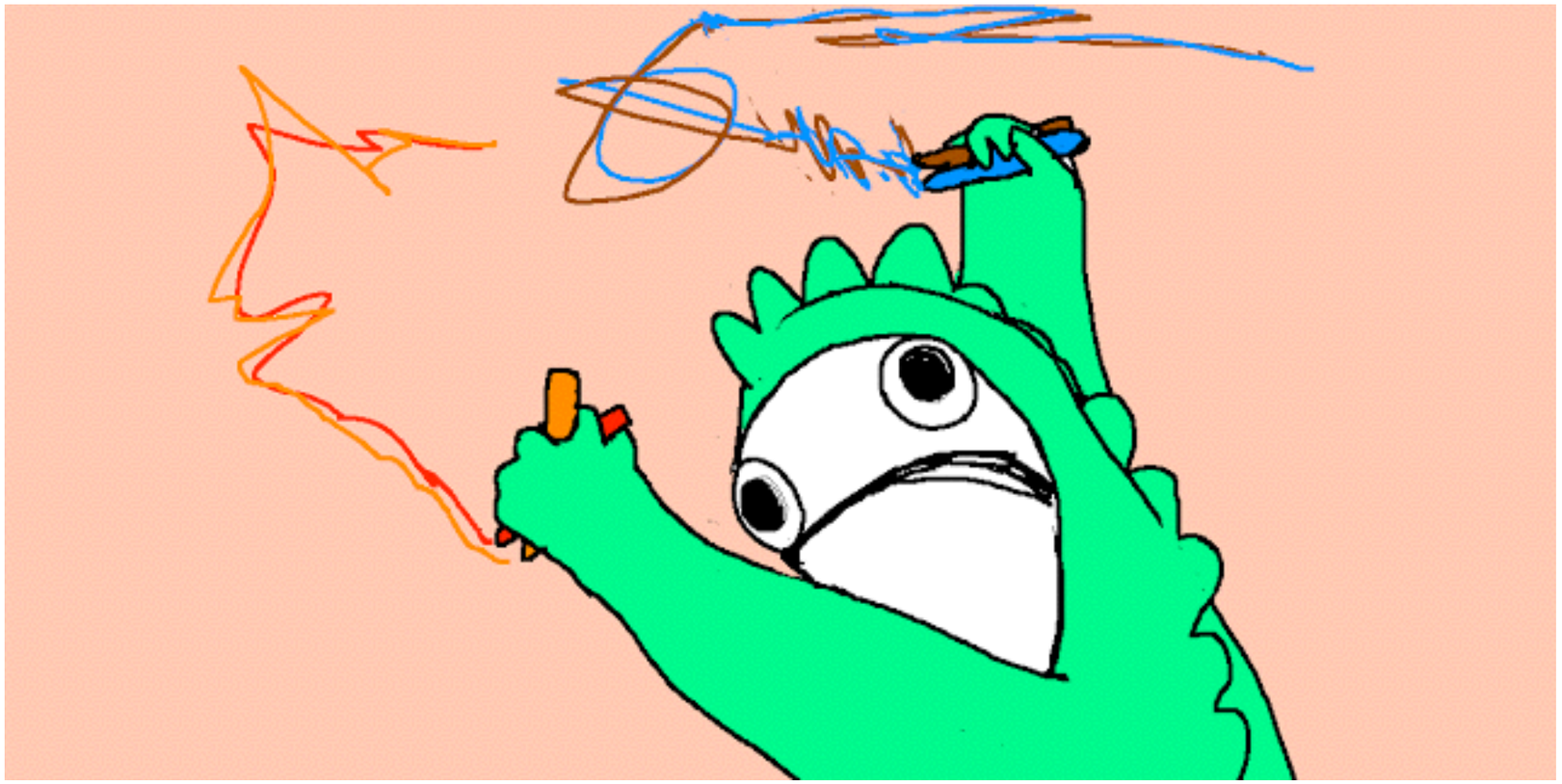


Box plots

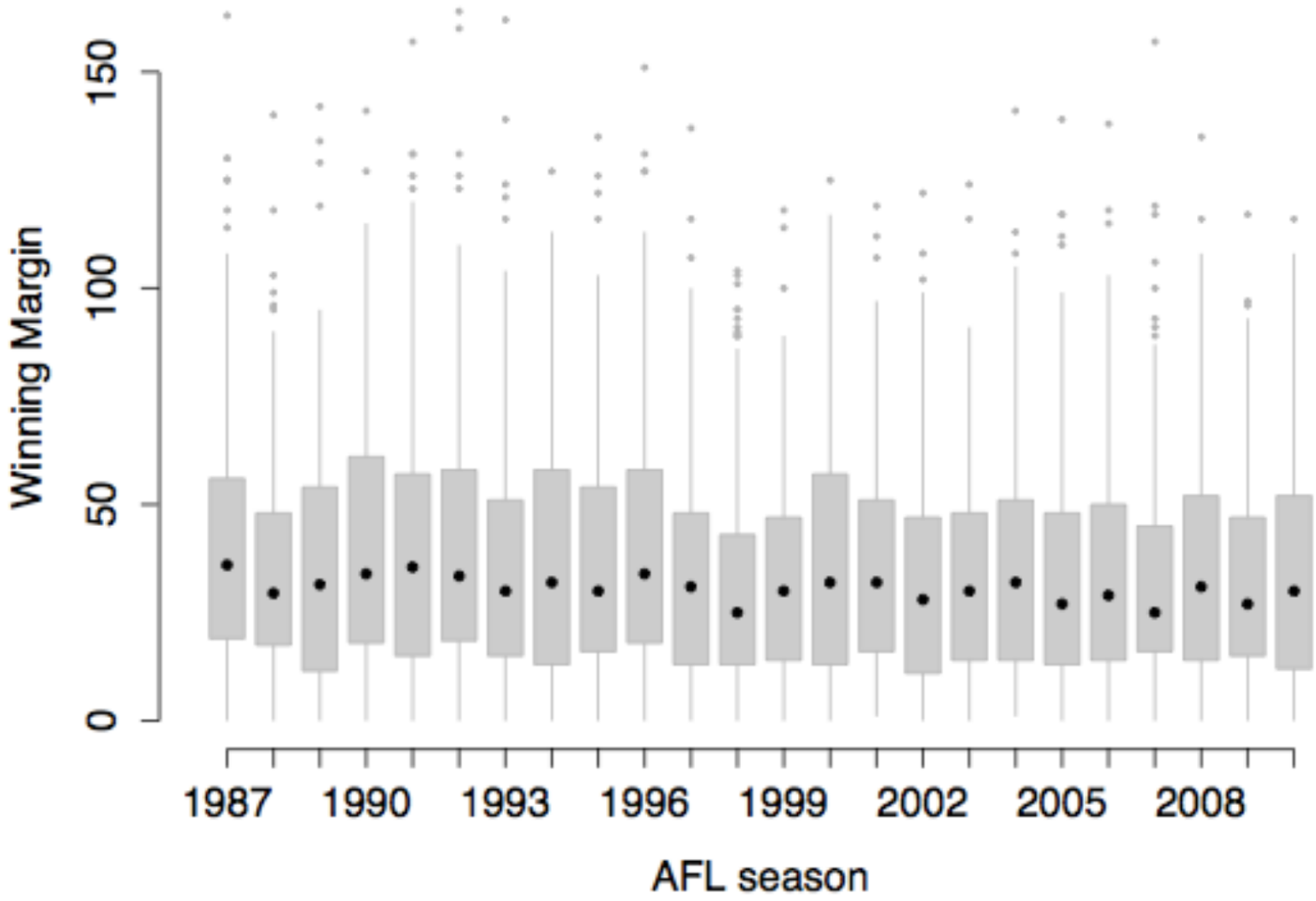
```
boxplot(
  formula = age ~ gender,
  data = expt,
  xlab = "gender",
  ylab = "age",
  main = "box plot"
)
```



Try it yourself (Exercise 2.2.2)



Very customisable!



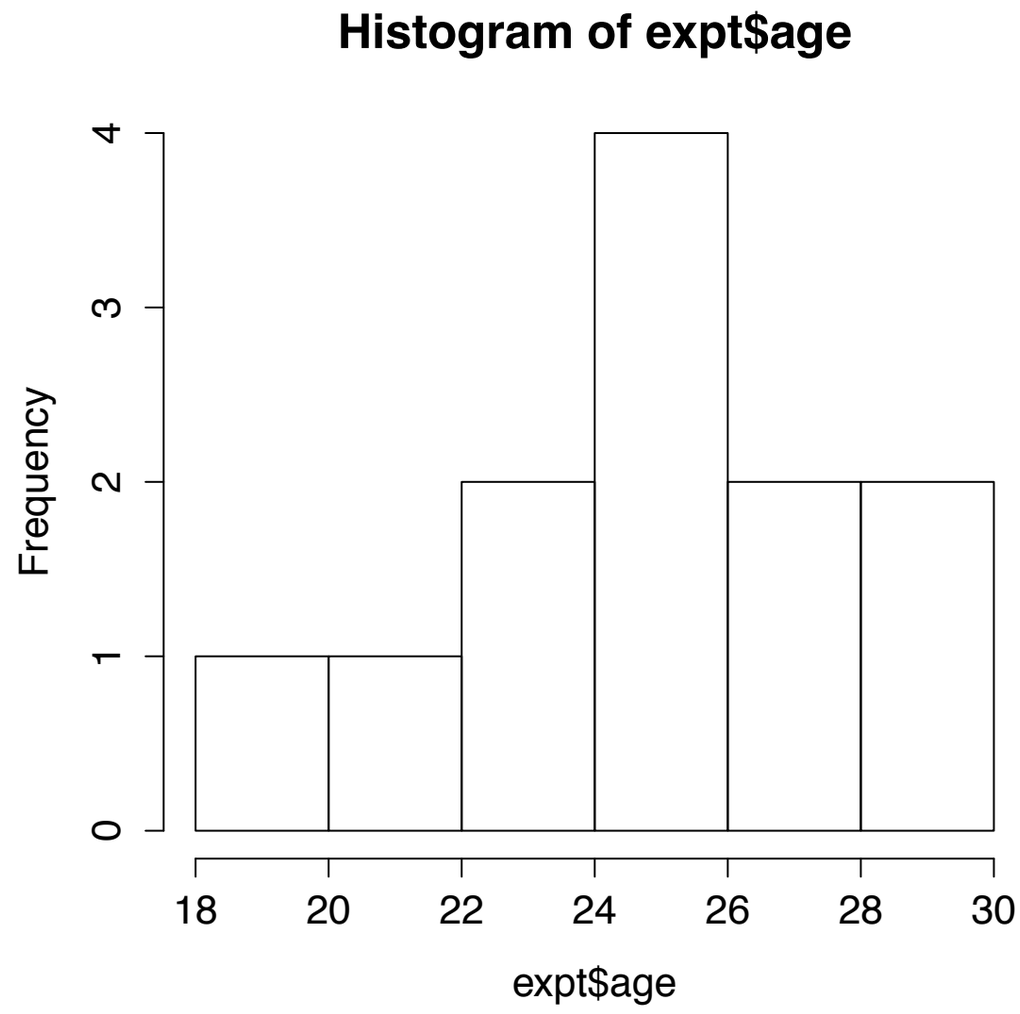
But it does take a while to learn all the options...

```
boxplot(

  formula =  margin ~ year,    # the formula
  data = afl2,                 # the data set
  xlab = "AFL season",         # x axis label
  ylab = "Winning Margin",     # y axis label
  frame.plot = FALSE,          # don't draw a frame
  staplewex = 0,               # don't draw staples
  staplecol = "white",         # (fixes a tiny display issue)
  boxwex = .75,                # narrow the boxes slightly
  boxfill = "grey80",          # lightly shade the boxes
  whisklty = 1,                # solid line for whiskers
  whiskcol = "grey70",         # dim the whiskers
  boxcol = "grey70",           # dim the box borders
  outcol = "grey70",           # dim the outliers
  outpch = 20,                 # outliers as solid dots
  outcex = .5,                 # shrink the outliers
  medlty = "blank",            # no line for the medians
  medpch = 20,                 # instead, draw solid dots
  medlwd = 1.5                 # make them larger

)
```

Histograms

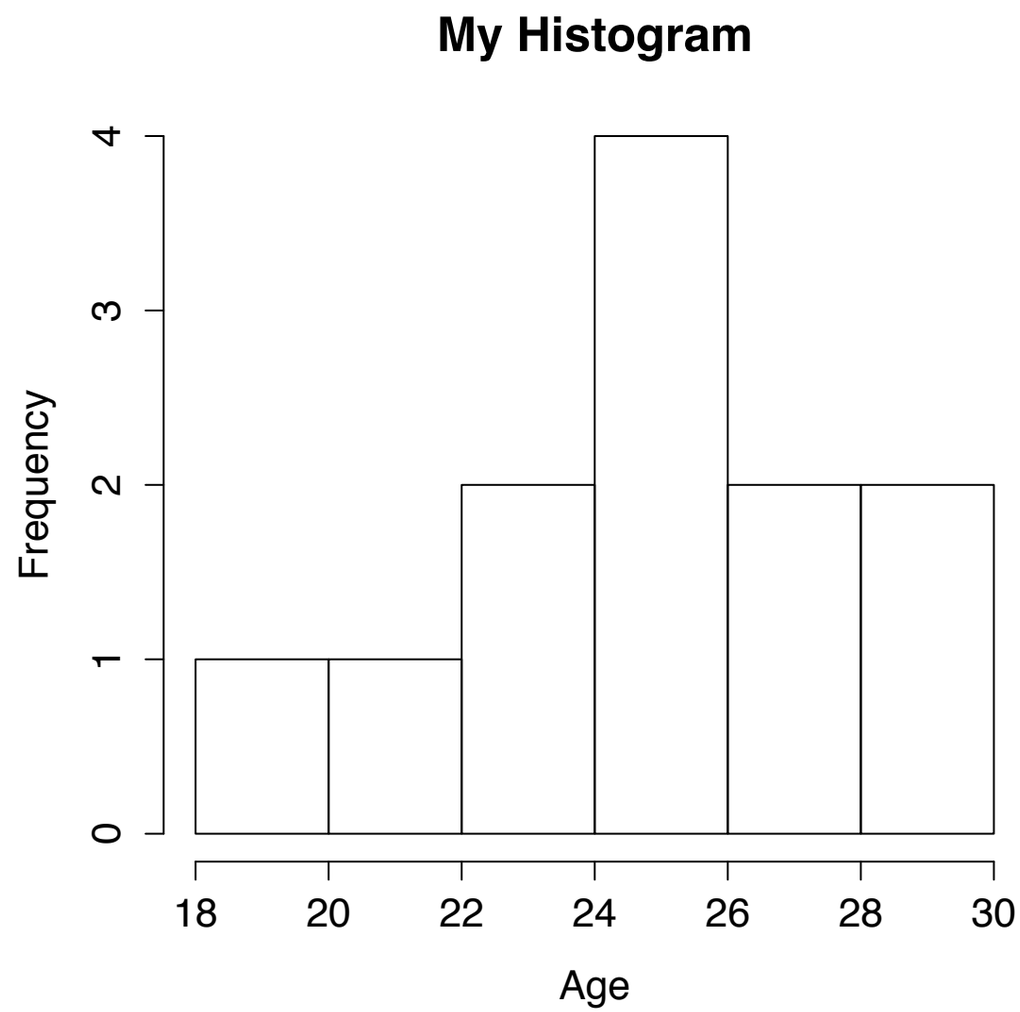
Histograms

```
hist( expt$age )
```



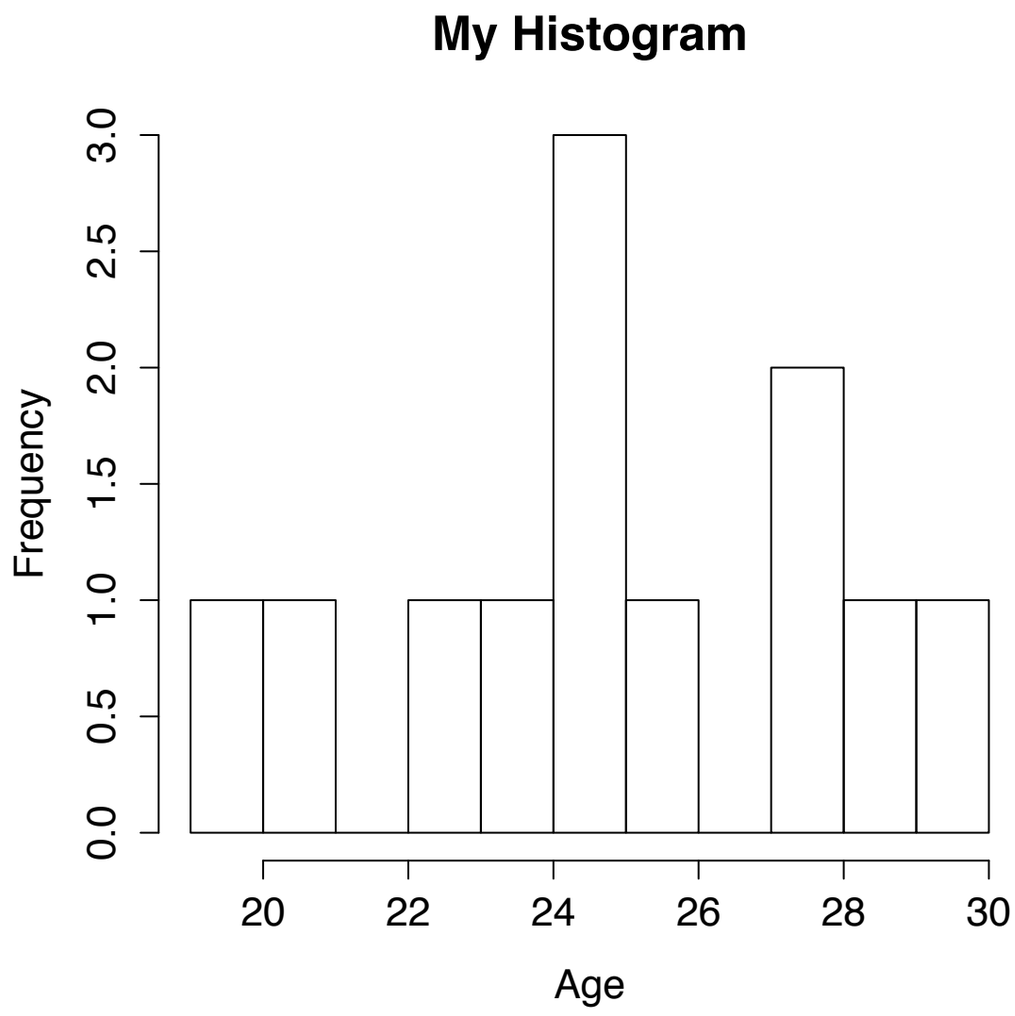
Histograms

```
hist(
   x = expt$age,
   xlab = "Age",
   main = "My Histogram"
)
```



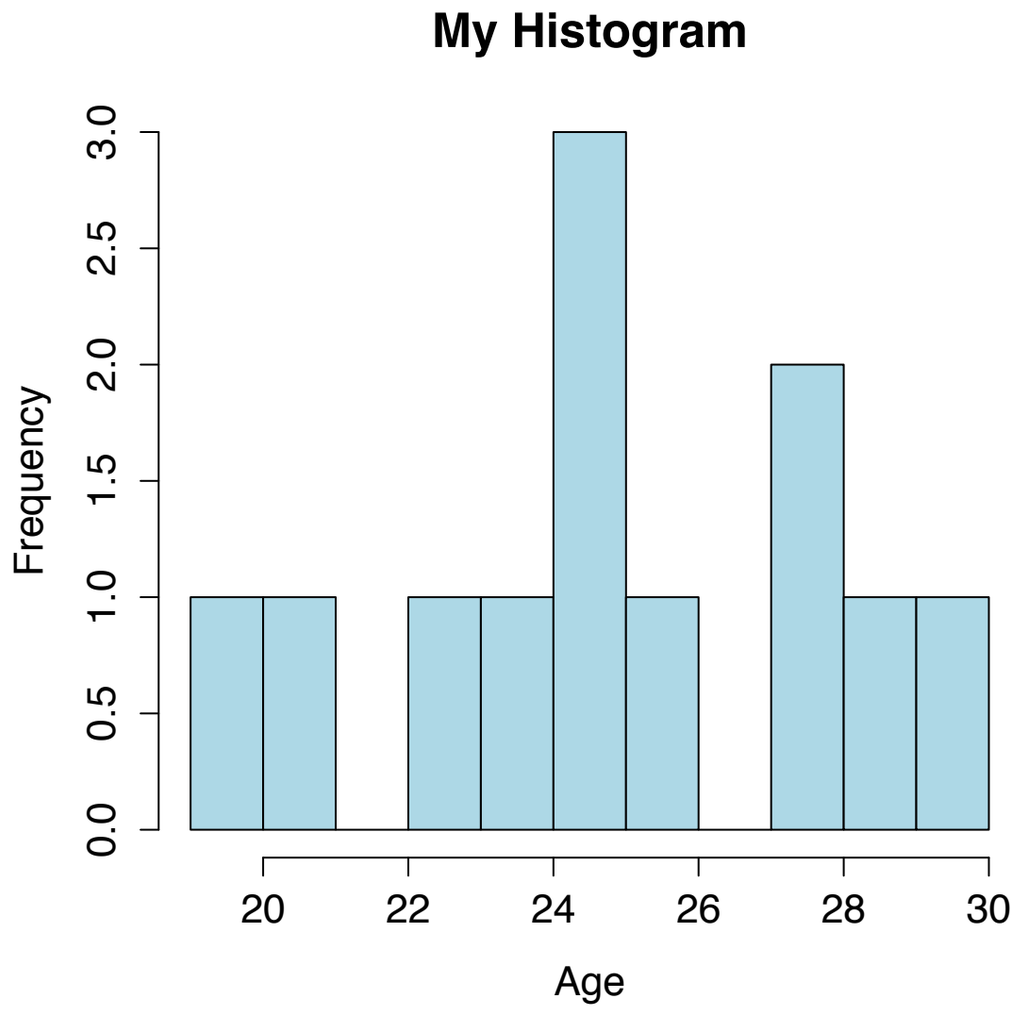
Histograms

```
hist(
   x = expt$age,
   xlab = "Age",
   main = "My Histogram",
   breaks = 10
)
```

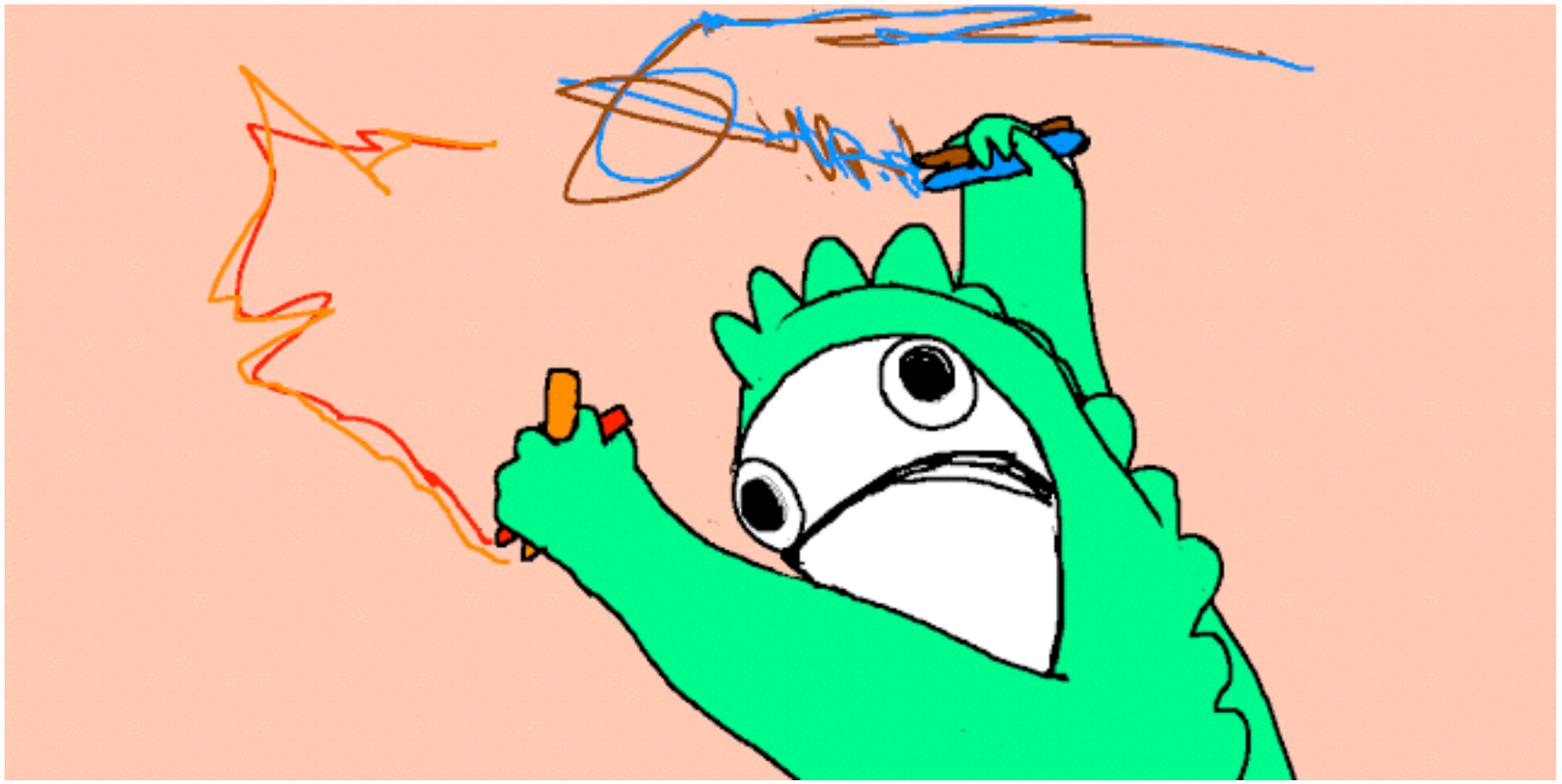


Histograms

```
hist(
  x = expt$age,
  xlab = "Age",
  main = "My Histogram",
  breaks = 10,
  col = "light blue"
)
```



Try it yourself (Exercise 2.2.3)



Digression:
How does R define colours?

R knows lots of "human" colour names

```
> colours()
  [1] "white"           "aliceblue"           "antiquewhite"
  [4] "antiquewhite1"   "antiquewhite2"       "antiquewhite3"
  [7] "antiquewhite4"   "aquamarine"          "aquamarine1"
 [10] "aquamarine2"     "aquamarine3"         "aquamarine4"
 [13] "azure"           "azure1"              "azure2"
 [16] "azure3"          "azure4"              "beige"
 [19] "bisque"          "bisque1"             "bisque2"
...
[637] "turquoise2"     "turquoise3"          "turquoise4"
[640] "violet"         "violetred"           "violetred1"
[643] "violetred2"     "violetred3"          "violetred4"
[646] "wheat"          "wheat1"              "wheat2"
[649] "wheat3"         "wheat4"              "whitesmoke"
[652] "yellow"         "yellow1"             "yellow2"
[655] "yellow3"        "yellow4"             "yellowgreen"
```

But it also does colours like a machine

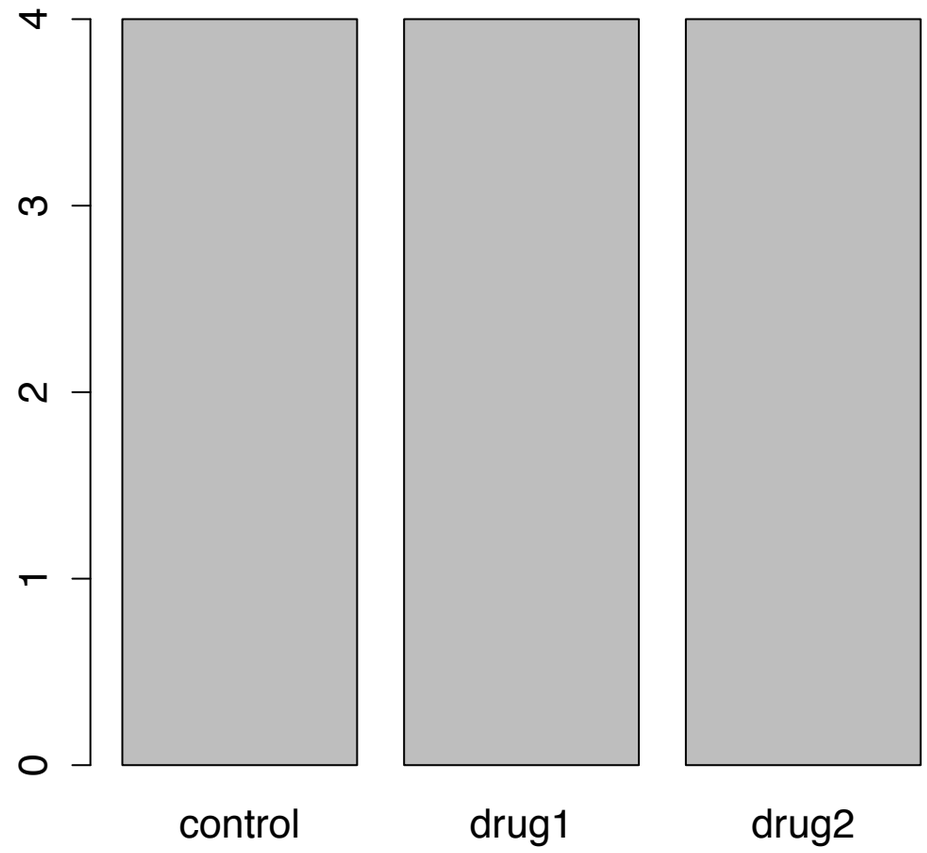
- Lots of options...
  - You can specify RGB values, using rgb()
  - You can specify HSV values, using hsv()
  - You can create themed palettes using commands like rainbow() or heat.colors()
  - You can also manually specify an HTML-like hexadecimal code

- But I find I don't need them often:
  - R understands 657 "natural language" colours... which is more than me!

Bar plots of frequency tables

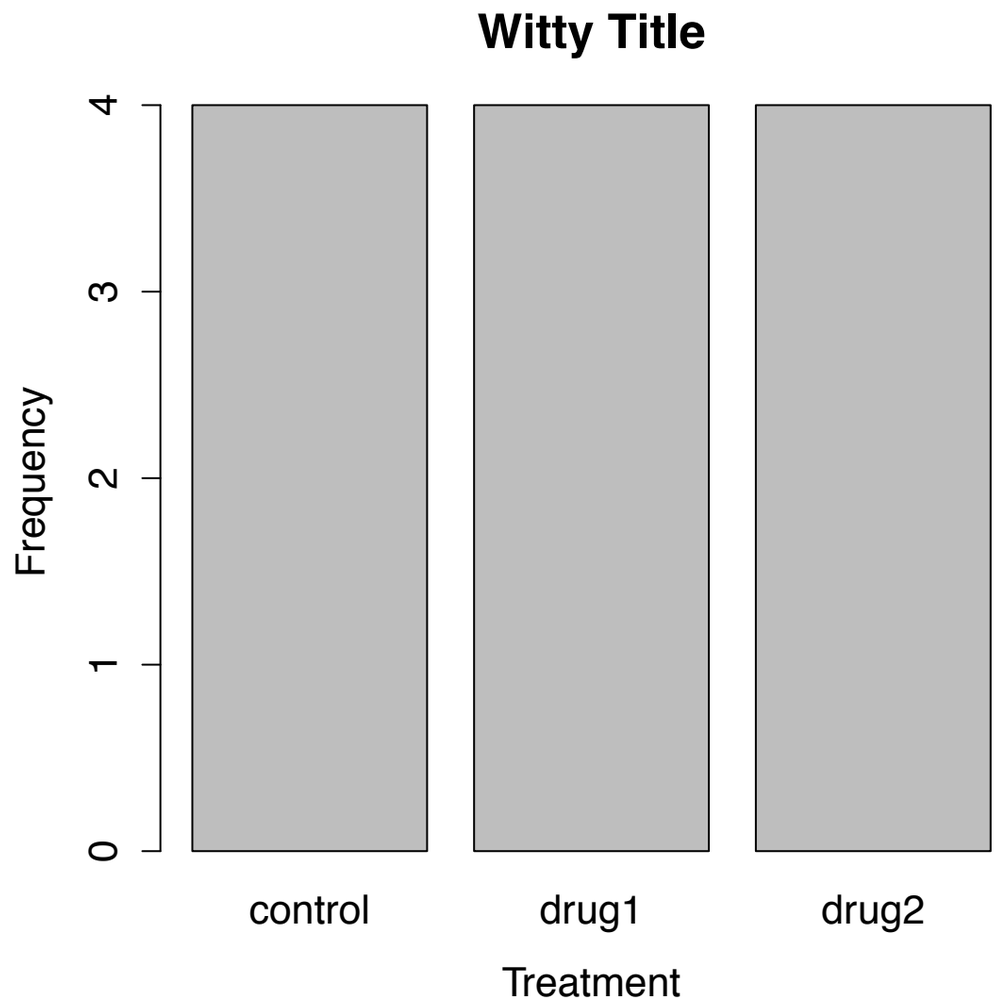
Bar plots of frequency tables

```
counts <- table( expt$treatment )
barplot( counts )
```

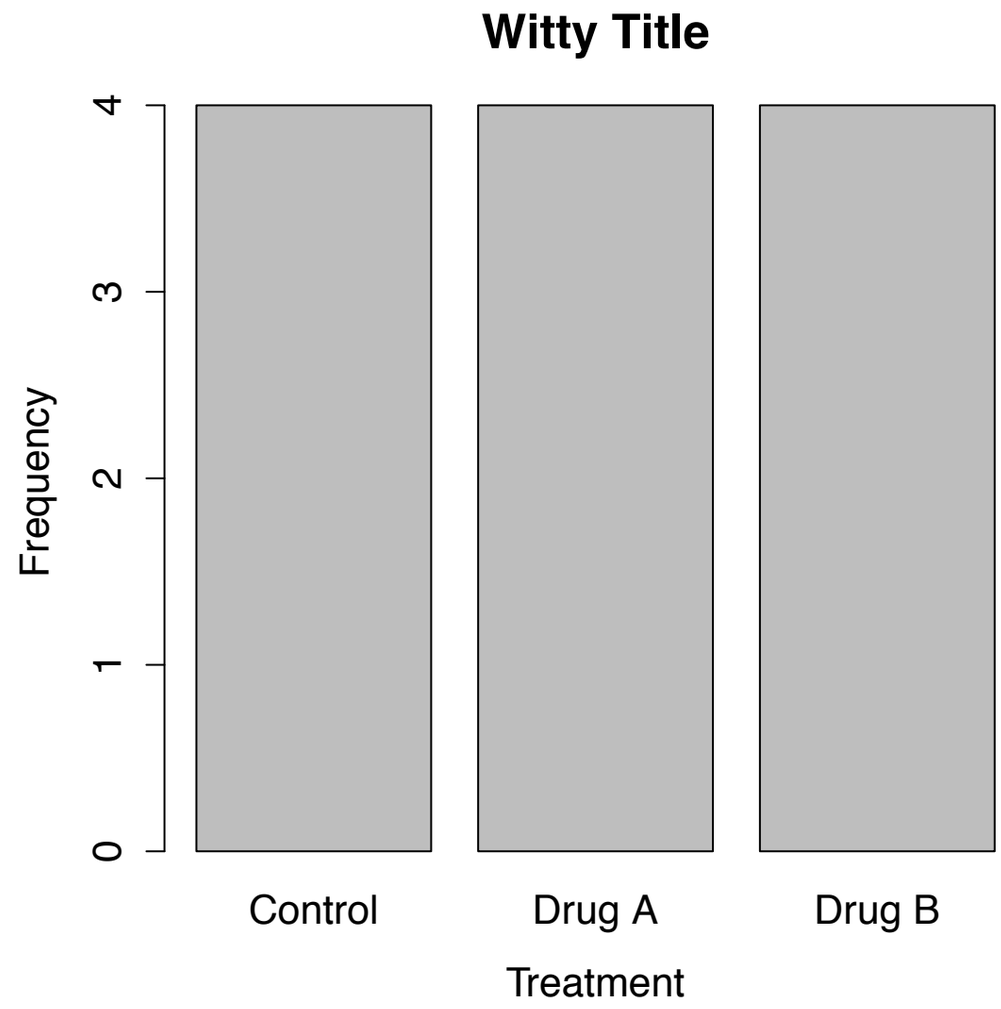


Bar plots of frequency tables

```
barplot(
  height = counts,
  xlab = "Treatment",
  ylab = "Frequency",
  main = "Witty Title"
)
```

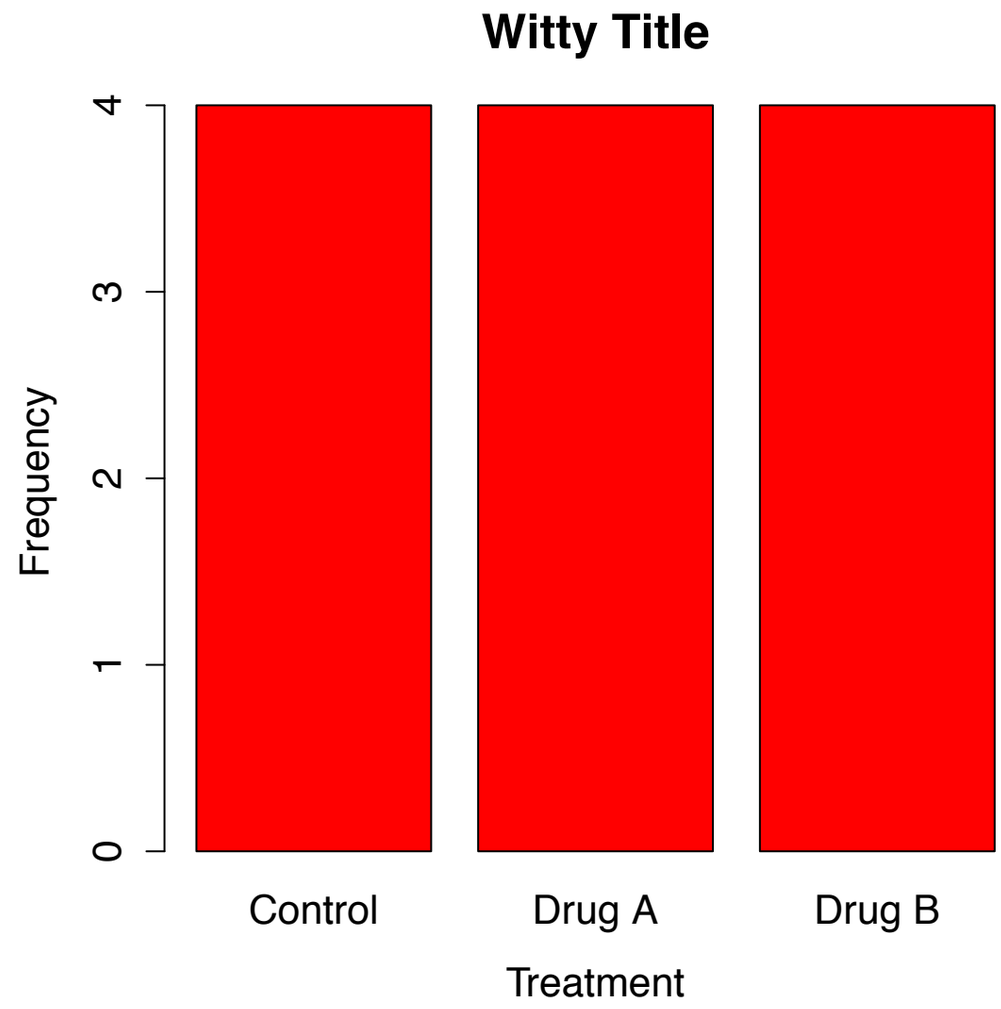


Bar plots of frequency tables



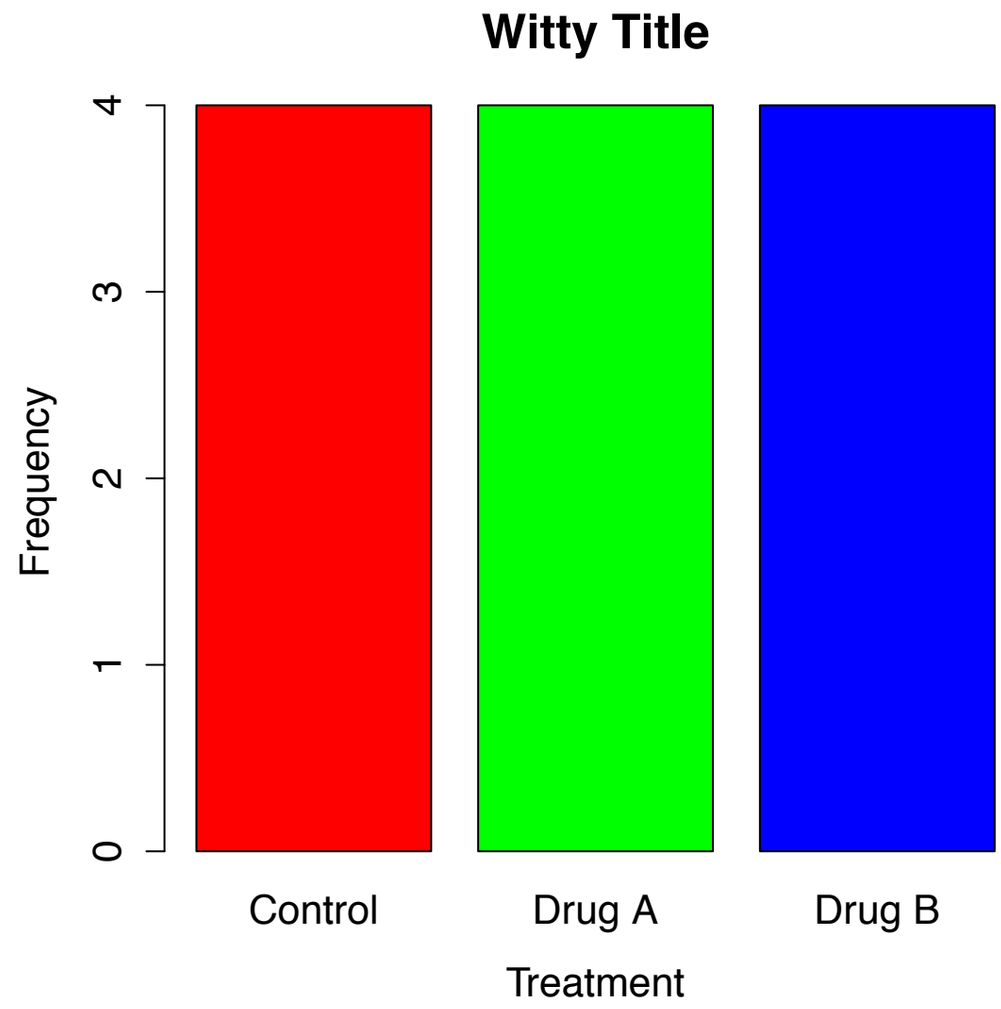

```
barplot(
  height = counts,
  xlab = "Treatment",
  ylab = "Frequency",
  main = "Witty Title",
  names.arg = c("Control","Drug A","Drug B")
)
```

Bar plots of frequency tables



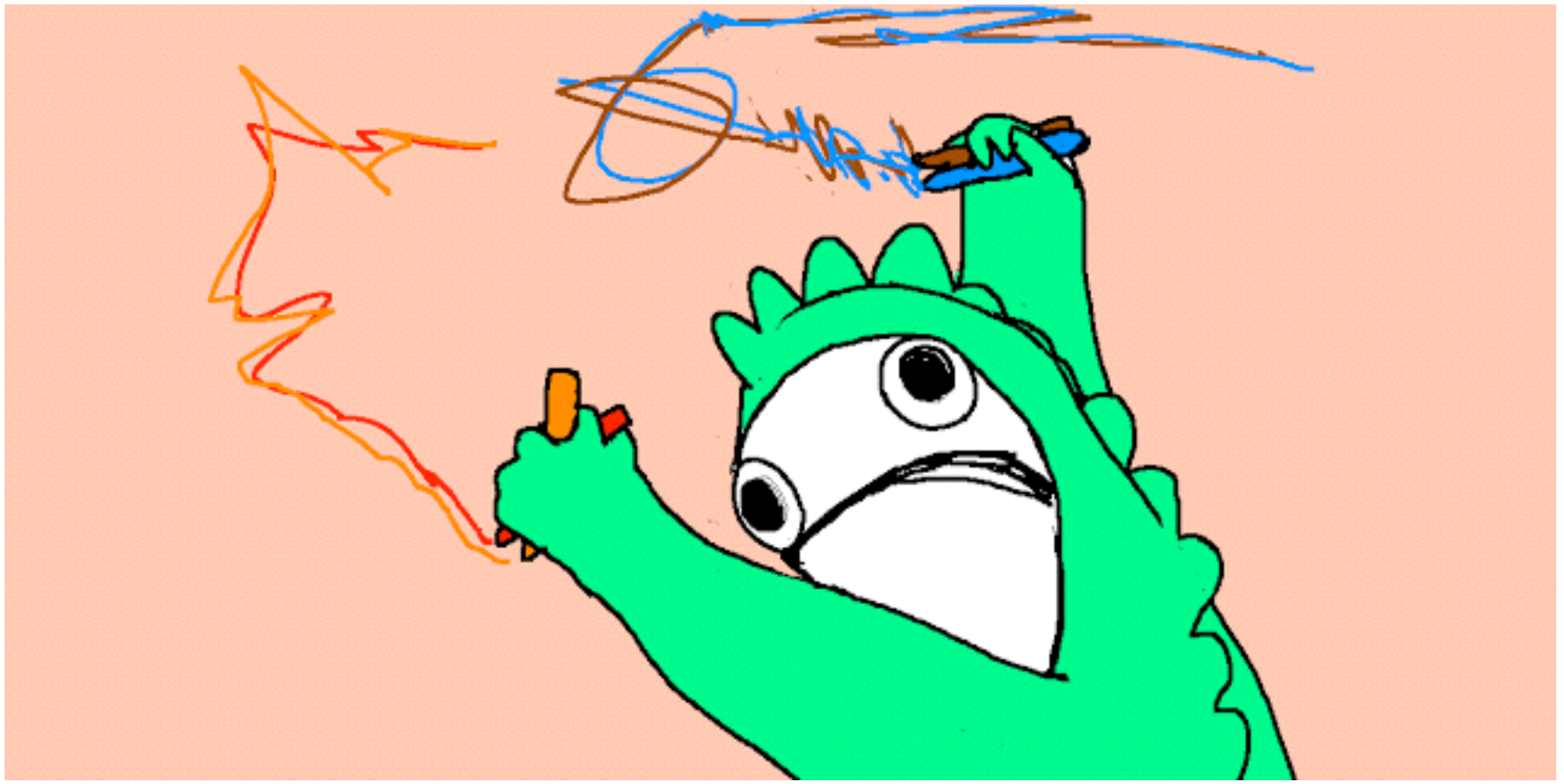

```
barplot(
  height = counts,
  xlab = "Treatment",
  ylab = "Frequency",
  main = "Witty Title",
  names.arg = c("Control","Drug A","Drug B"),
  col = "red"
)
```

Bar plots of frequency tables



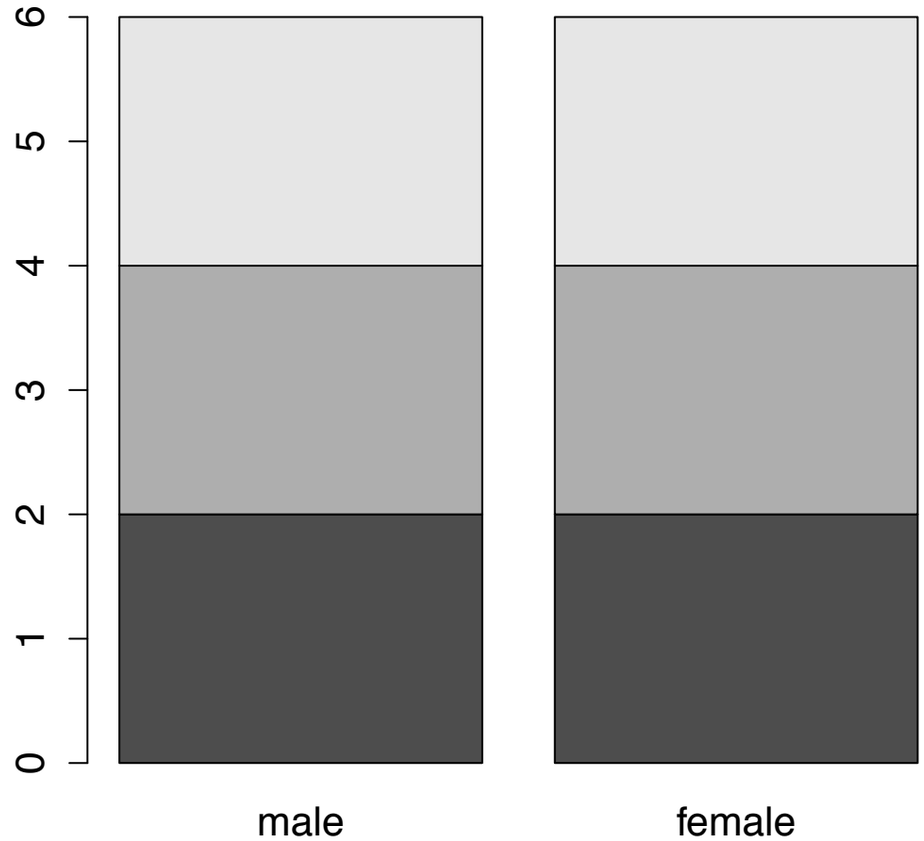

```
barplot(
  height = counts,
  xlab = "Treatment",
  ylab = "Frequency",
  main = "Witty Title",
  names.arg = c("Control","Drug A","Drug B"),
  col = c("red", "green", "blue")
)
```

Try it yourself (Exercise 2.2.4)



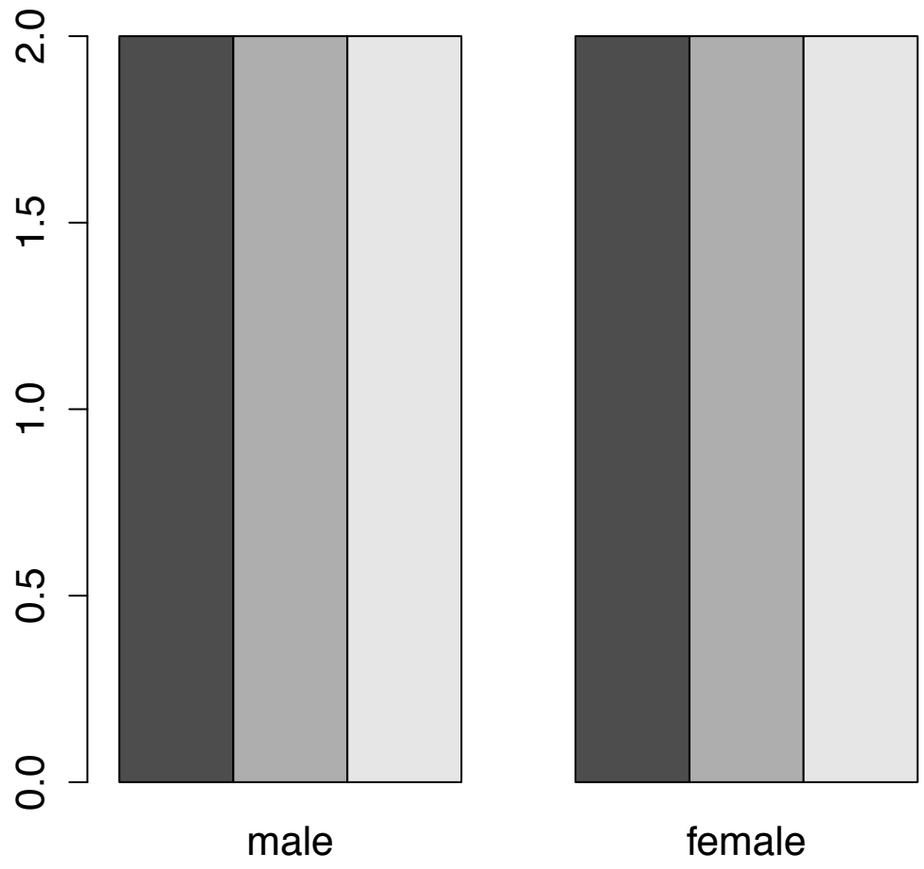
The default for a cross-tabs matrix is a stacked bar chart. Which I hate...

```
> counts <- xtabs( ~ treatment + gender, expt )
> counts
          gender
treatment male female
  control    2      2
  drug1      2      2
  drug2      2      2

> barplot( counts )
```



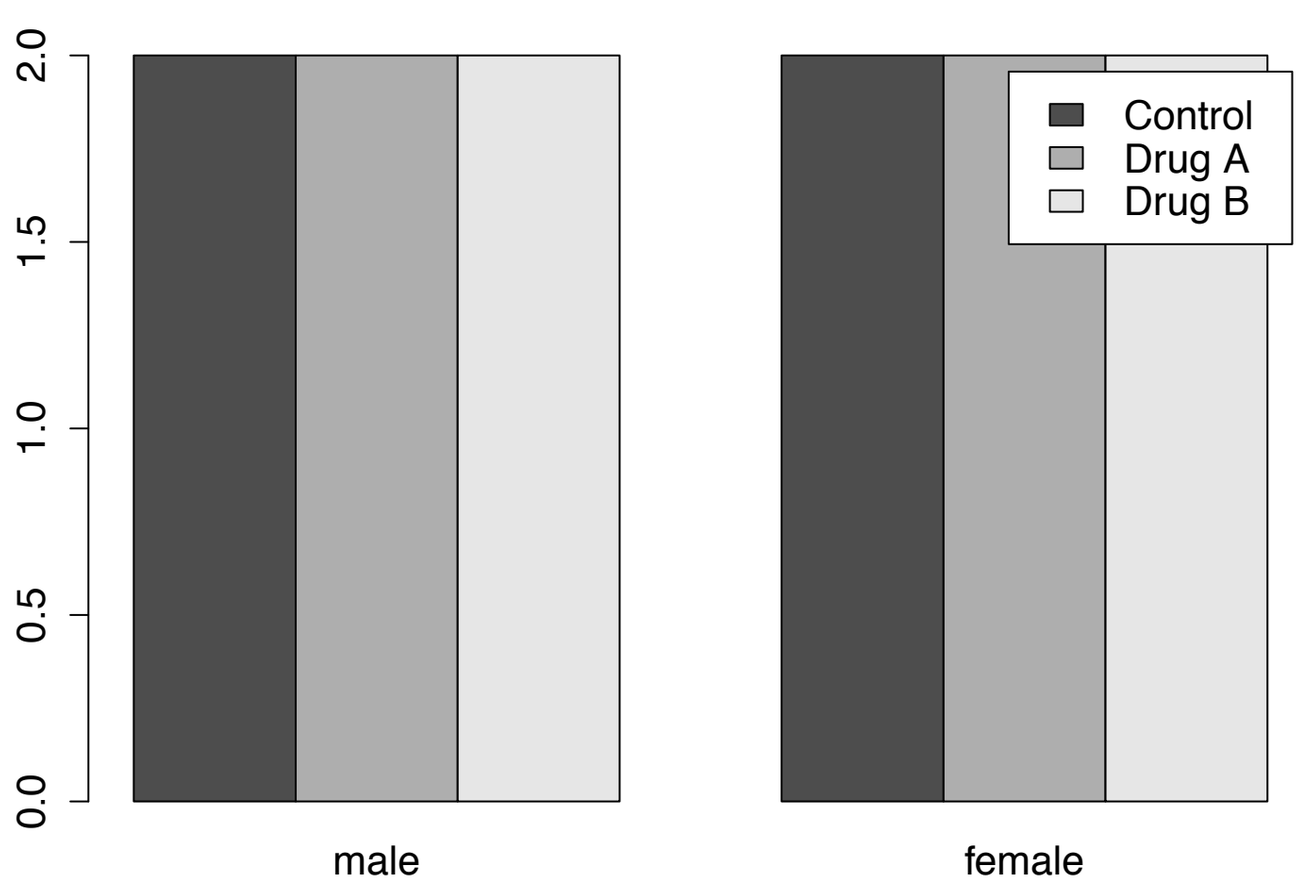
I think it's nice side by side...

```
barplot( counts, beside=TRUE )
```



It's even better with a legend!

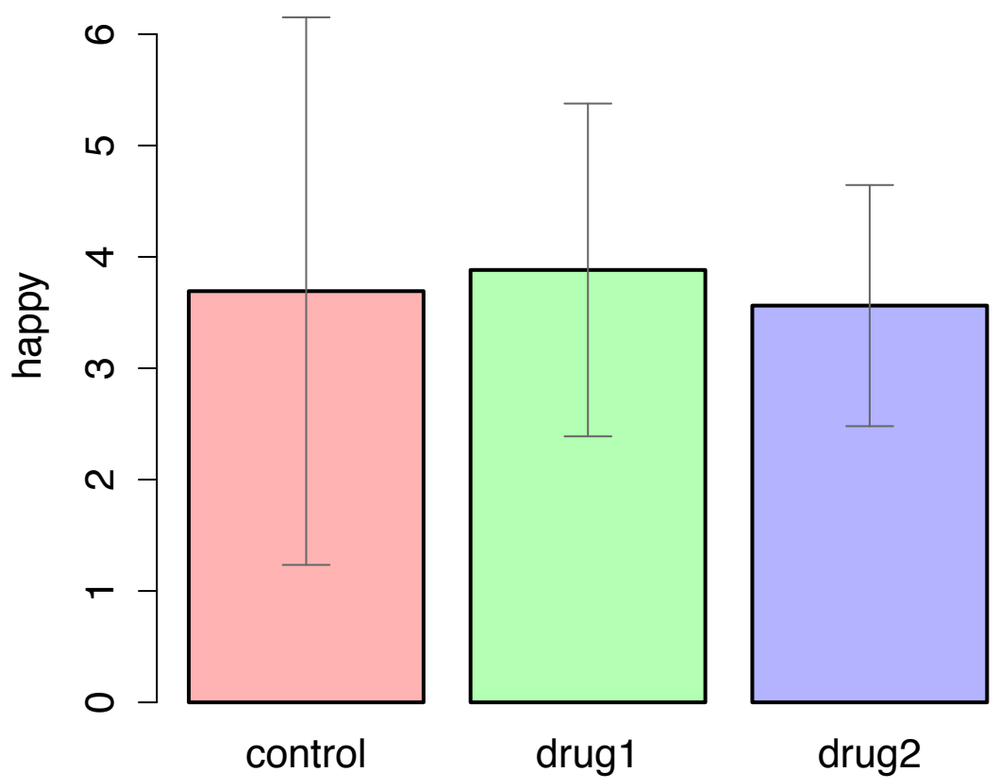
```
barplot(
  height = counts,
  beside = TRUE,
  legend.text = c("Control", "Drug A", "Drug B")
)
```



Bar plots of group means

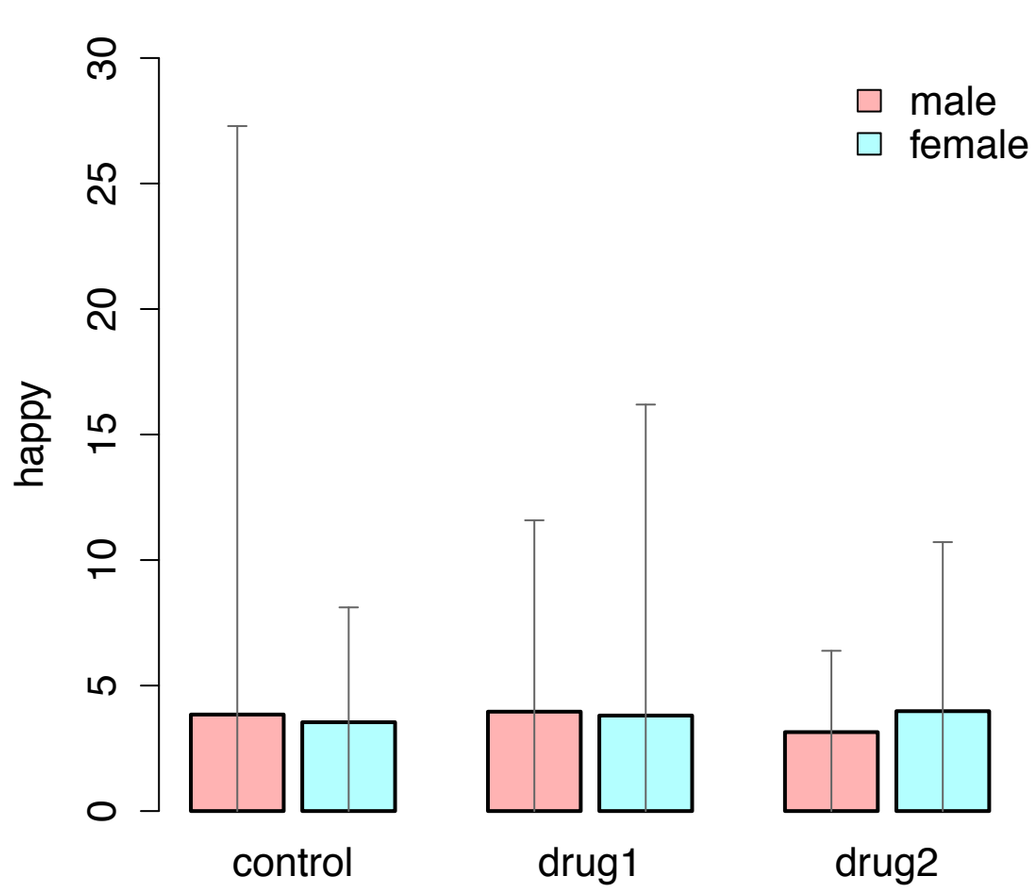
Group means with 95% CI

```
library(lsr)
bars(
  formula = happy ~ treatment,
  data = expt
)
```



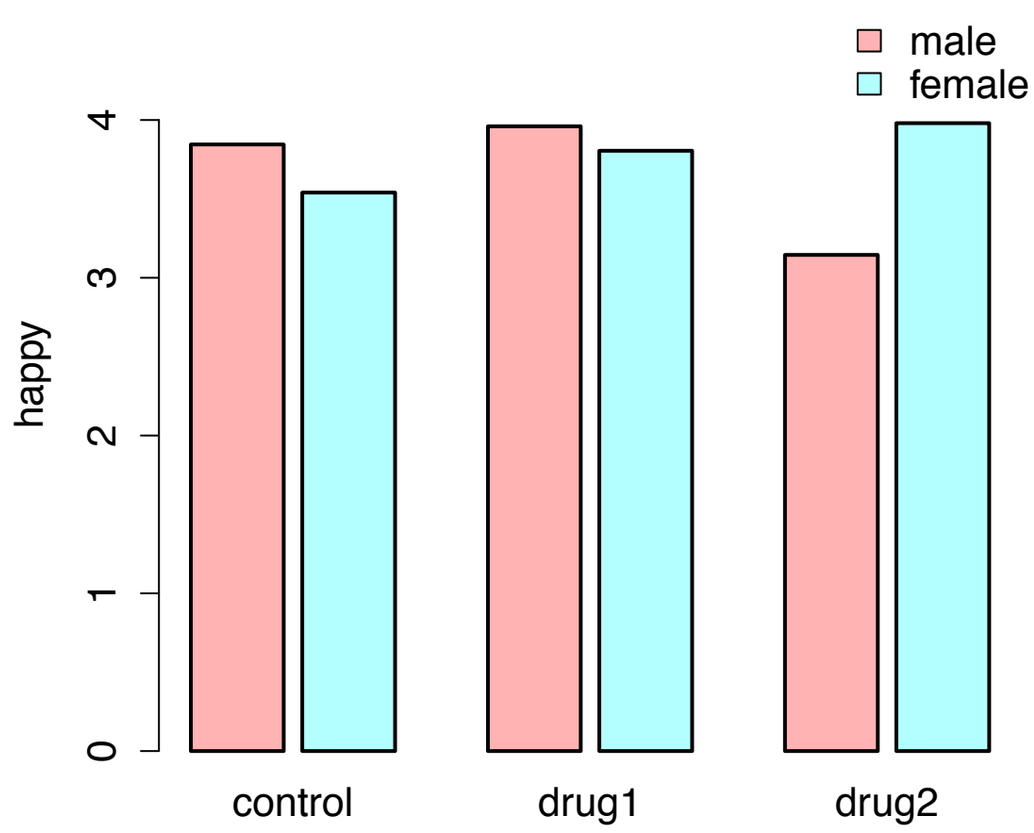
Bars allows two grouping factors

```
bars(
 formula = happy ~ treatment + gender,
 data = expt
)
```



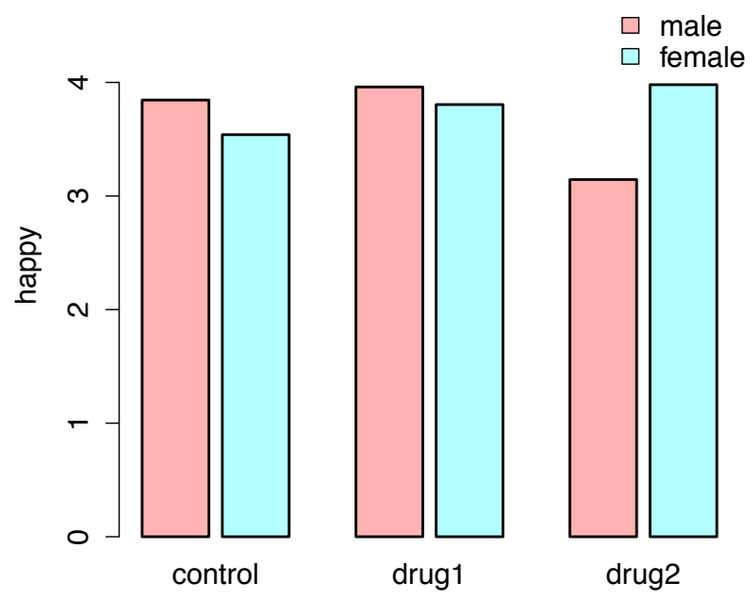
You can remove the error bars

```
bars(
 formula = happy ~ treatment + gender,
 data = expt,
 errorFun = FALSE
)
```



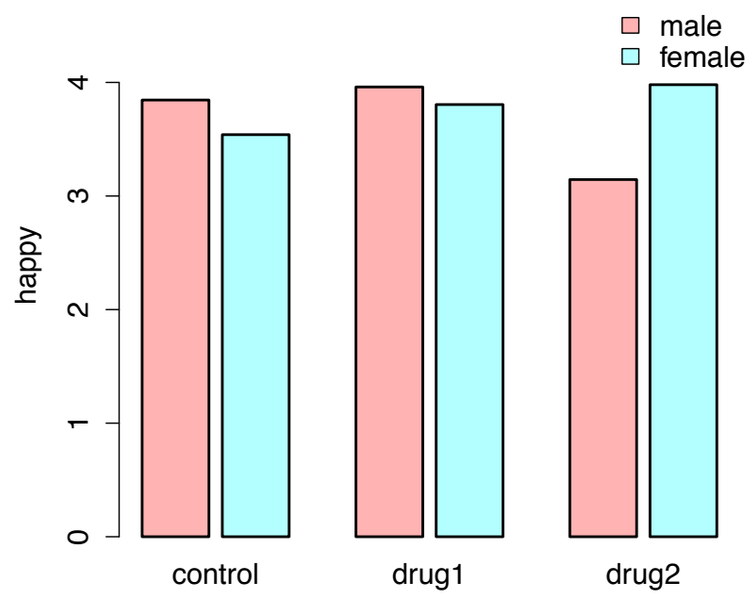
The ordering of grouping factors matters

```
bars(
 formula = happy ~ treatment + gender,
 data = expt,
 errorFun = FALSE
)
```

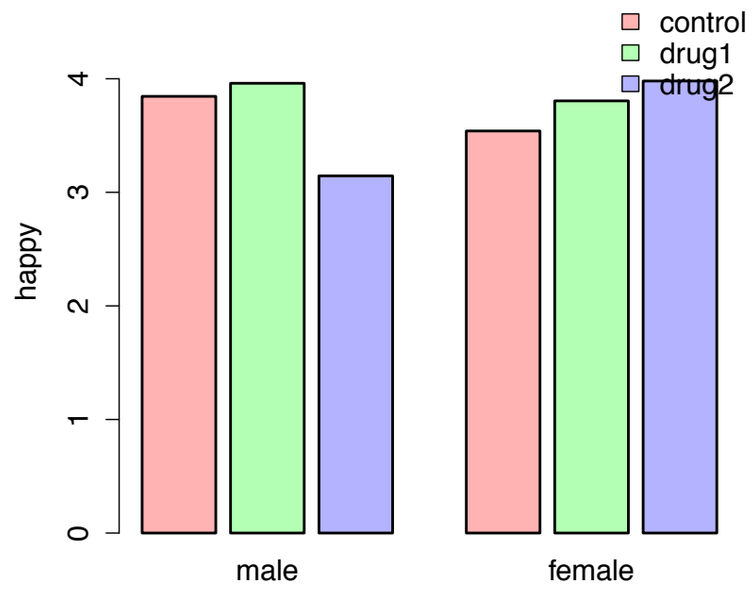


The ordering of grouping factors matters

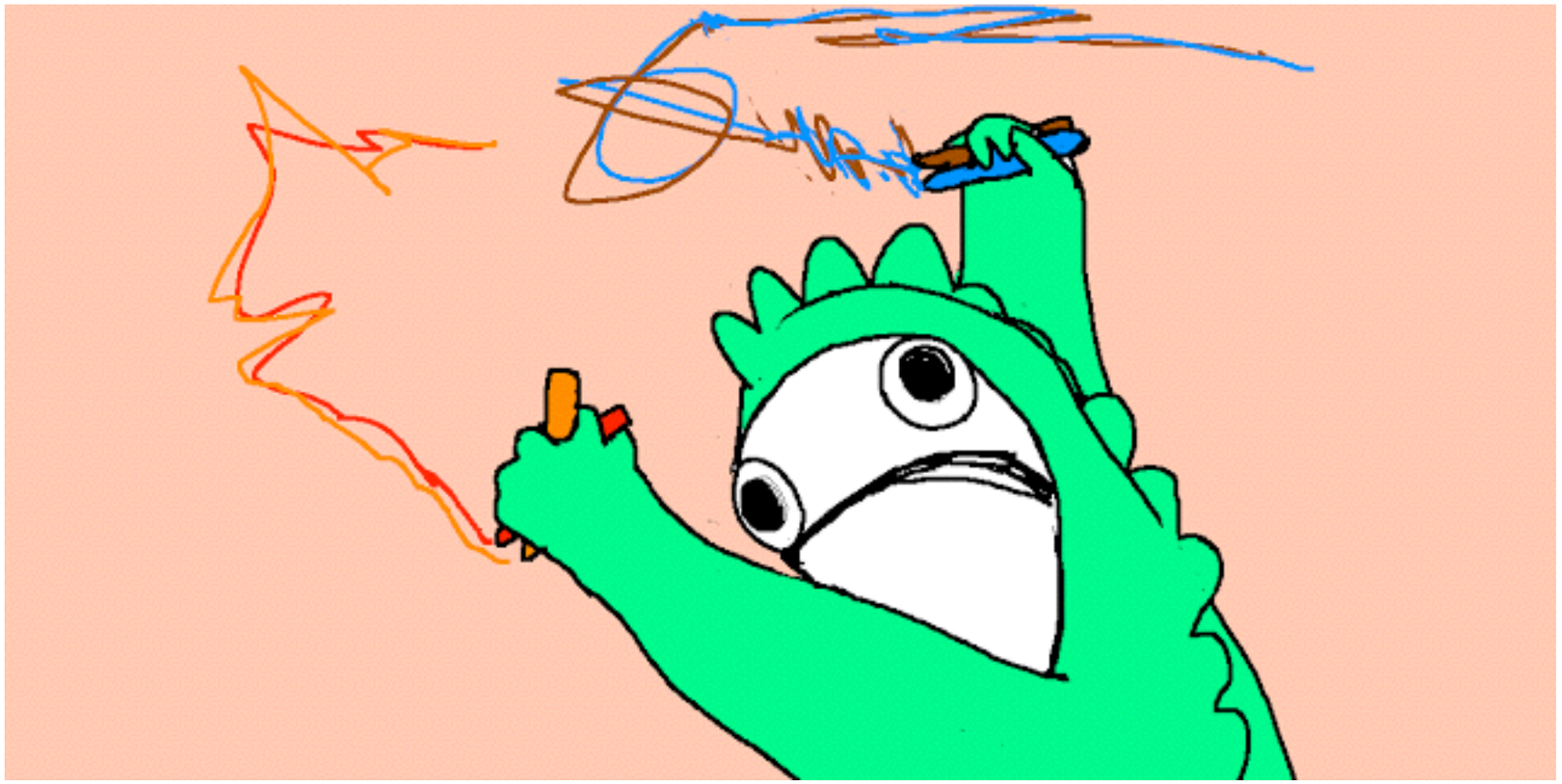
```
bars(
 formula = happy ~ treatment + gender,
 data = expt,
 errorFun = FALSE
)
```




```
bars(
 formula = happy ~ gender + treatment,
 data = expt,
 errorFun = FALSE
)
```

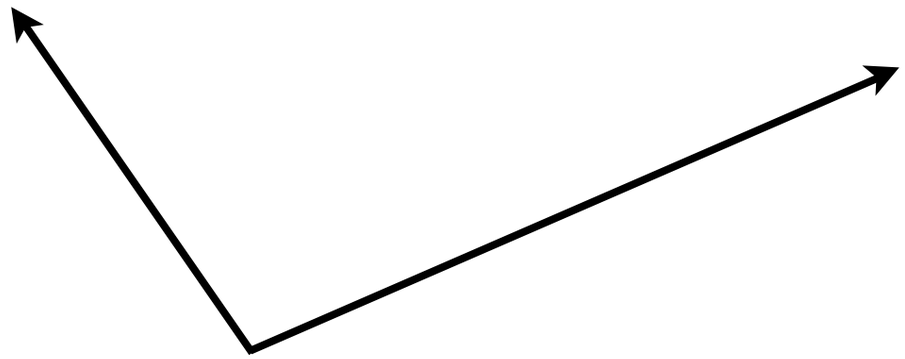


Try it yourself (Exercise 2.2.5)

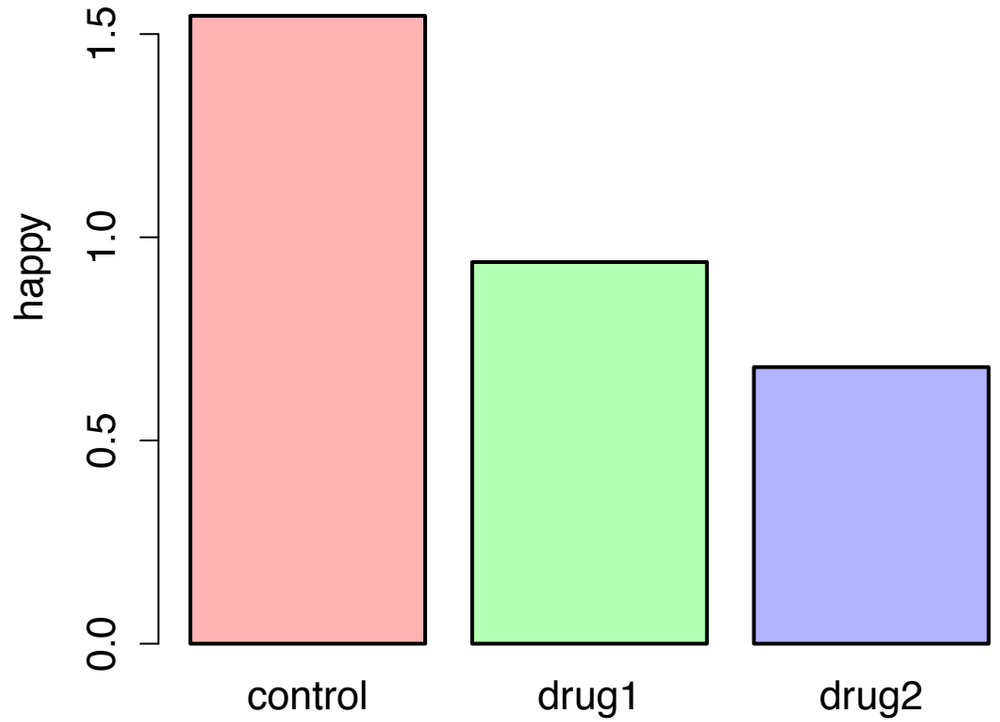


It doesn't have to be the group means

```
bars(
  formula = happy ~ treatment,
  data = expt,
  errorFun = FALSE,
  heightFun = sd
)
```

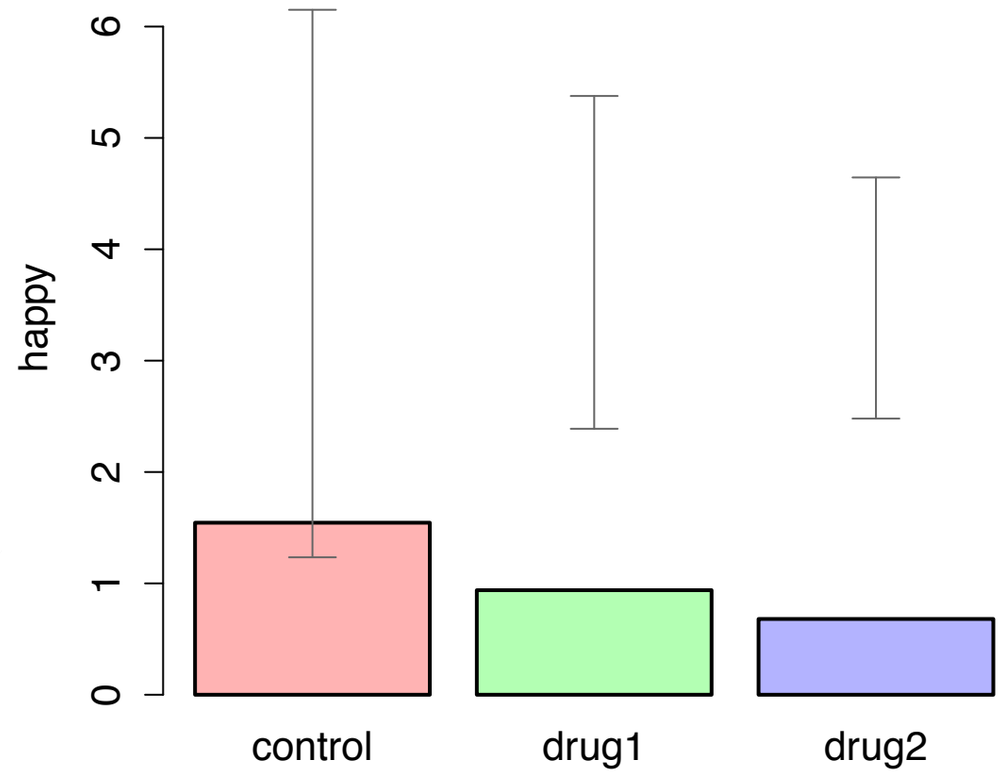


This plots the standard deviation for each group



Warning: if you change the "height" function, you also need to change the error bar function

```
bars(
  formula = happy ~ treatment,
  data = expt,
  heightFun = sd
)
```

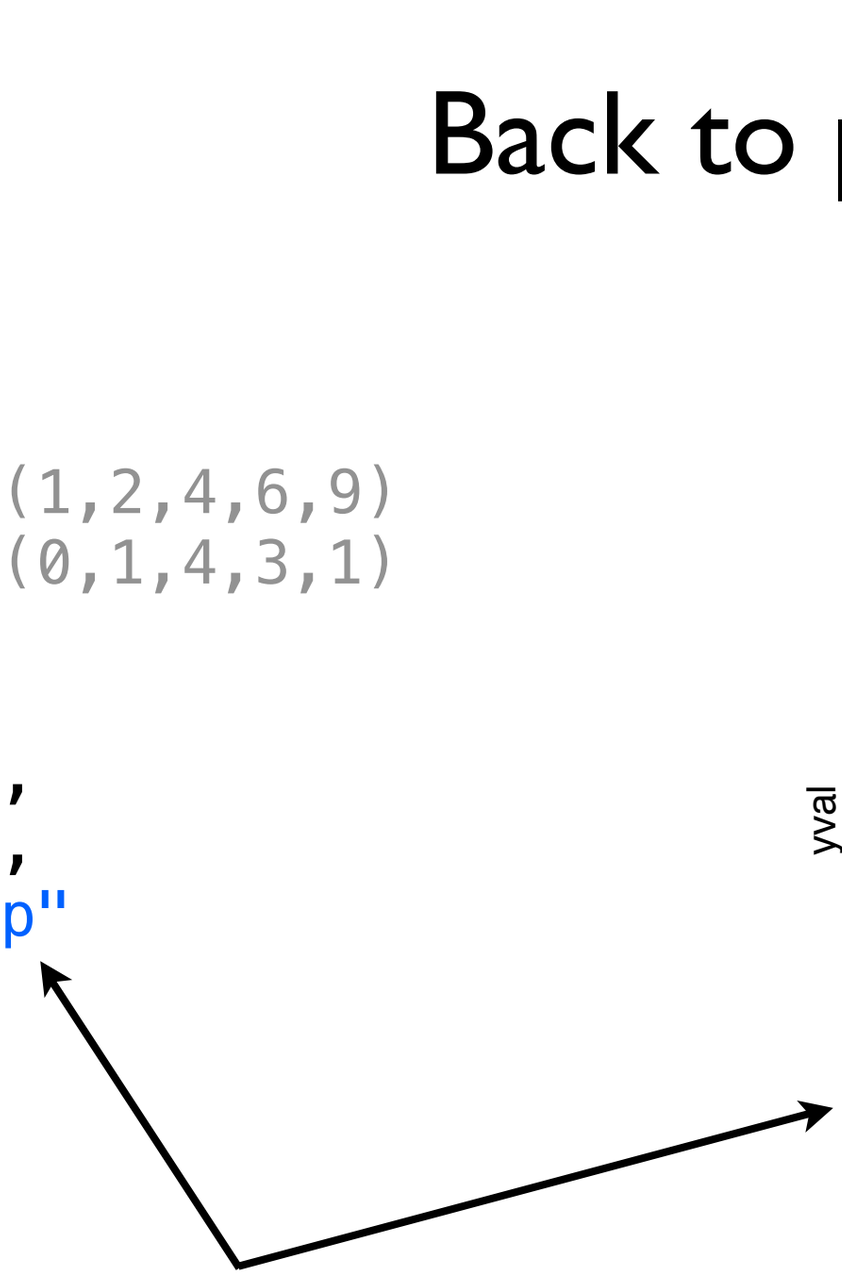


This plot is garbage! The heights are customised to show the group standard deviations, but the error bars are still showing the 95% confidence intervals for the group means!!!!

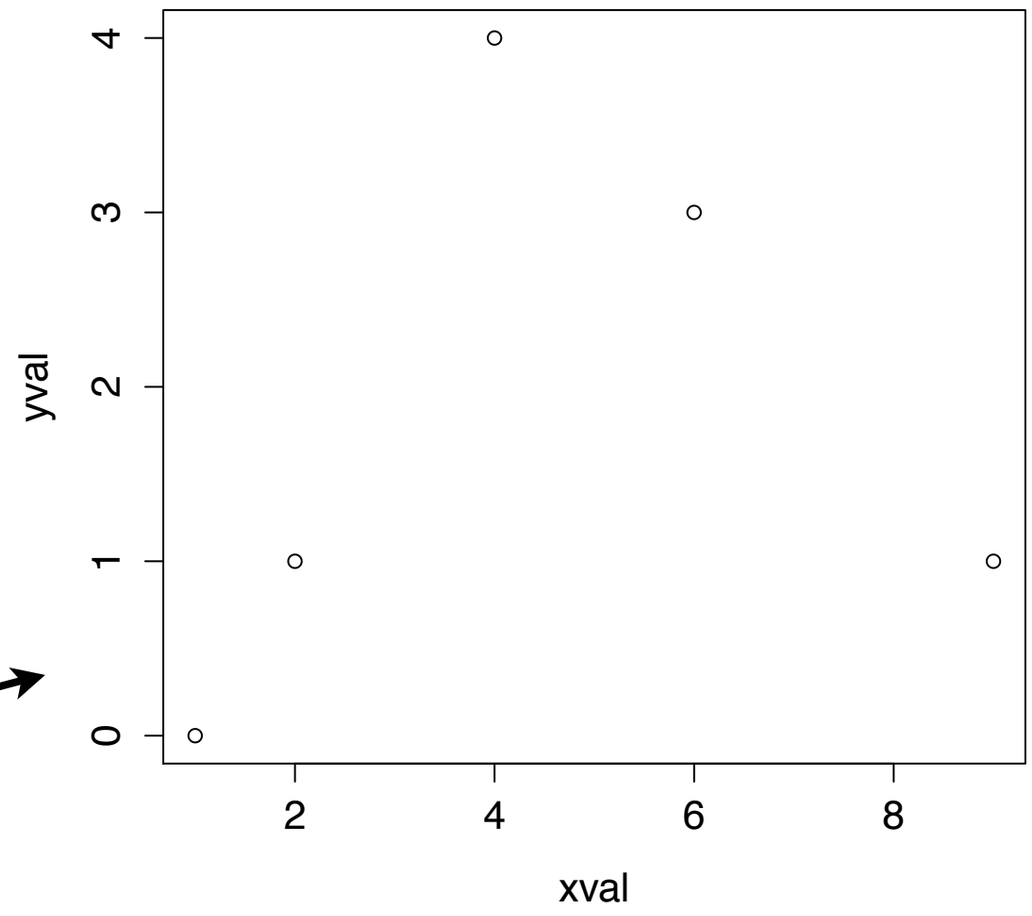
Line graphs
(briefly)

Back to plot()

```
xval <- c(1,2,4,6,9)
yval <- c(0,1,4,3,1)

plot(
 x = xval,
 y = yval,
 type = "p"
)
```



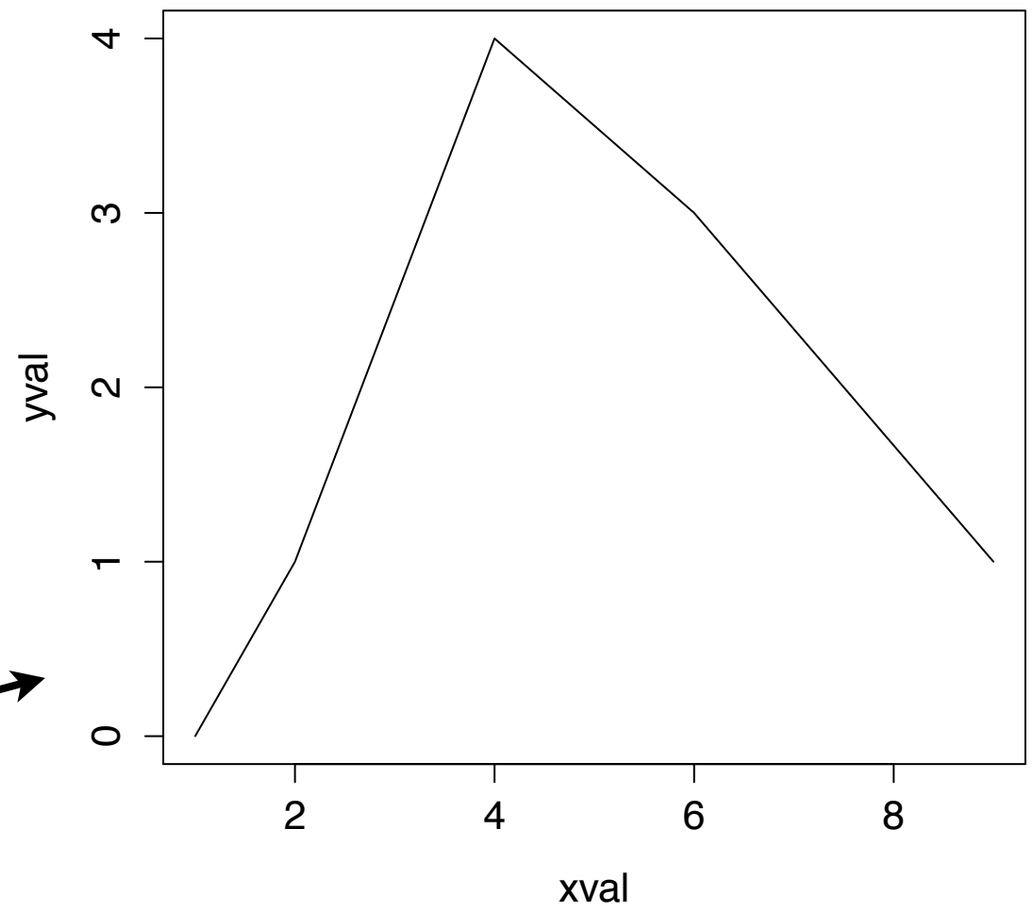
Draw points



Customise the plot “type”

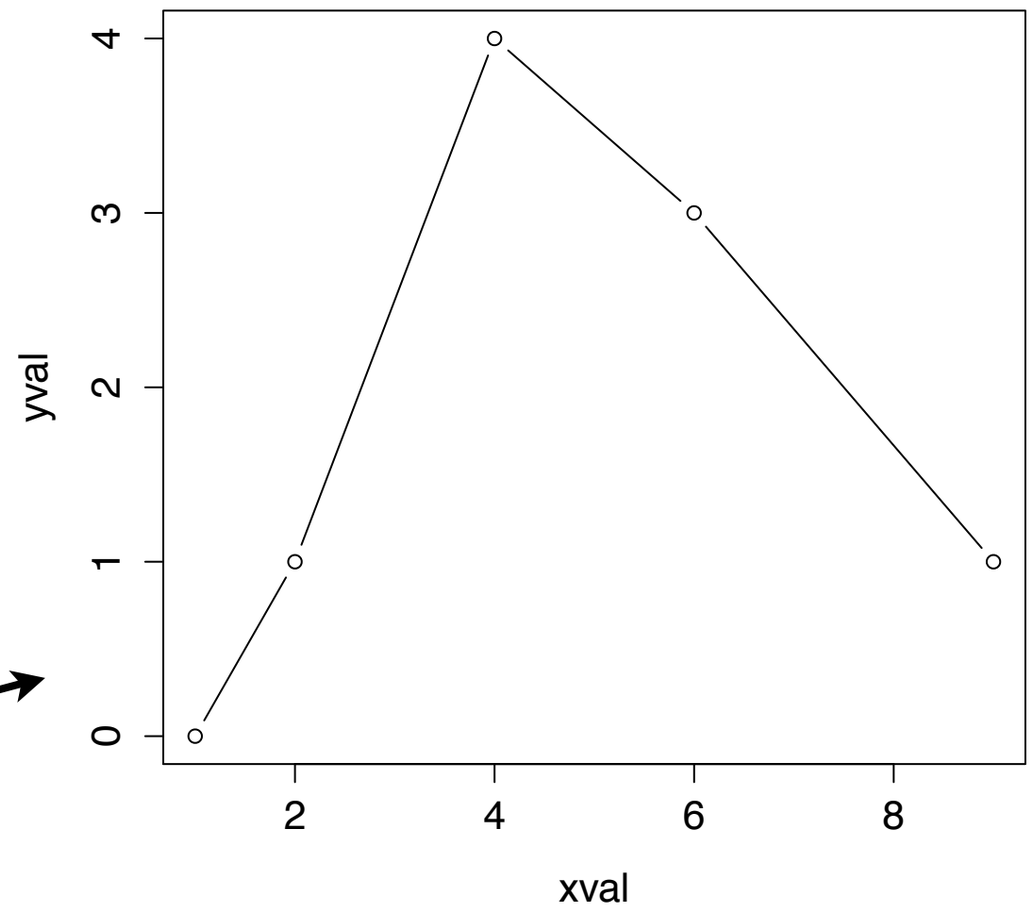
```
xval <- c(1,2,4,6,9)
yval <- c(0,1,4,3,1)

plot(
 x = xval,
 y = yval,
 type = "l"
)
```

Draw a line



Customise the plot “type”

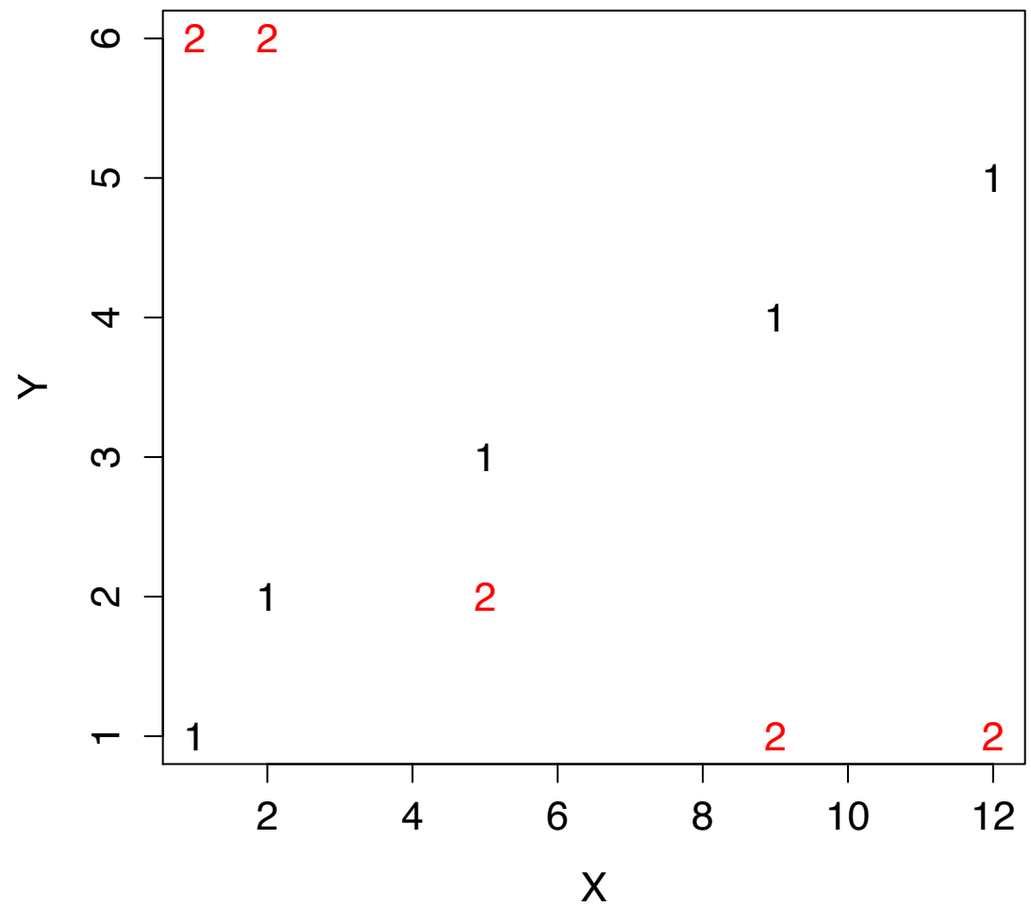
```
xval <- c(1,2,4,6,9)
yval <- c(0,1,4,3,1)

plot(
 x = xval,
 y = yval,
 type = "b"
)
```



Draw both points and a line

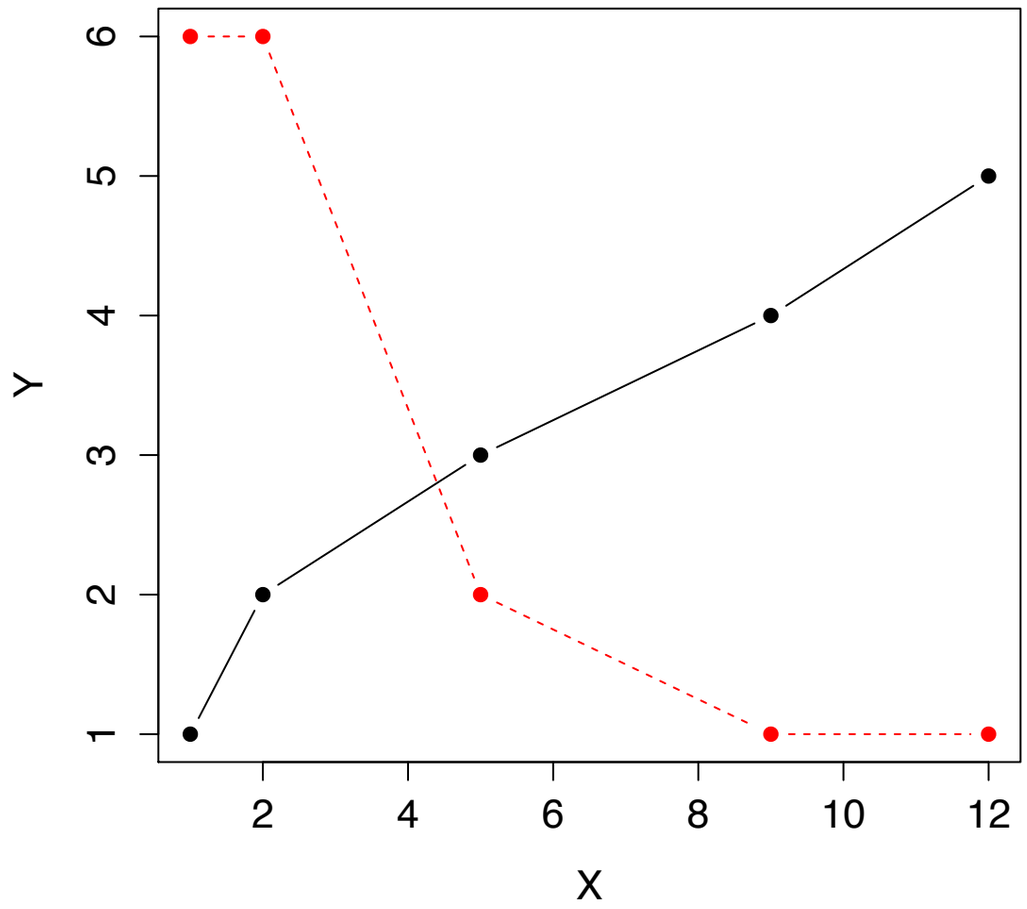
Display a matrix as a scatter plot

```
Y <- cbind( y1=c(1,2,3,4,5),
            y2=c(6,6,2,1,1) )
X <- c(1,2,5,9,12)

matplot( X,Y )
```



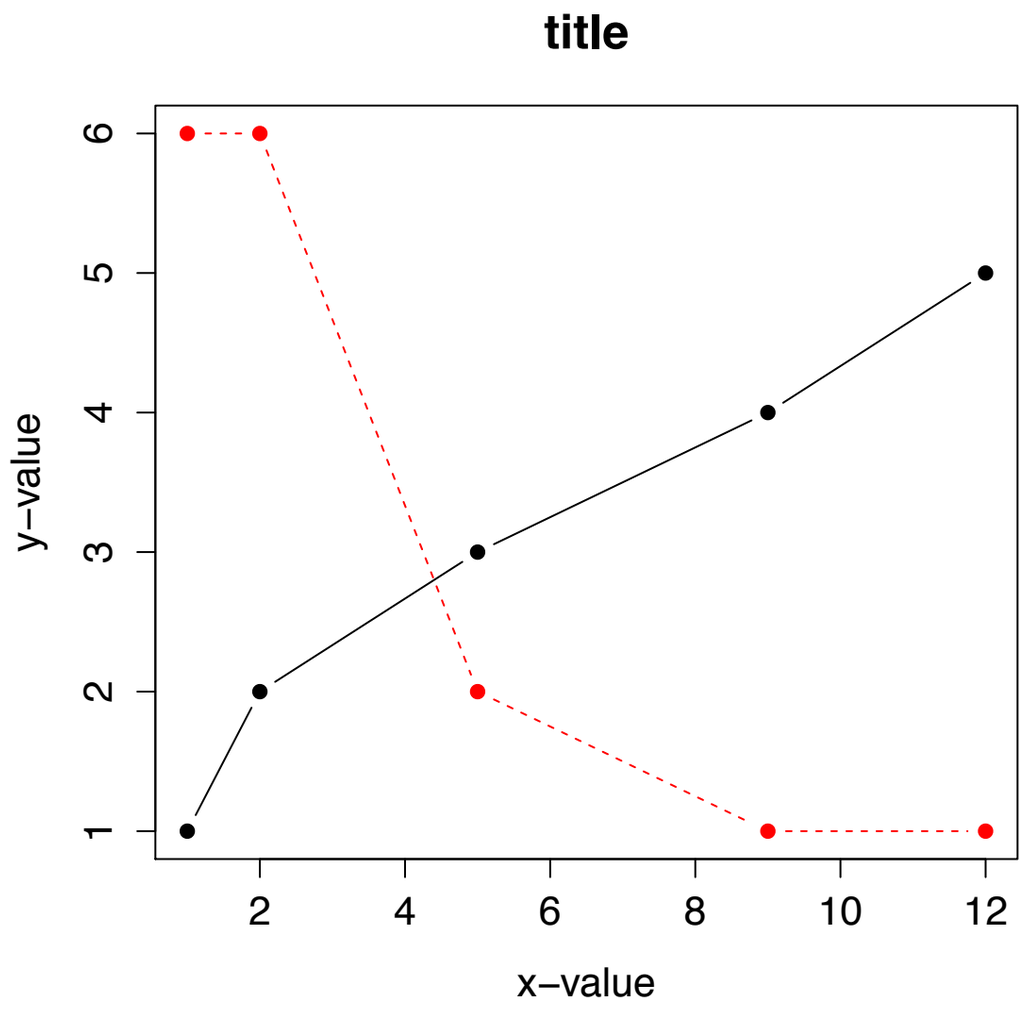
Or as a line plot

```
matplot(
  X,Y,
  type = "b",
  pch = 19
)
```



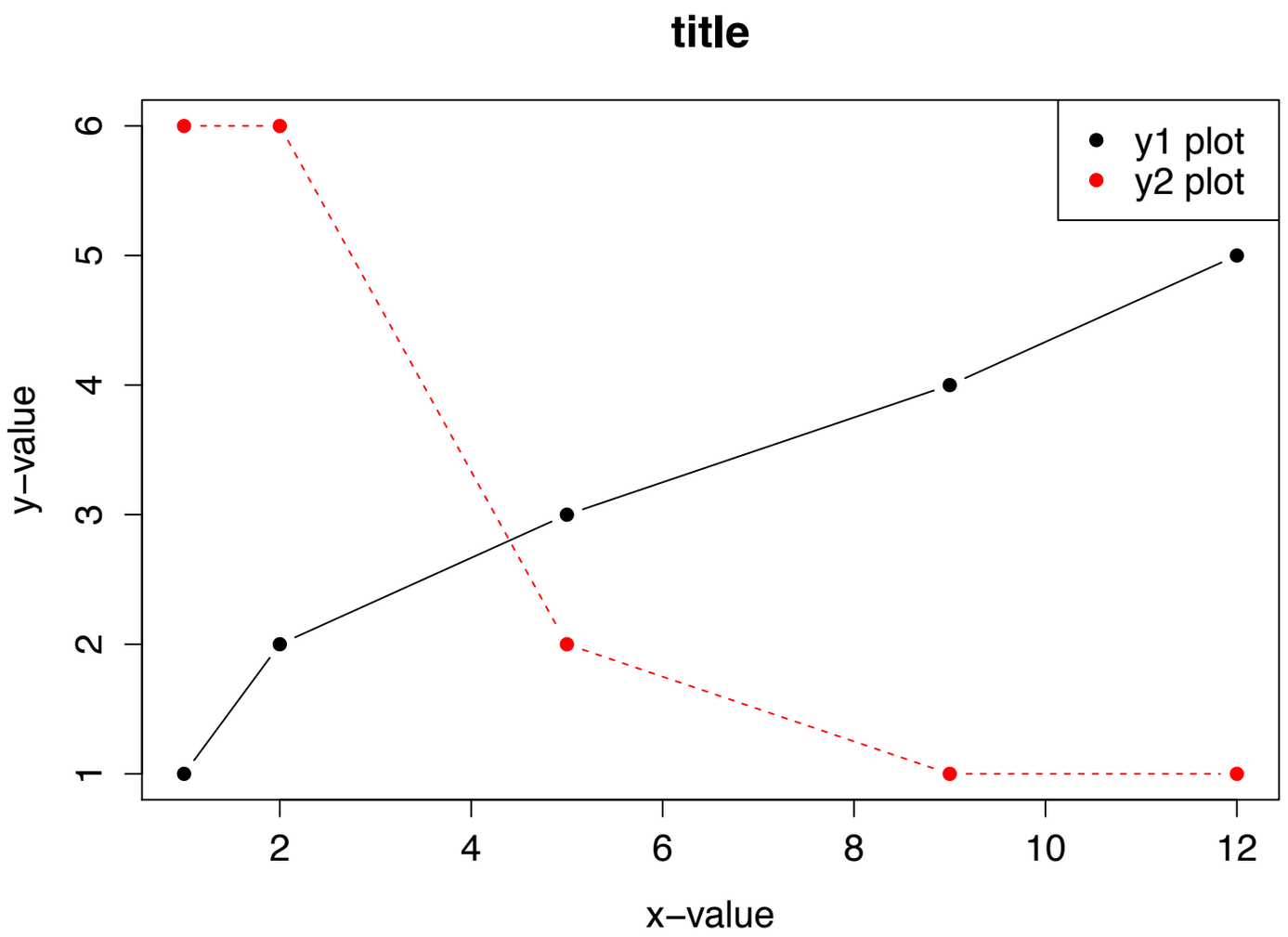
Make it pretty if you like...

```
matplot( X,Y,
   type="b",
   pch=19,
   xlab="x-value",
   ylab="y-value",
   main="title"
)
```



Manually add the legend...

```
matplot( X,Y,
  type="b",
  pch=19,
  xlab="x-value",
  ylab="y-value",
  main="title"
)

legend( "topright",
  legend = c("y1 plot","y2 plot"),
  pch = 19,
  col = c("black","red")
)
```



Saving the output to files

You can get low level control of the image file using commands...

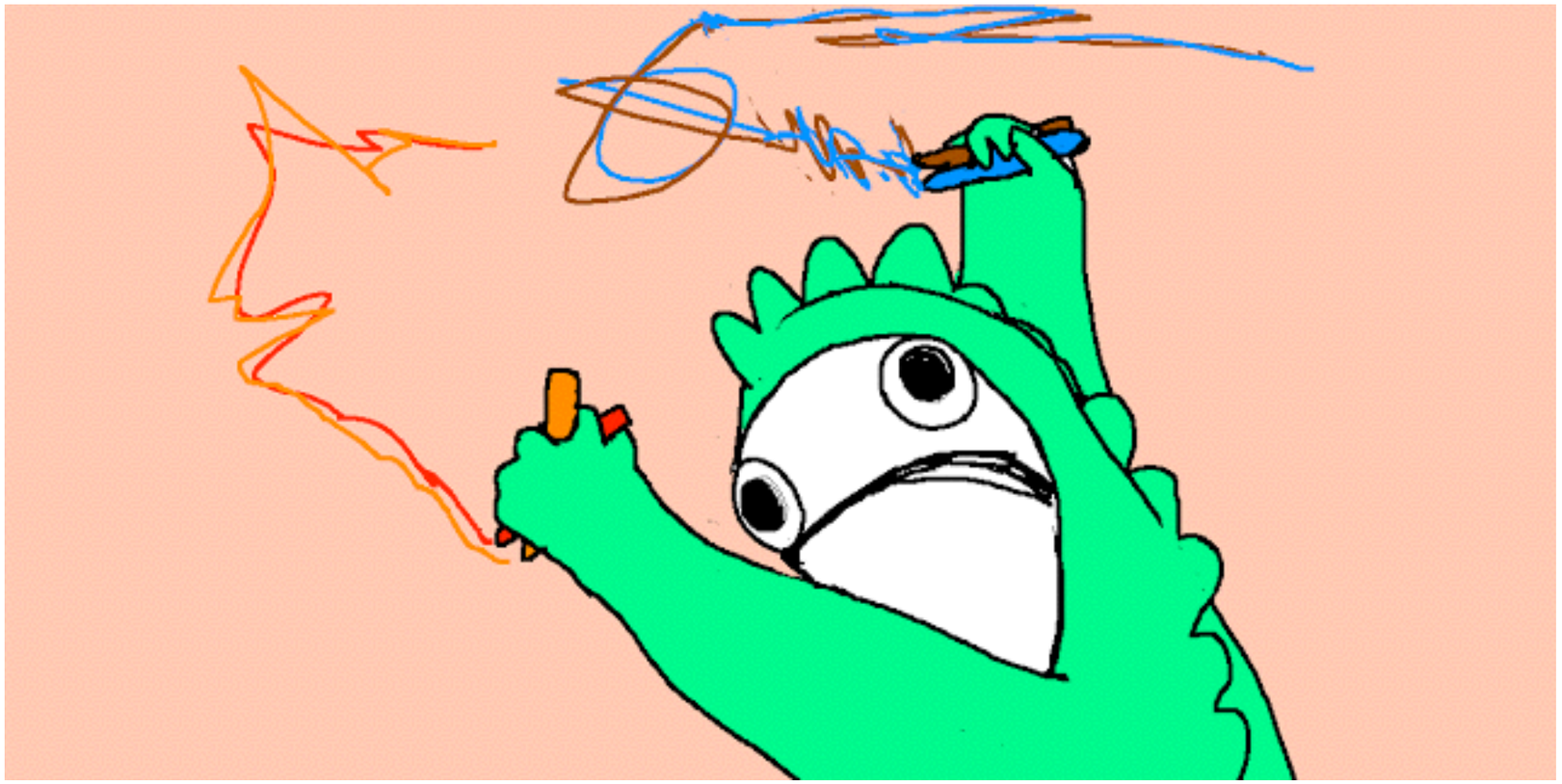
```
dev.print(
  device = pdf,
  file = "../my_img/scatter5.pdf",
  width = 8,
  height = 8
)
```

This command would export the currently visible plot to a pdf file

But it's easier to use Rstudio

In the plots tab, click on the "export" button and then select either the "save plot as image" or "save plot as pdf" options... then follow the instructions it gives you!

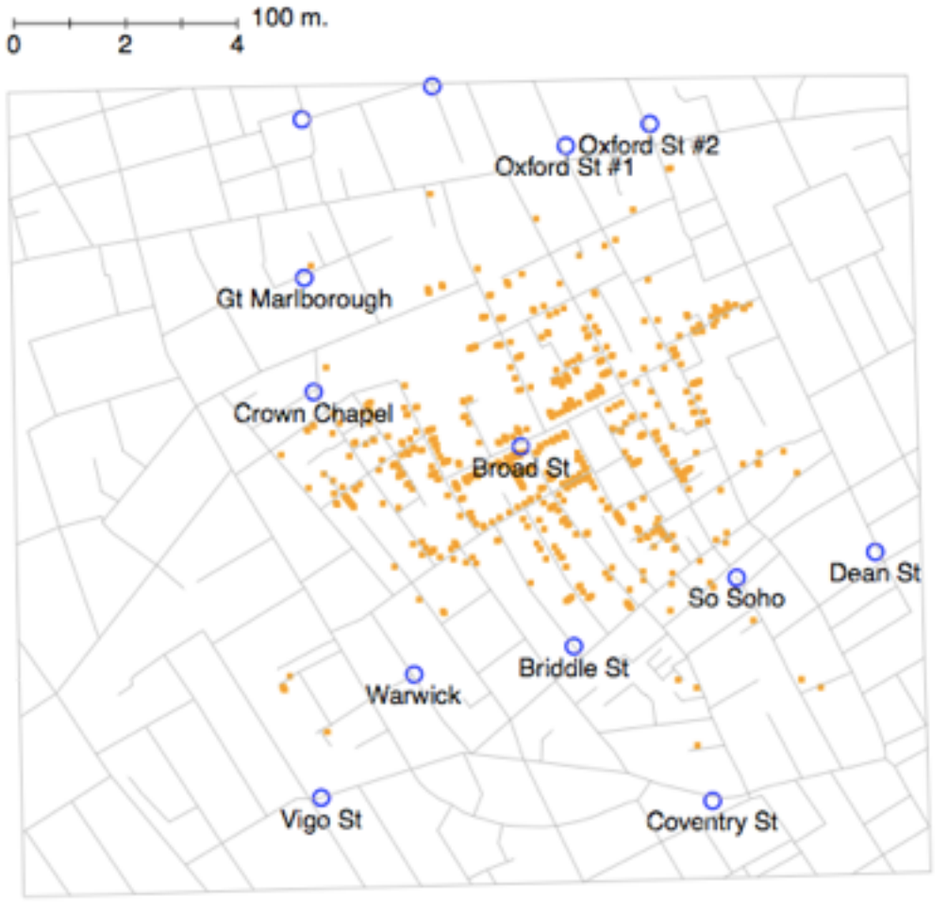
Try it yourself (Exercise 2.2.6)



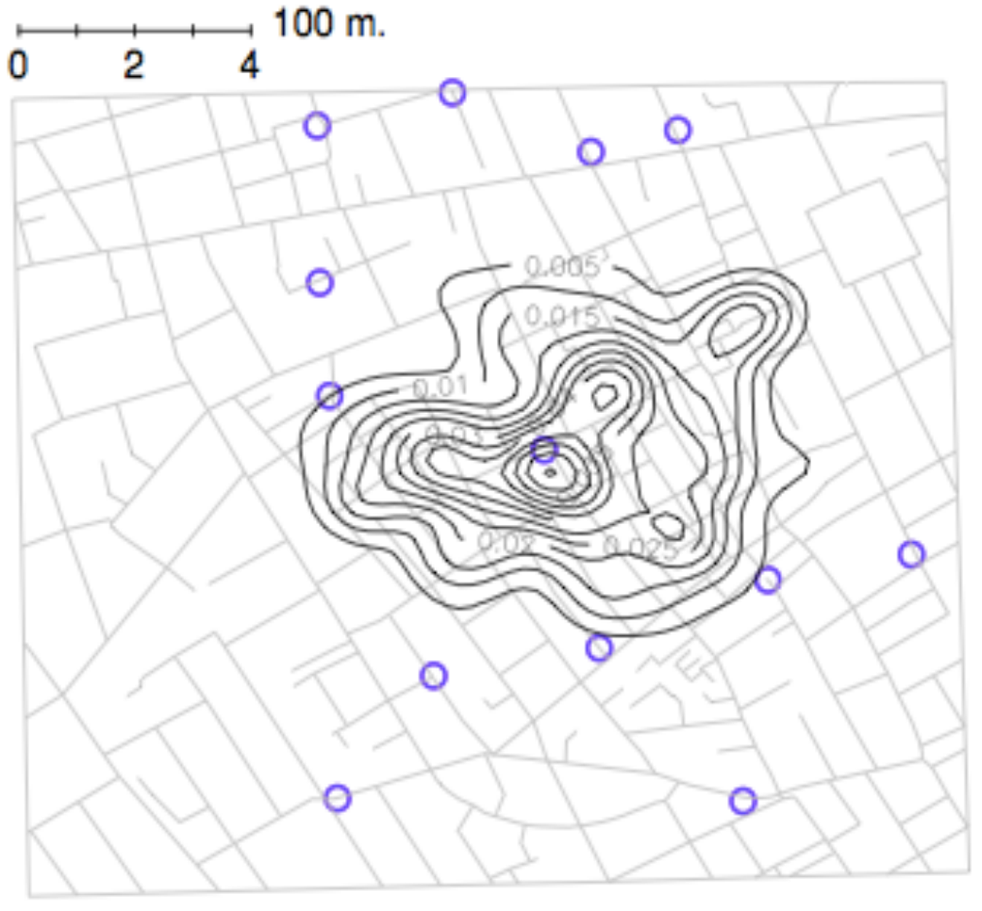
Cute examples of advanced stuff

(no real reason!)

Snow's cholera map of London



Distribution of cholera cases



End of this section