Graphics in R

Plots with One Variable

One Variable Plots

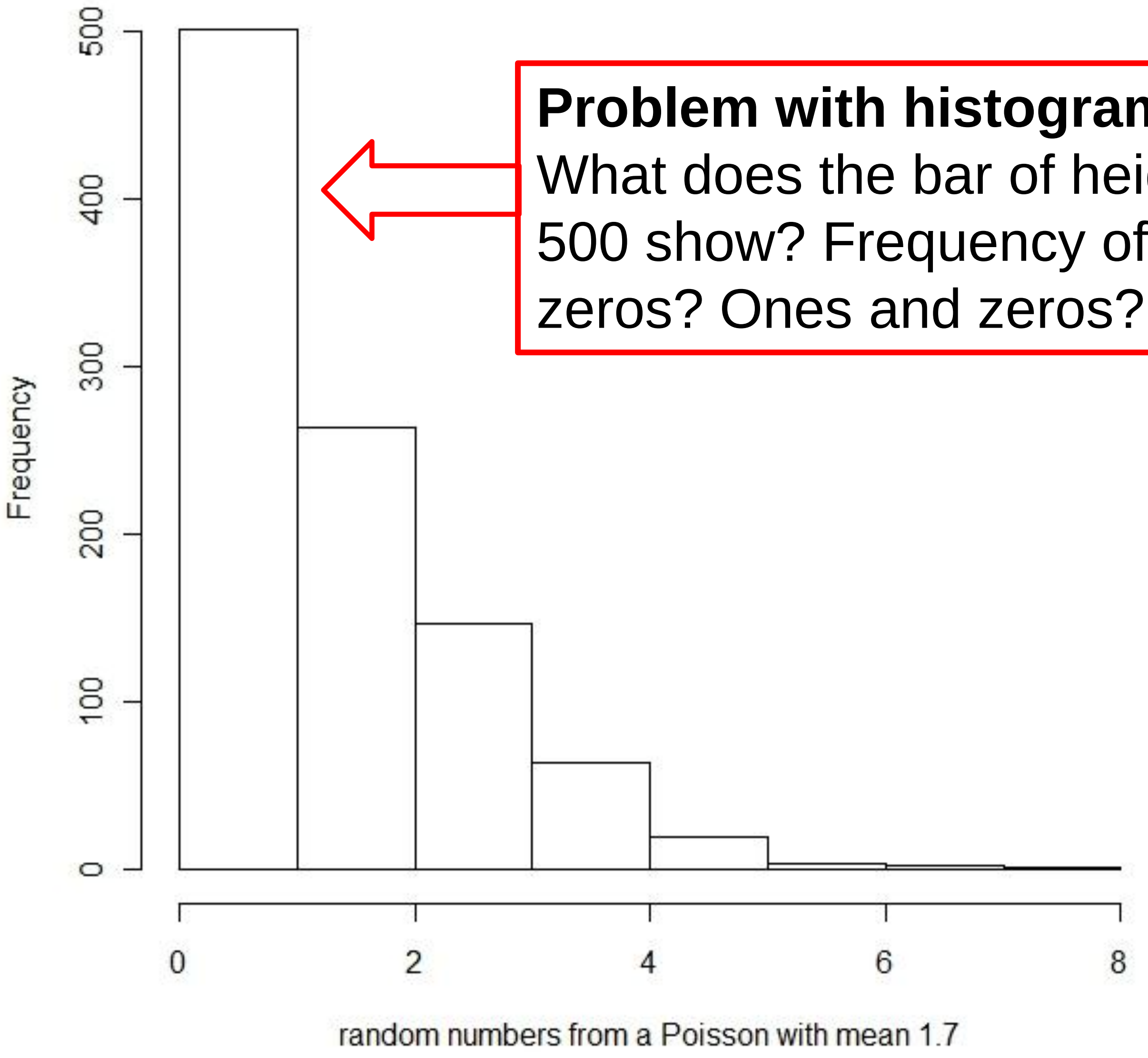
- With **one variable**, choice of plots is restricted:
  - **Histograms** to show a frequency distribution;
  - **Index plots** to show the values of *y* in sequence;
  - **Time series** plots;
  - **Compositional plots** like pie diagrams.

Histograms

- **Histograms** are excellent for showing *mode*, the *spread*, and the *symmetry* (skew) of a set of data.
- Here is the script for a histogram of 1,000 random points drawn from a Poisson distribution with a mean of 1.7:

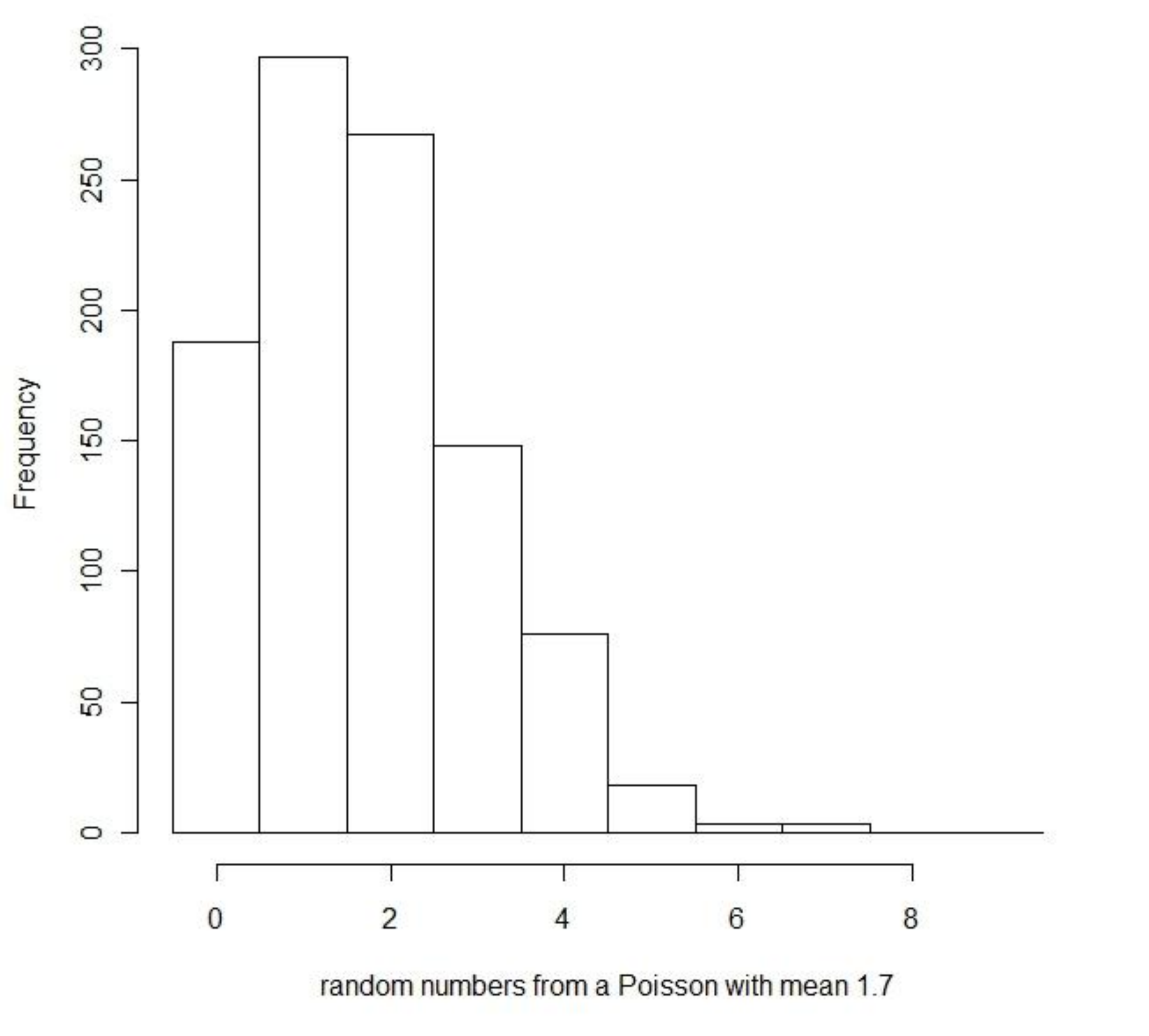
```
> hist(rpois(1000,1.7),main="",xlab="random
+ numbers from a Poisson with mean 1.7")
```

Histograms



Problem with histograms: What does the bar of height 500 show? Frequency of zeros? Ones and zeros?

Histograms



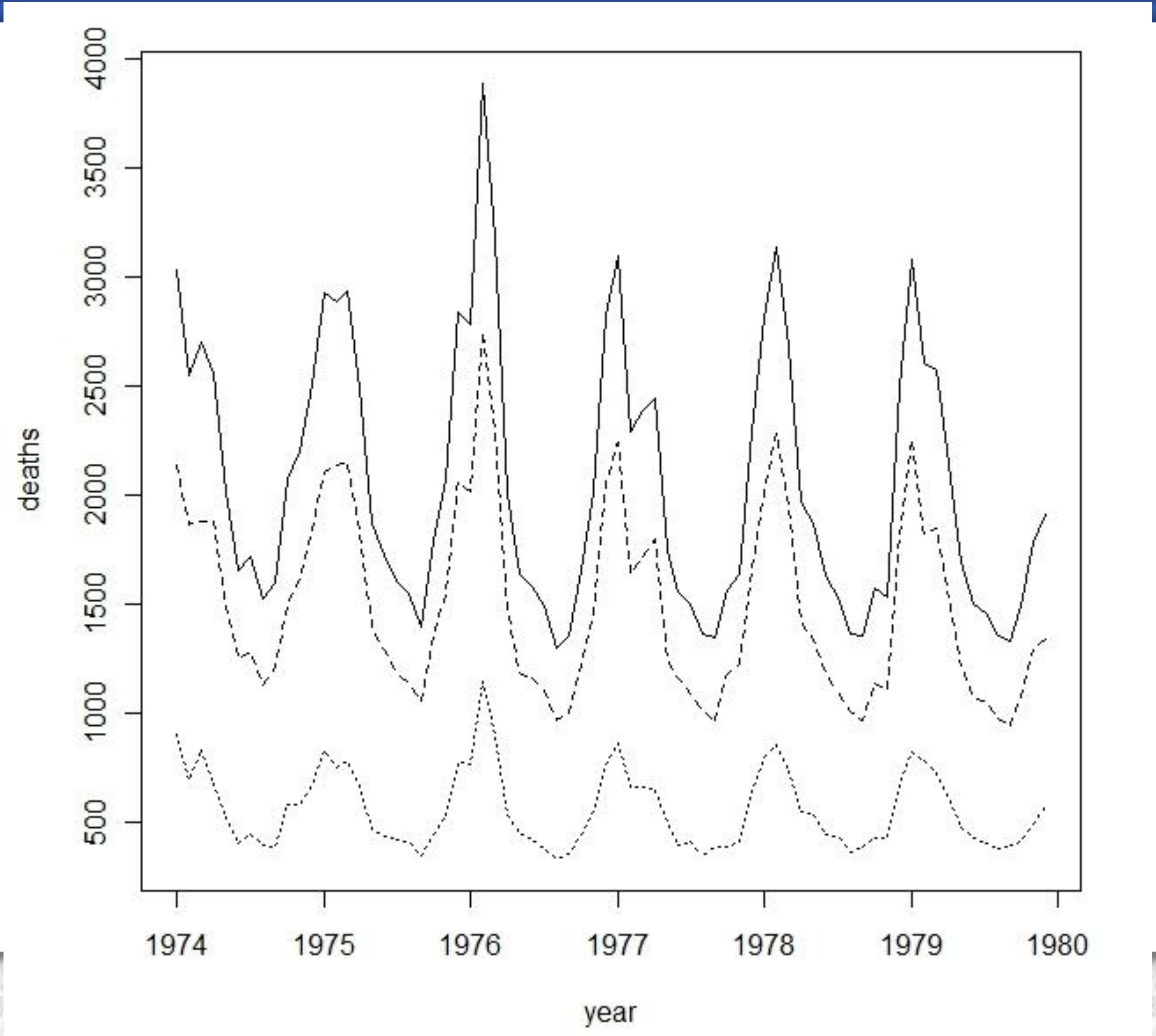
Frequency
0
50
100
150
200
250
300
0
2
4
6
8
random numbers from a Poisson with mean 1.7

Time Series Plots

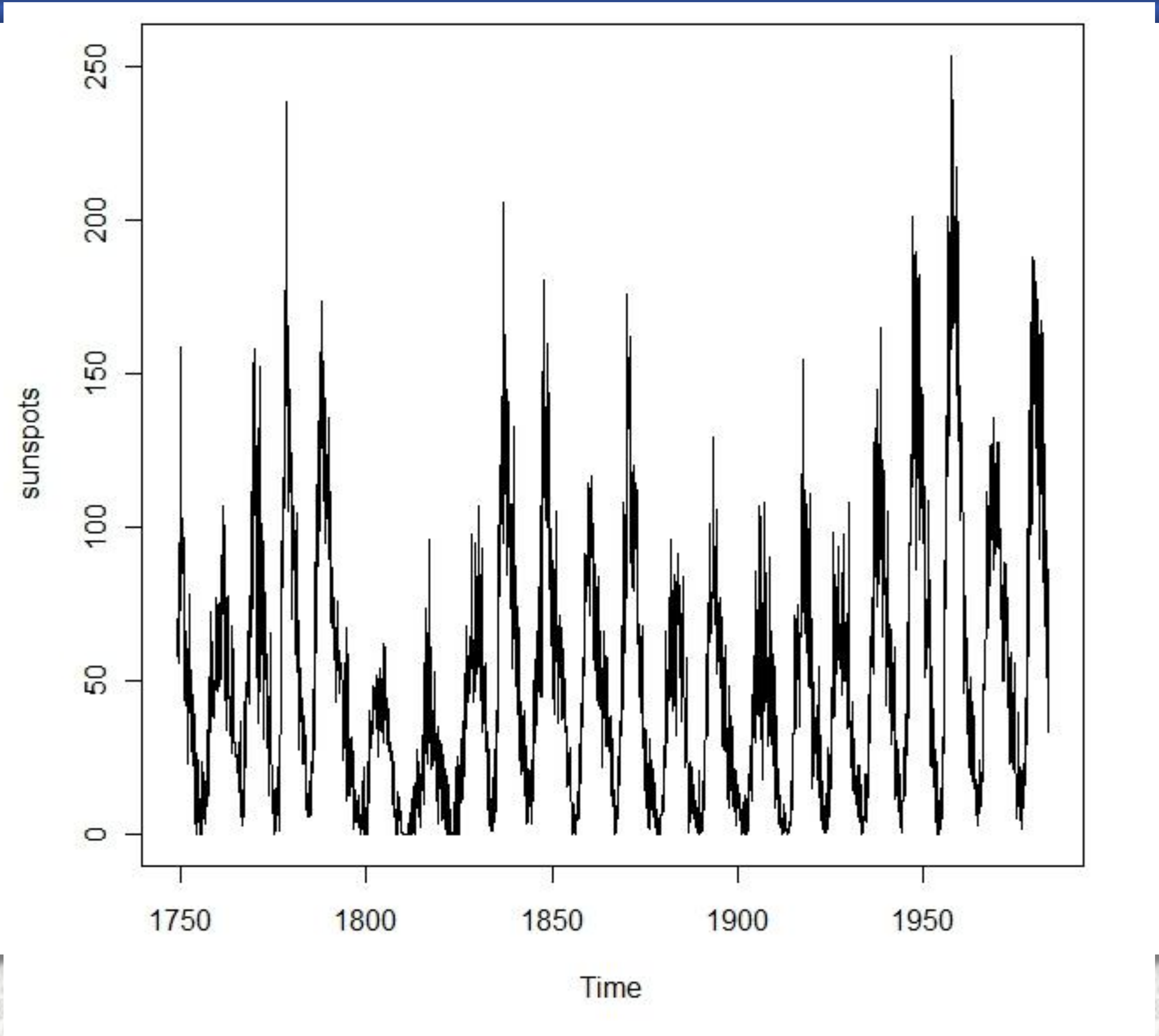
- **Time series** amount to ‘joining dots’ in an ordered set of *y* values.
- The are two R functions for plotting time series data: ts.plots() and plot.ts()

```
> data(UKLungDeaths)
> ts.plot(ldeaths,mdeaths,fdeaths,xlab="year",ylab="deaths",
+ lty=c(1:3))
> data(sunspots)
> plot.ts(sunspots)
> class(sunspots)
# [1] "ts"
> is.ts(sunspots)
# [1] TRUE
```

Time Series Plots



Time Series Plots



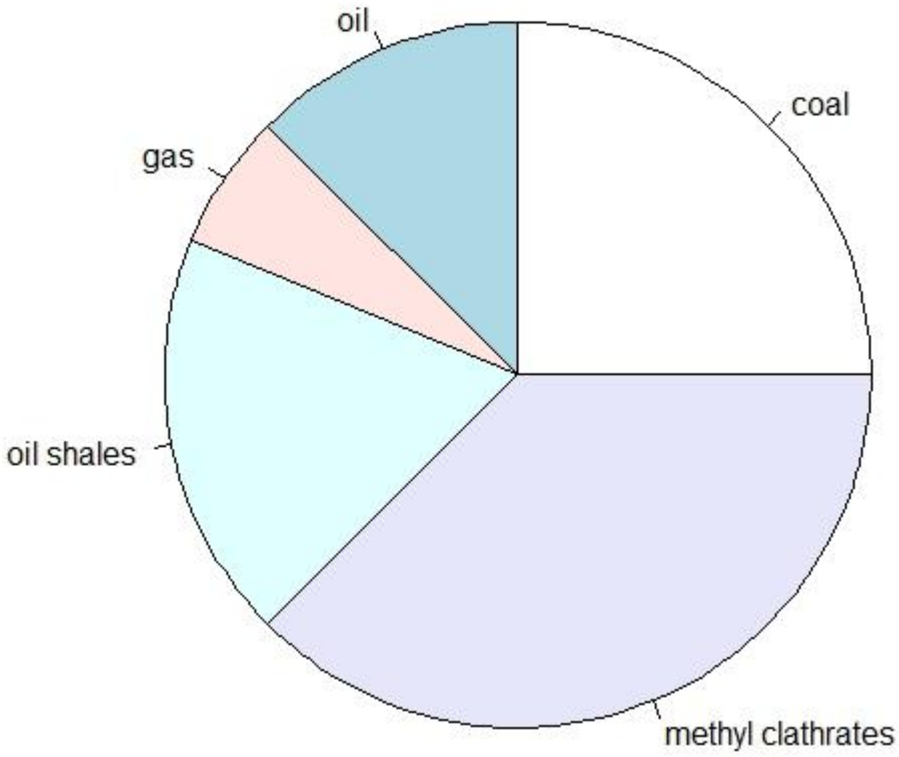
Pie Chart

- **Pie charts** are useful to illustrate the proportional make-up of a sample in presentations.

```
> data<-read.csv("http://www.bio.ic.ac.uk/research/mjcraw/
+ therbook/data/piedata.csv",header=TRUE)
> pie(data$amounts,labels=as.character(data$names))
```

Pie Chart

```
> pie(data$amounts,labels=as.character(data$names))
```



Plots with Two Variables

Two Variable Plots

- With **two variables**, kind of plot depends on nature of *explanatory variable*:
  - **Scatterplot** if continuous
  - **Box-and-whisker plot** if categorical and want to emphasize the scatter
  - **Barplot** if categorical and want to emphasize effect sizes.
- **Most frequent** plotting functions in **R** with two variables:
  - **`plot(x,y)`** scatterplot of y against x
  - **`plot(factor,y)`** box-and-whisker plot of y at factor levels
  - **`barplot(y)`** heights from a vector of y values

Scatterplots

- **`plot()`** function draws axes and adds a scatterplot of points.
- Two additional functions, **`points()`** and **`lines()`**, add extra points or lines to an *existing* plot.
- **Cartesian** approach (syntax):
  - **`plot(x,y)`**
- **Formula** approach (syntax):
  - **`plot(y~x)`**

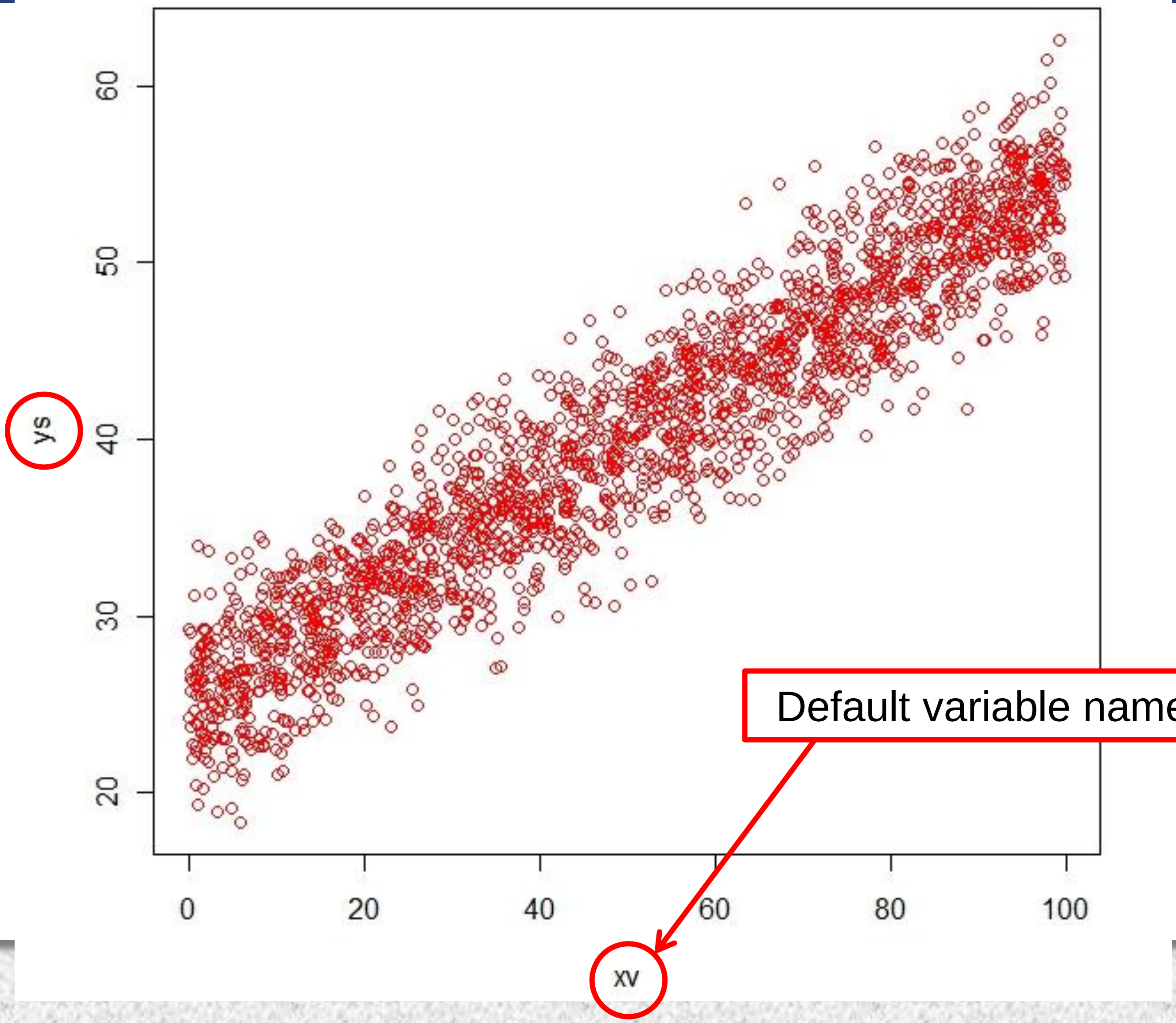
`plot()` Function

- At basic level, `plot()` function needs only two arguments:
    - Name of **explanatory variable** (*x* here)
    - Name of **response variable** (*y* here)

```
> data1<-read.table("http://www.bio.ic.ac.uk/
+ research/mjcraw/therbook/data/scatter1.txt",
+ header=TRUE)
> head(data1)
> attach(data1)
> names(data1)
[1] "xv" "ys"

> plot(xv,ys,col="red")
```

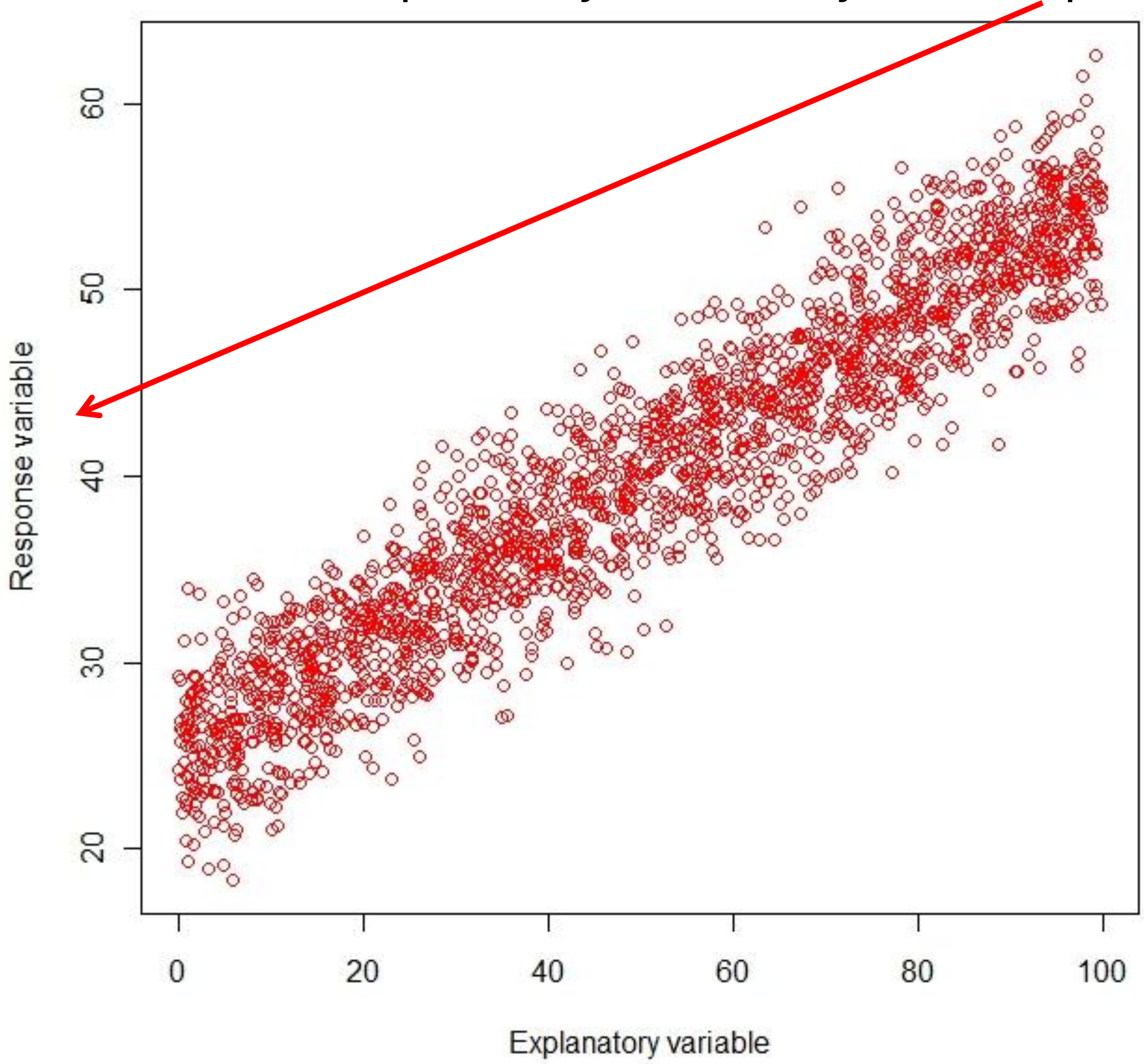
`plot()` Function



Default variable names on axes

`plot()` Function

```
> plot(xv,ys,col="red",xlab="Explanatory variable", ylab="Response variable")
```



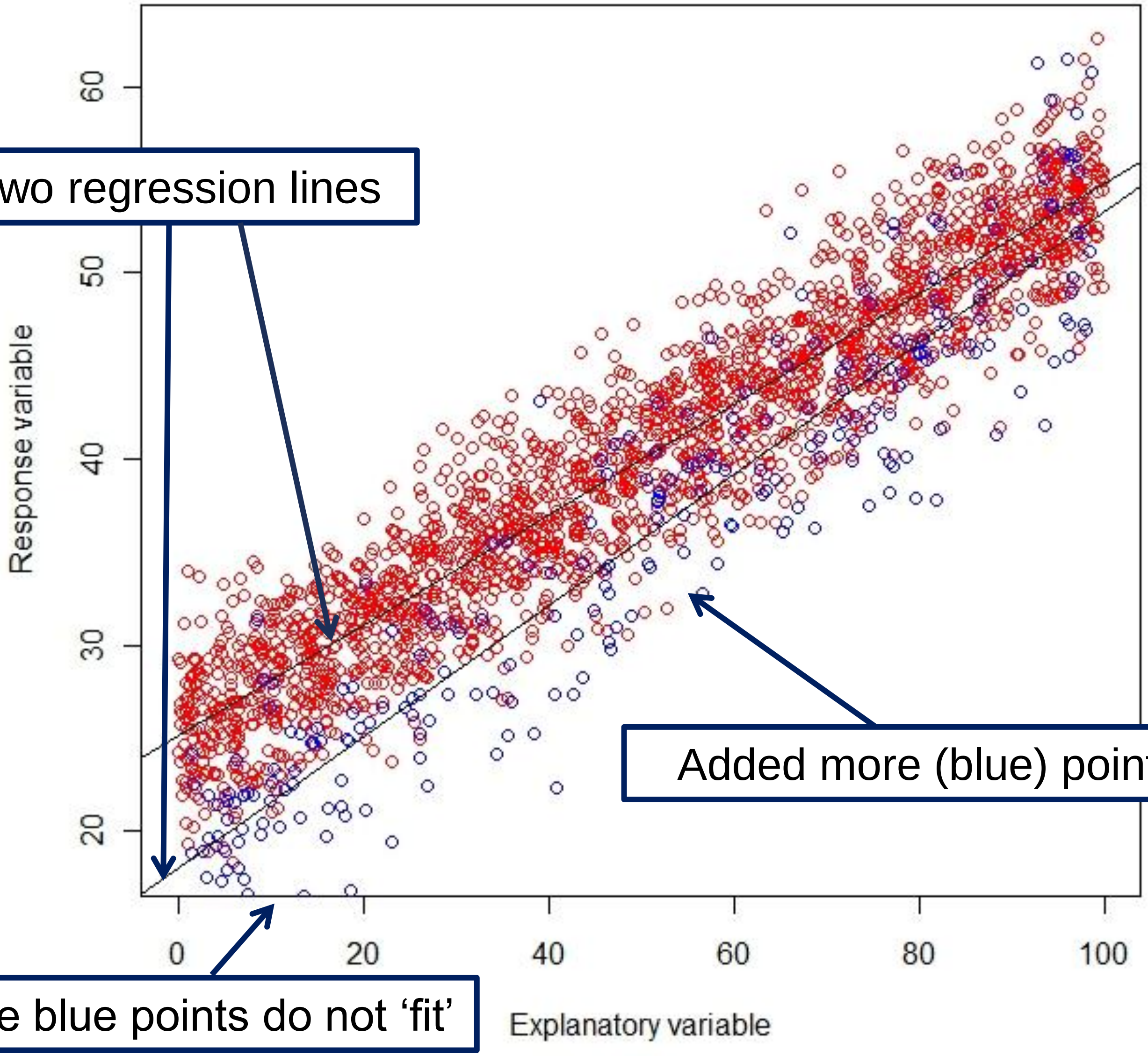
plot() Function

- We add a regression line (**abline()** function) and more points (**points()** function):

```
> abline(lm(ys~xv))
> data2 <- read.table("http://www.bio.ic.ac.uk/
+ research/mjcraw/therbook/data/scatter2.txt",
+ header=TRUE)
> attach(data2)
> names(data2)
[1] "xv2" "ys2"
> points(xv2,ys2,col="blue")
> abline(lm(ys2~xv2))
```

See resulting plot on next slide

plot() Function



Added two regression lines

Added more (blue) points

Notice some blue points do not ‘fit’

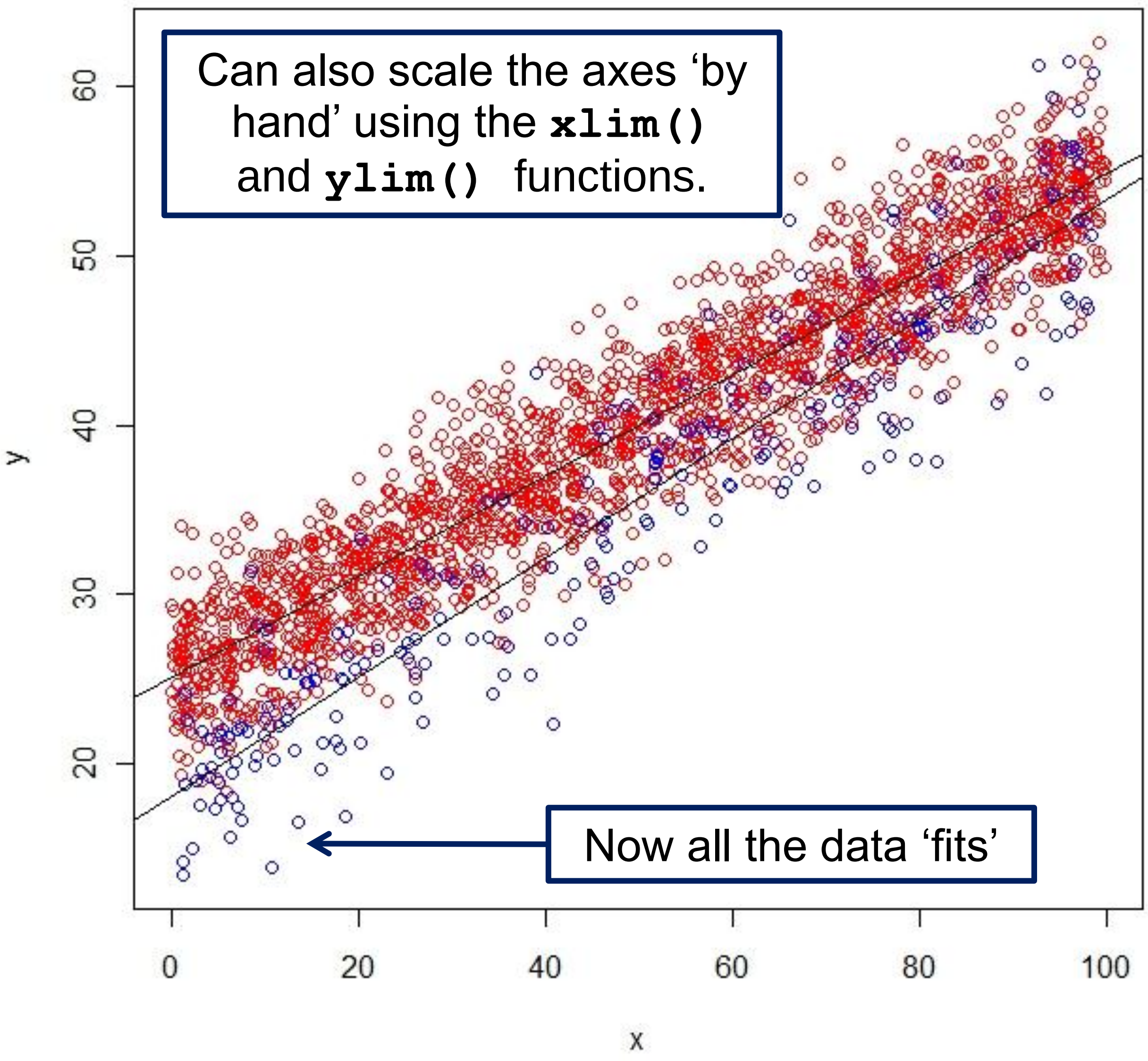
plot() Function

- Fix by plotting all the data with **type=“n”** so axes are scaled:

```
> plot(c(xv,xv2),c(ys,ys2),xlab="x",ylab="y",type="n")
> points(xv,ys,col="red")
> points(xv2,ys2,col="blue")
> abline(lm(ys~xv))
> abline(lm(ys2~xv2))
```

Can also scale the axes yourself using the **xlim()** and **ylim()** functions.

`plot()` Function



Can also scale the axes ‘by hand’ using the **`xlim()`** and **`ylim()`** functions.

Now all the data ‘fits’

plot() Function

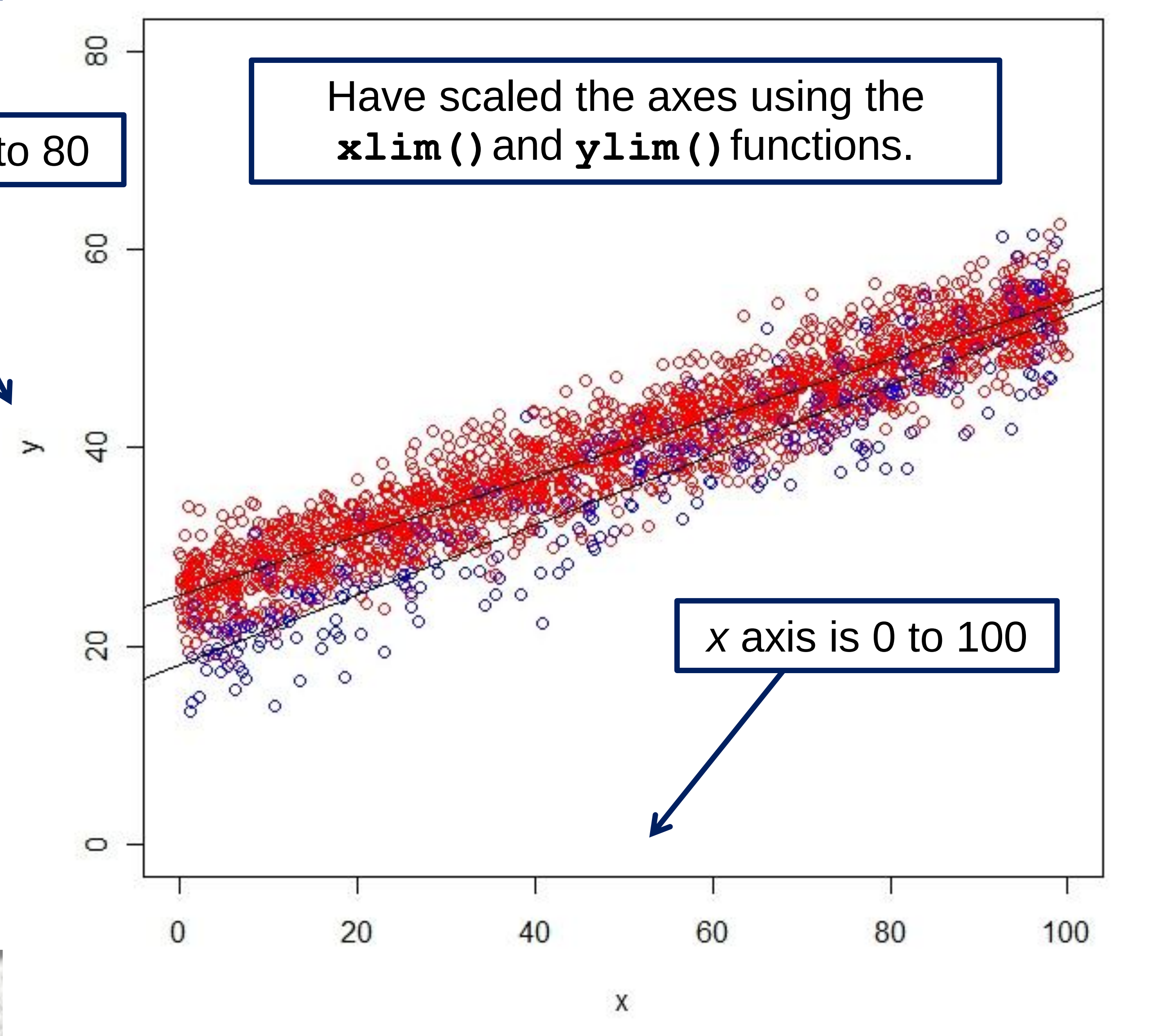
- Can determine the axis values using **range()** function:

```
> range(c(xv,xv2))
[1] 0.02849861 99.93262000

> range(c(ys,ys2))
[1] 13.41794 62.59482

> plot(c(xv,xv2),c(ys,ys2),xlim=c(0,100),ylim=c(0,80),
+ xlab="x",ylab="y",type="n")
> points(xv,ys,col="red")
> points(xv2,ys2,col="blue")
> abline(lm(ys~xv))
> abline(lm(ys2~xv2))
```

`plot()` Function



y axis is 0 to 80

Have scaled the axes using the **`xlim()`** and **`ylim()`** functions.

x axis is 0 to 100

`plot()` Function

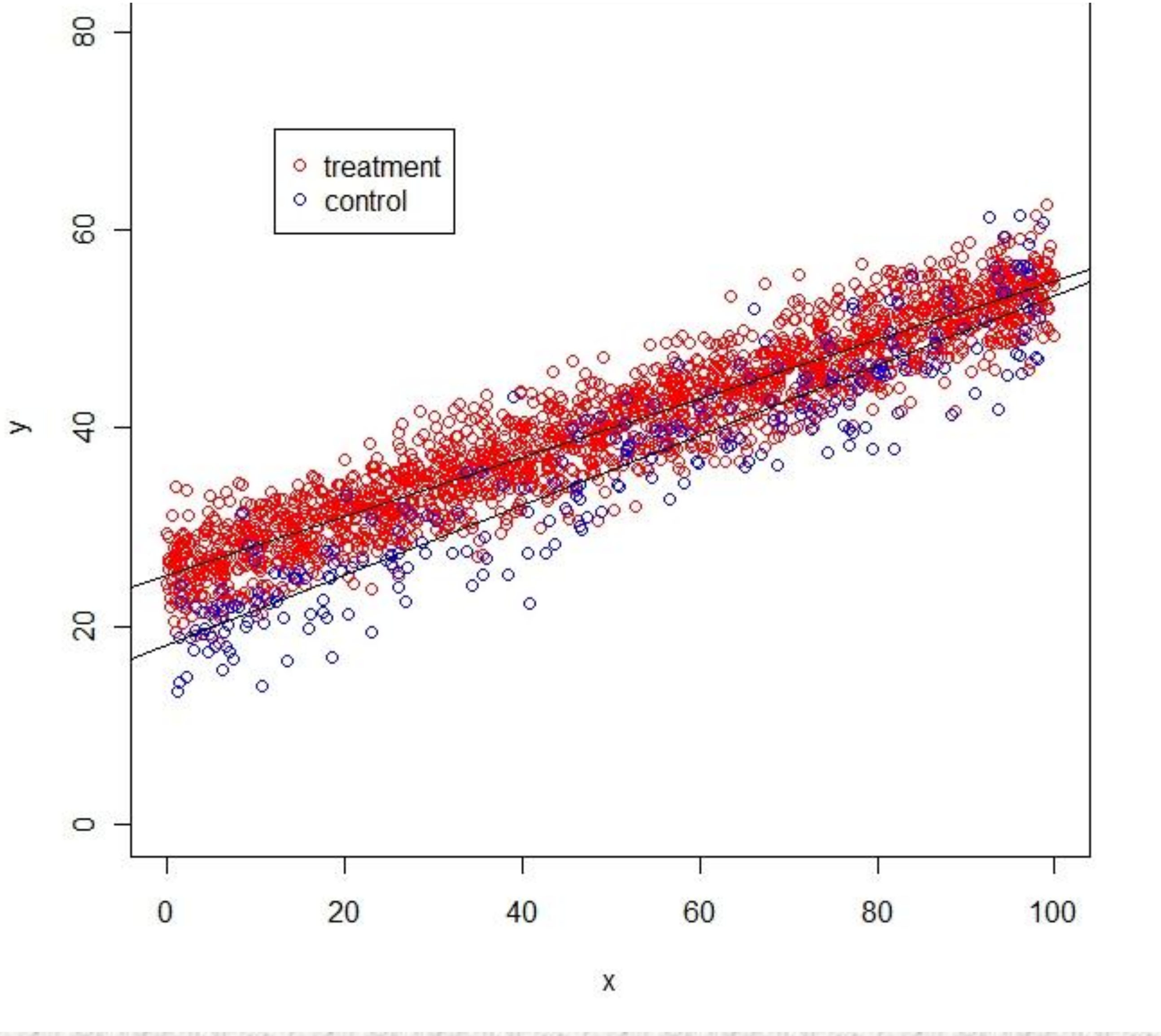
```
> legend(locator(1),c("treatment","control"),pch=c(1,1),col=c(2,4))
```

- Is useful to know the first six colors used by **`plot()`**:

1	black (the default)
2	red
3	green
4	blue
5	pale blue
6	purple

`plot()` Function

```
> legend(locator(1),c("treatment","control"),pch=c(1,1),col=c(2,4))
```

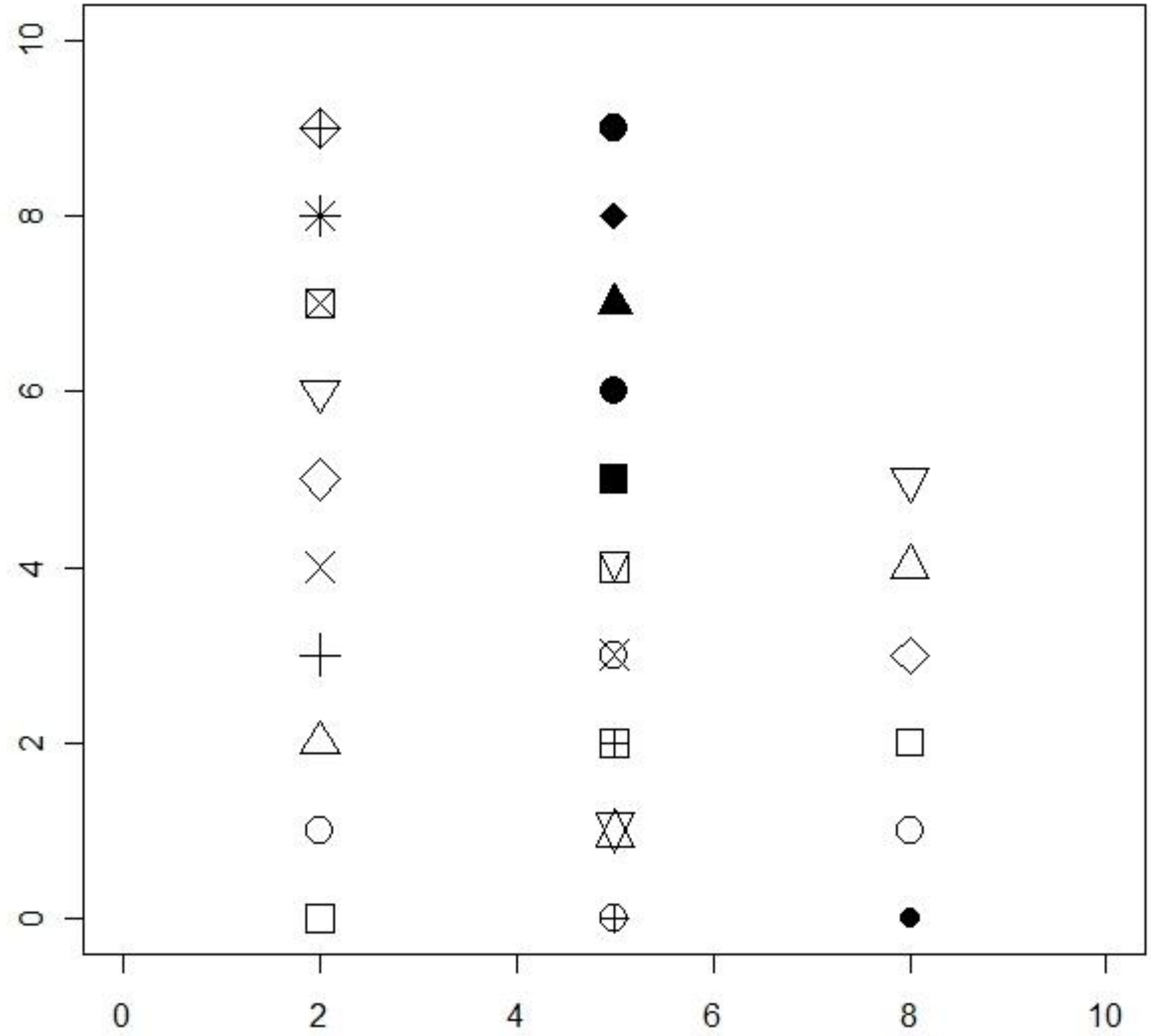


plot() Function

```
> plot(0:10,0:10,type="n",xlab="",ylab="")
> k <- -1
> for(i in c(2,5,8)){
+ for(j in 0:9){
+ k <- k+1
+ points(i,j,pch=k,cex=2)}}
```

This script draws many of the plotting characters

`pch()` Function Plotting Symbols



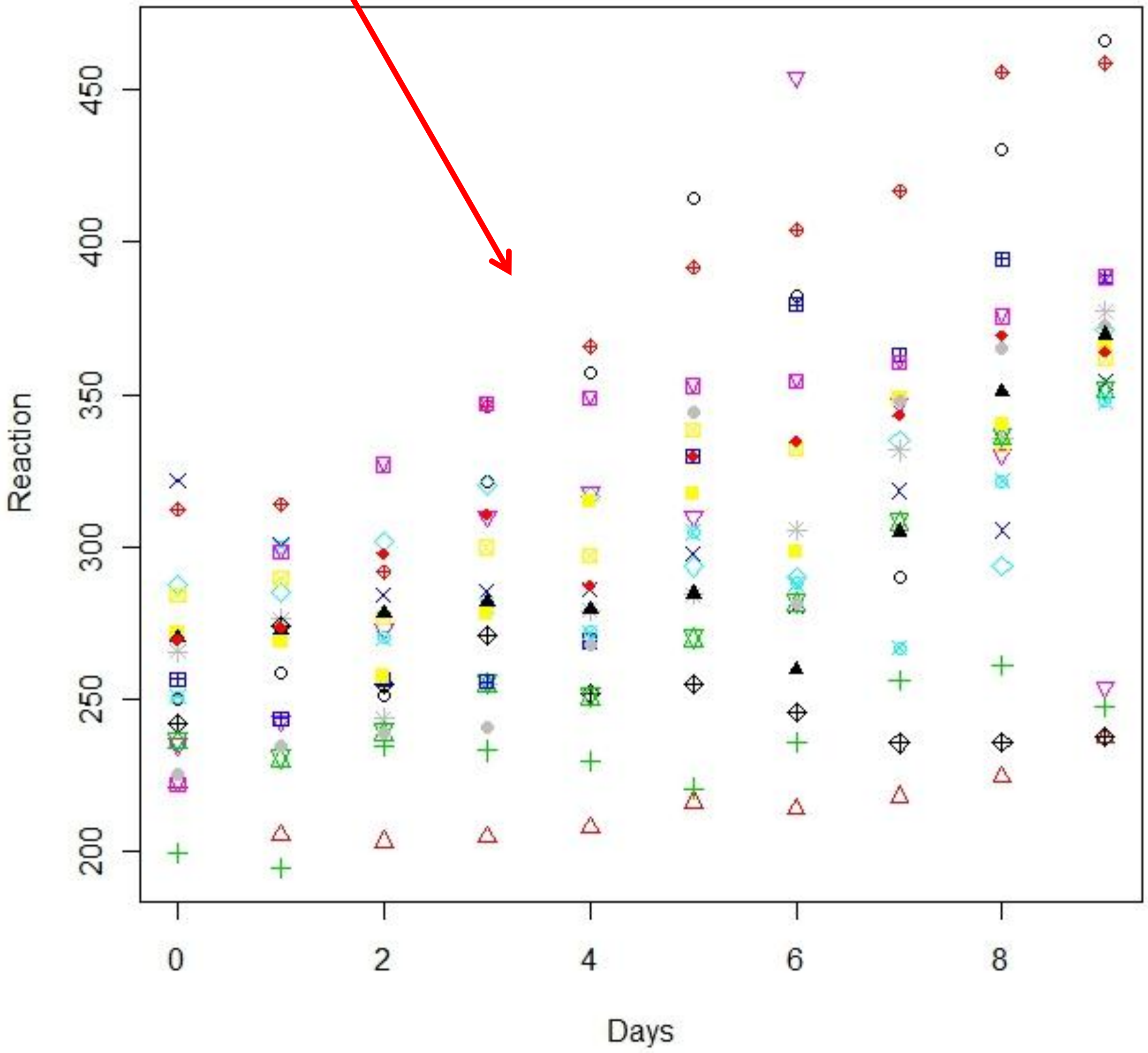
Identifying Individuals

- Use colors and symbols to identify individuals using **`as.numeric()`** to convert the grouping factor:

```
> data <- read.table("http://www.bio.ic.ac.uk/research/mjcraw/
+ therbook/data/sleep.txt",header=T)
> attach(data)
> Subject <- factor(Subject)
> plot(Days,Reaction,col=as.numeric(Subject),pch=as.numeric(Subject))
```

Identifying Individuals

Identify subjects with unique colors and symbols



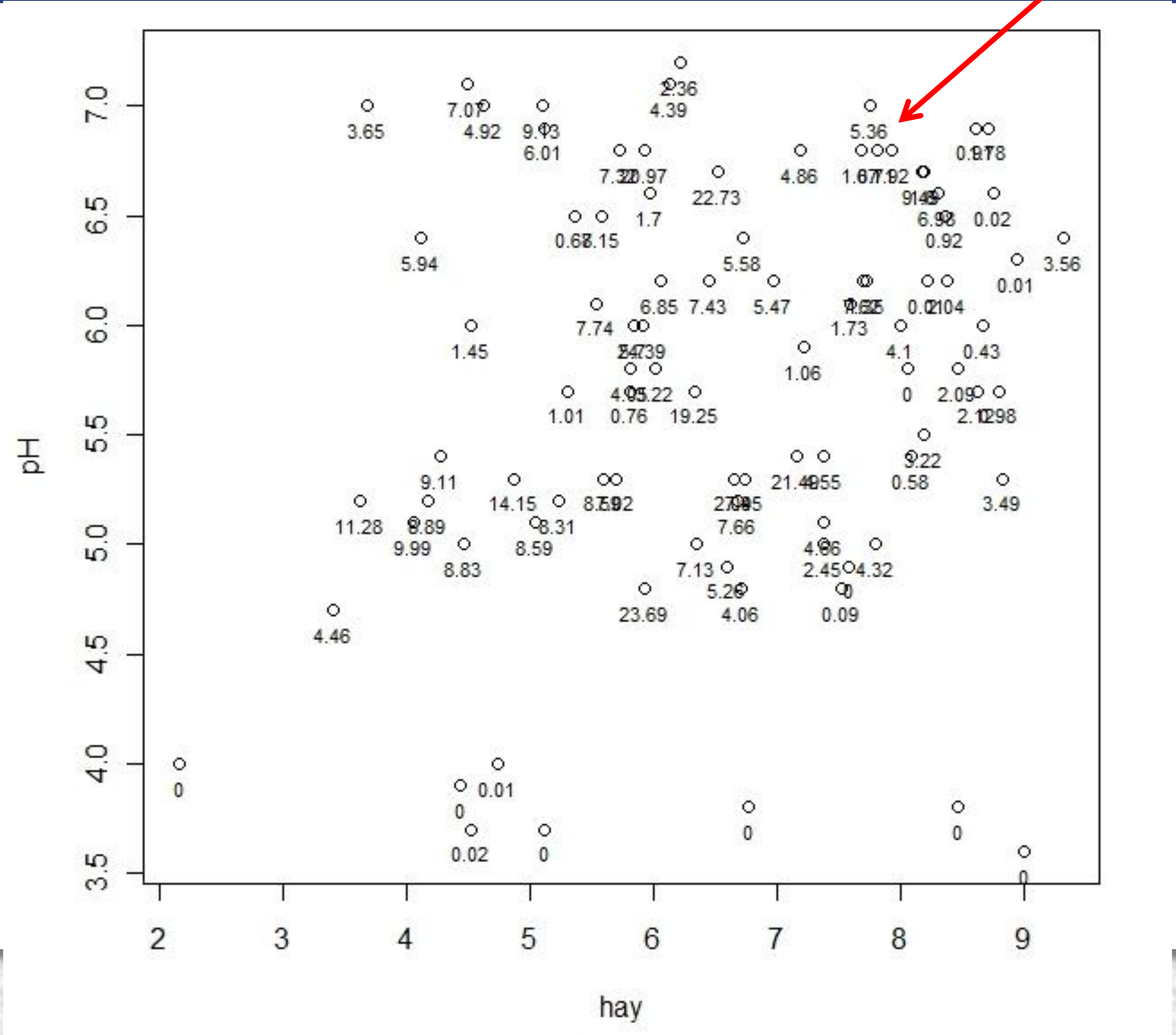
Label Scatterplot With Third Variable

- Use **`text()`** function to label each of the points:

```
> data <- read.table("http://www.bio.ic.ac.uk/research/mjcraw/
+ therbook/data/pgr.txt",header=T)
> attach(data)
> names(data)
# [1] "FR"  "hay" "pH"
> plot(hay,pH)
> text(hay,pH,labels=round(FR,2),pos=1,offset=0.5,cex=0.7)
```

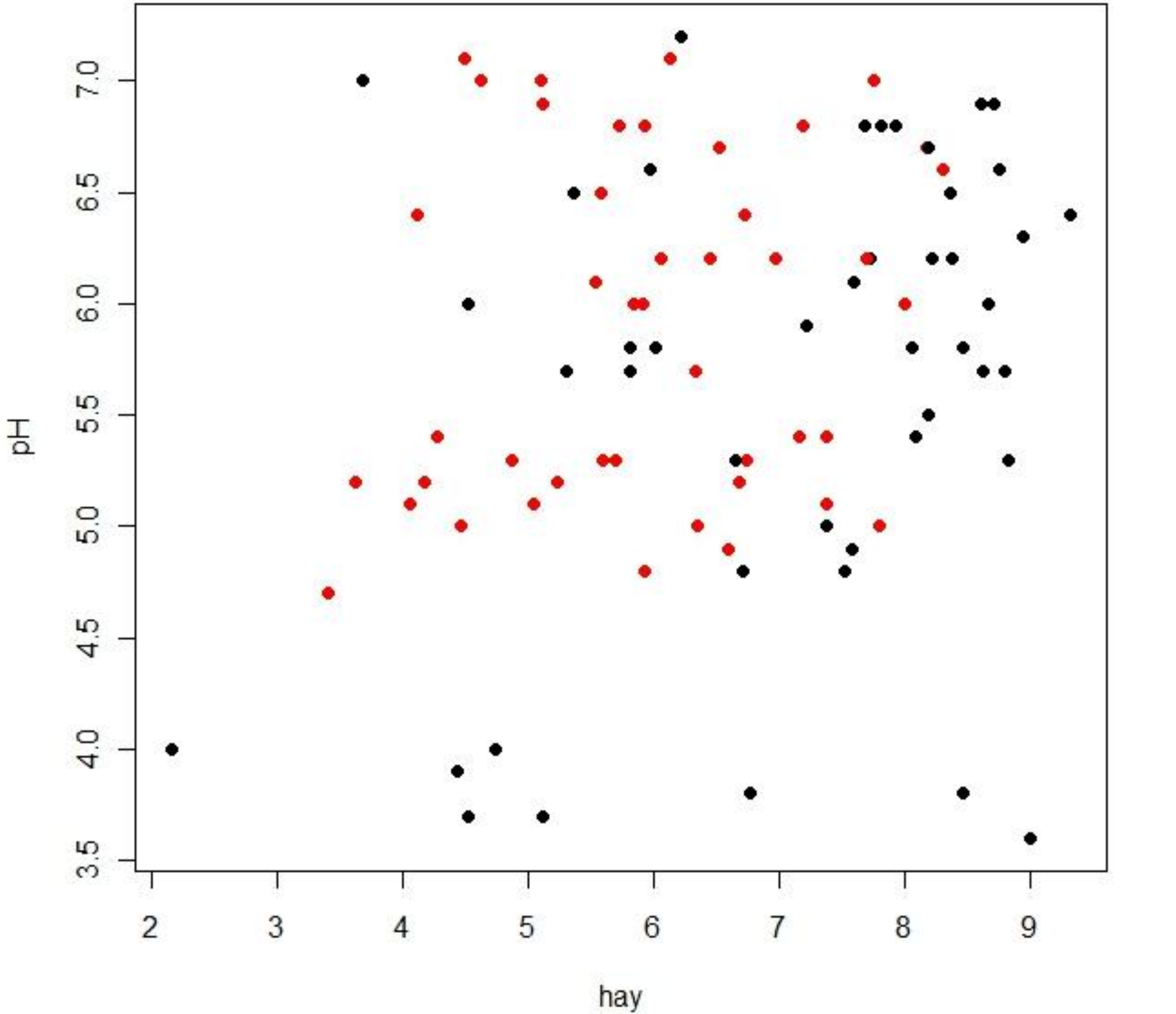
Label Scatterplot With Third Variable

The labels are *centered* on the x value of the point (pos=1) and are *offset half a character below* the point (offset=0.5).



Color Scatterplot With Third Variable

```
plot(hay,pH,pch=16,col=ifelse(FR>median(FR),"red","black"))
```

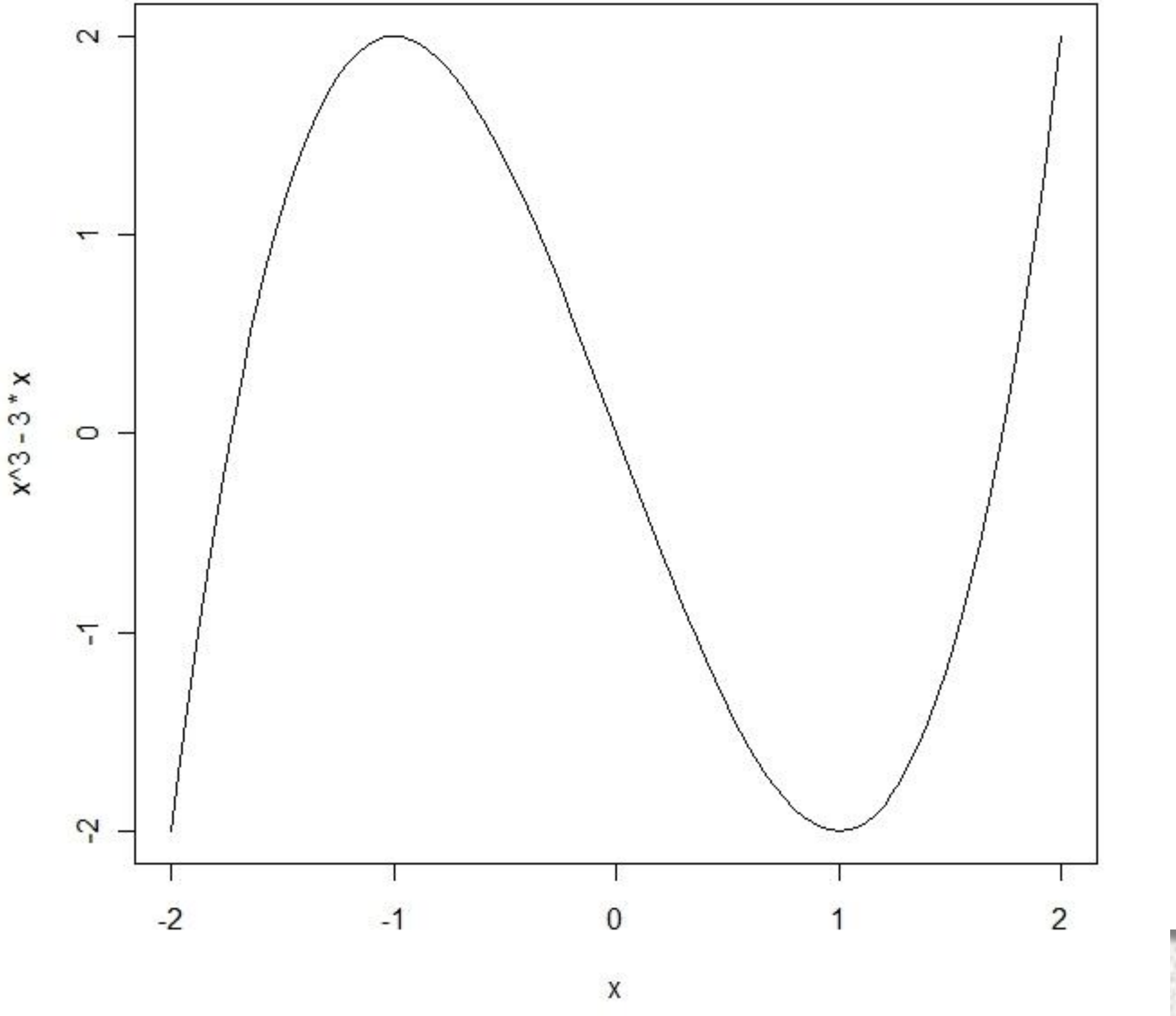


Drawing Mathematical Functions

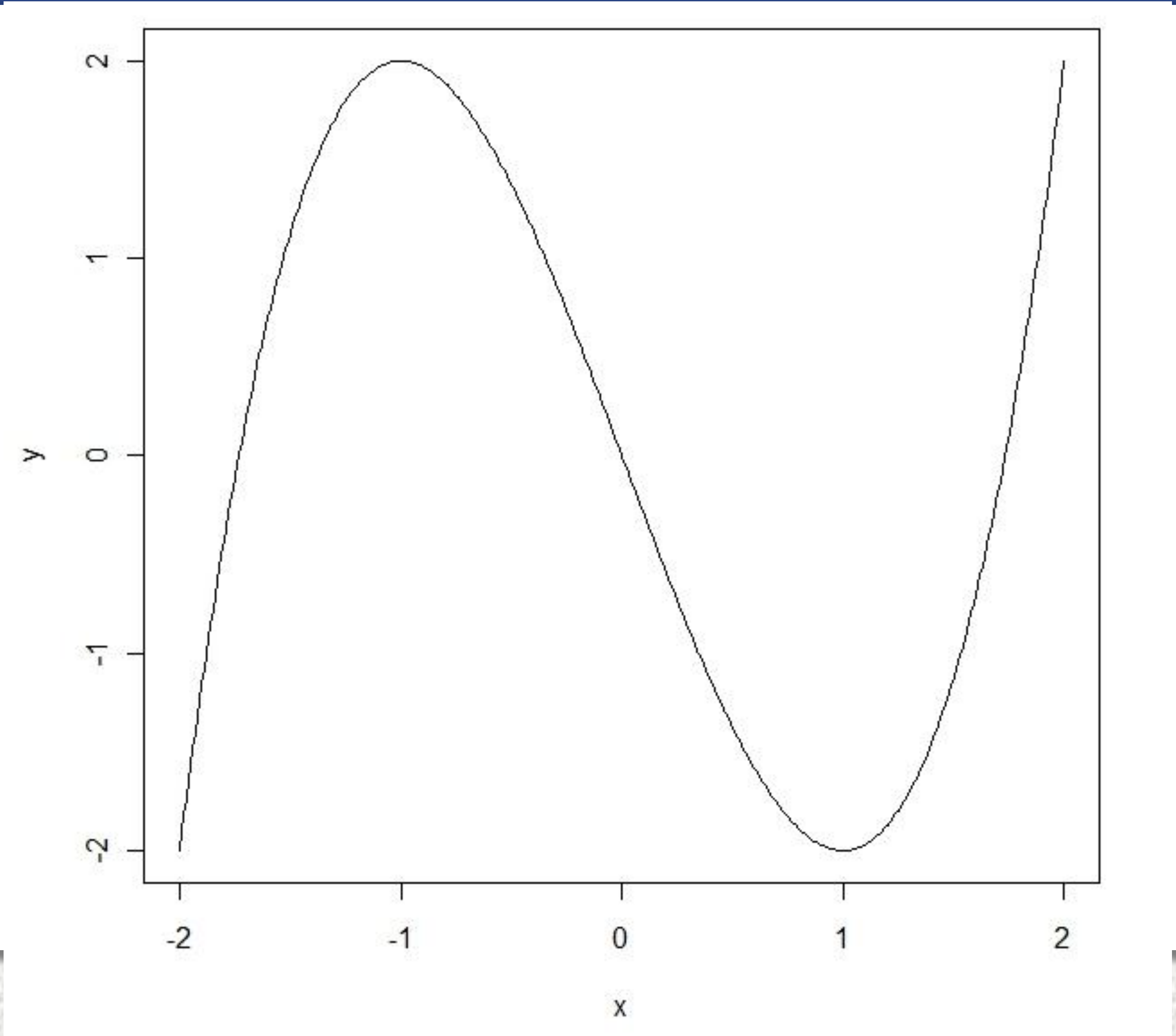
- Here we use the **`curve()`** function to plot x^3-3x between $x = -2$ and $x = 2$:

```
> curve(x^3-3*x,-2,2)
# More difficult approach using plot():
> x <- seq(-2,2,0.01)
> y <- x^3-3*x
> plot(x,y,type="l")
```

Drawing Mathematical Functions



Drawing Mathematical Functions

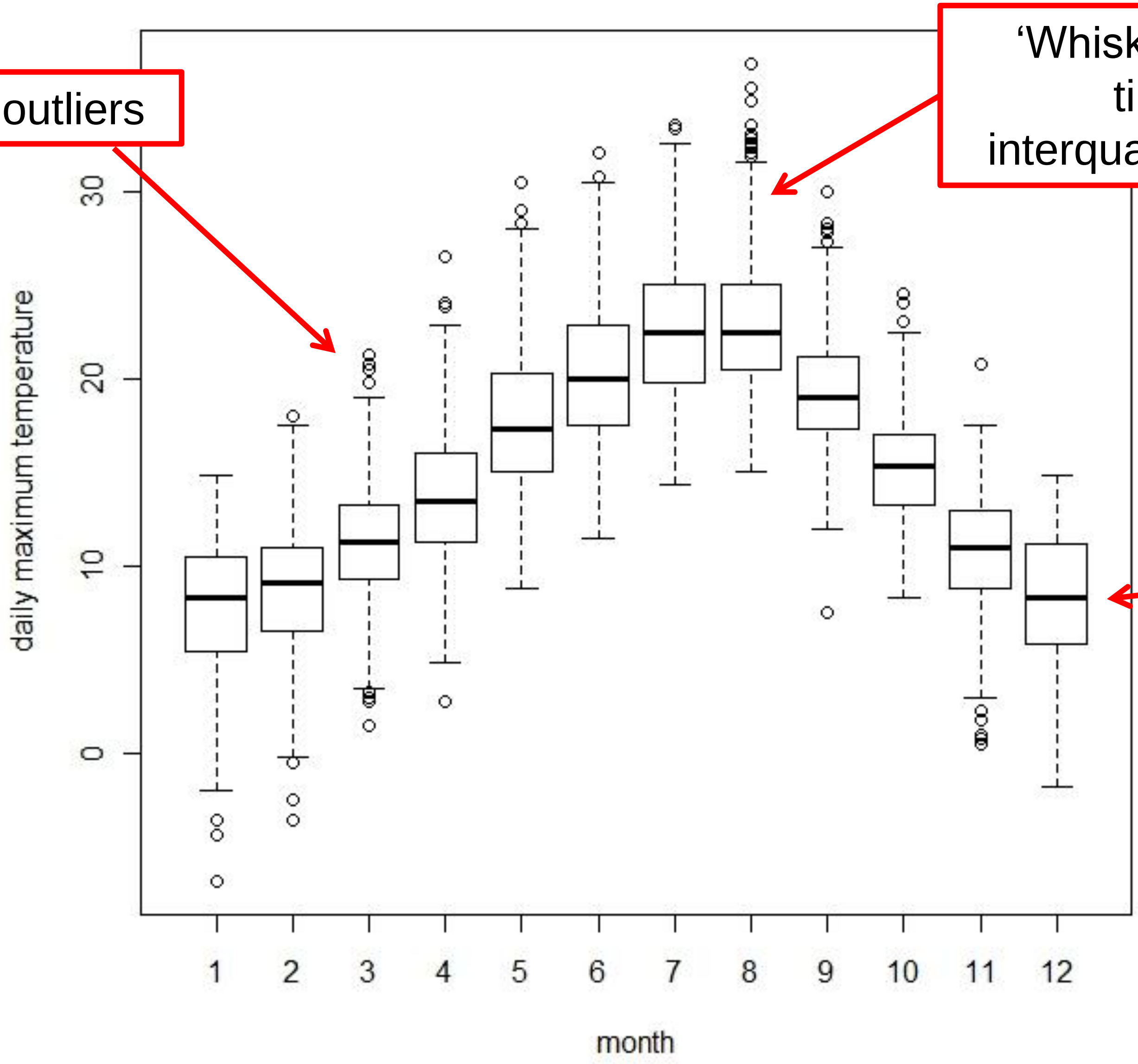


Plotting With A Categorical IV

- When the explanatory variable is categorical, we choose between a **`barplot()`** and a **`boxplot()`** :

```
> weather <- read.table("http://www.bio.ic.ac.uk/research/
+ mjcraw/therbook/data/SilwoodWeather.txt",header=T)
> attach(weather)
> names(weather)
# [1] "upper" "lower" "rain"  "month" "yr"
# Must declare month to be a factor (is numeric at this
point):
> month <- factor(month)
# Now we get a boxplot rather than a scatterplot:
> plot(month,upper,ylab="daily maximum
+ temperature",xlab="month")
```

Plotting With A Categorical IV



individual outliers

'Whiskers': 1 ½ times interquartile range

median

Plots With Multiple Variables

Plot Functions With Multiple Variables

- **`pairs()`** for a matrix of scatterplots of every variable against every other;
- **`coplot()`** for conditioning plots where *y* is plotted against *x* for different values of *z;*
- **`xyplot()`** where a set of panel plots is produced.

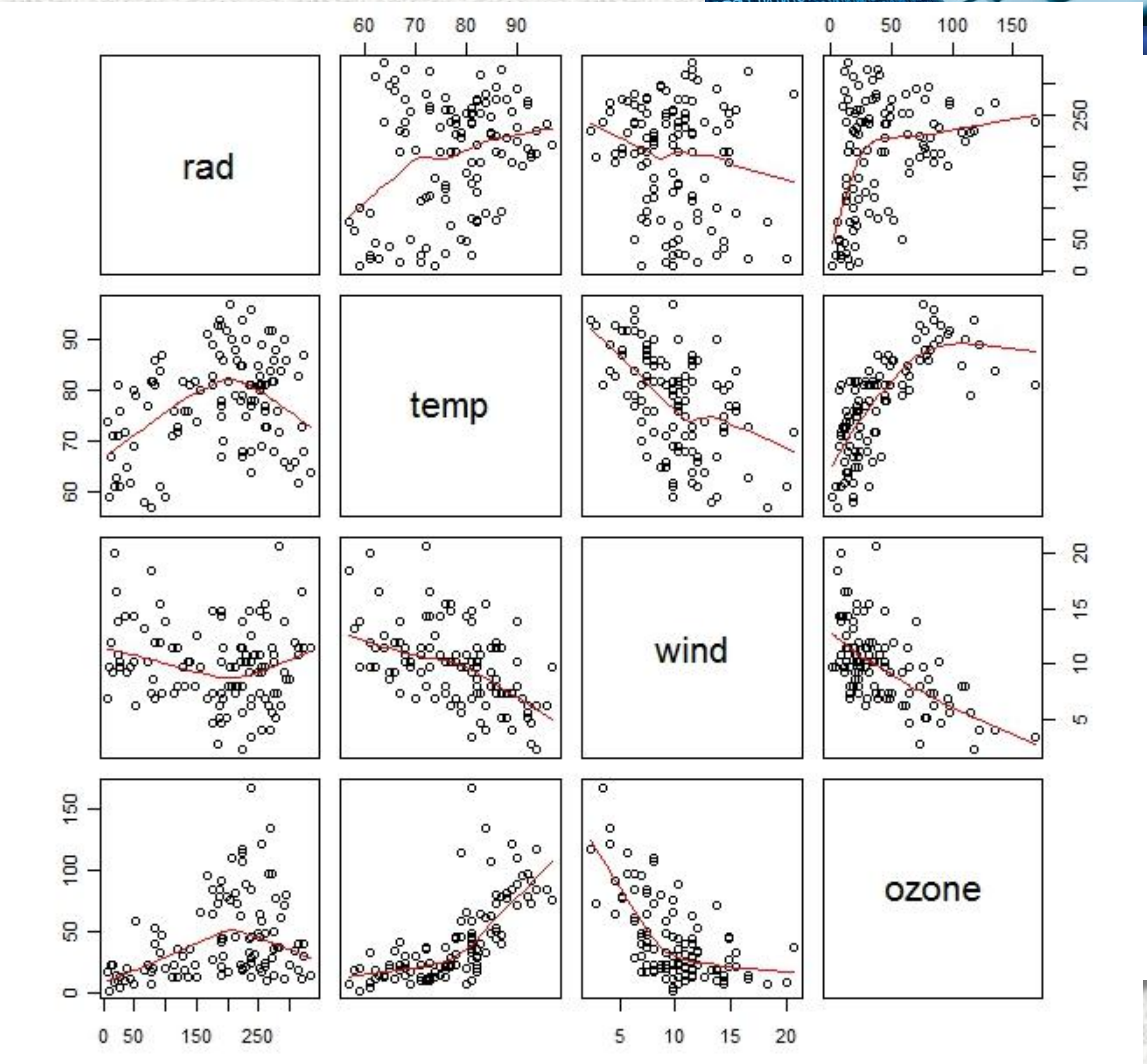
The `pairs()` Function

- The **`pairs()`** function plots every variable in the dataframe on the y axis against every other variable on the x axis:

```
> ozonedata <- read.table("c:\\temp\\ozone.data.txt",header=T)
> attach(ozonedata)
> names(ozonedata)
 [1] "rad"   "temp"  "wind"  "ozone"

> pairs(ozonedata,panel=panel.smooth)
```

The `pairs()` Function

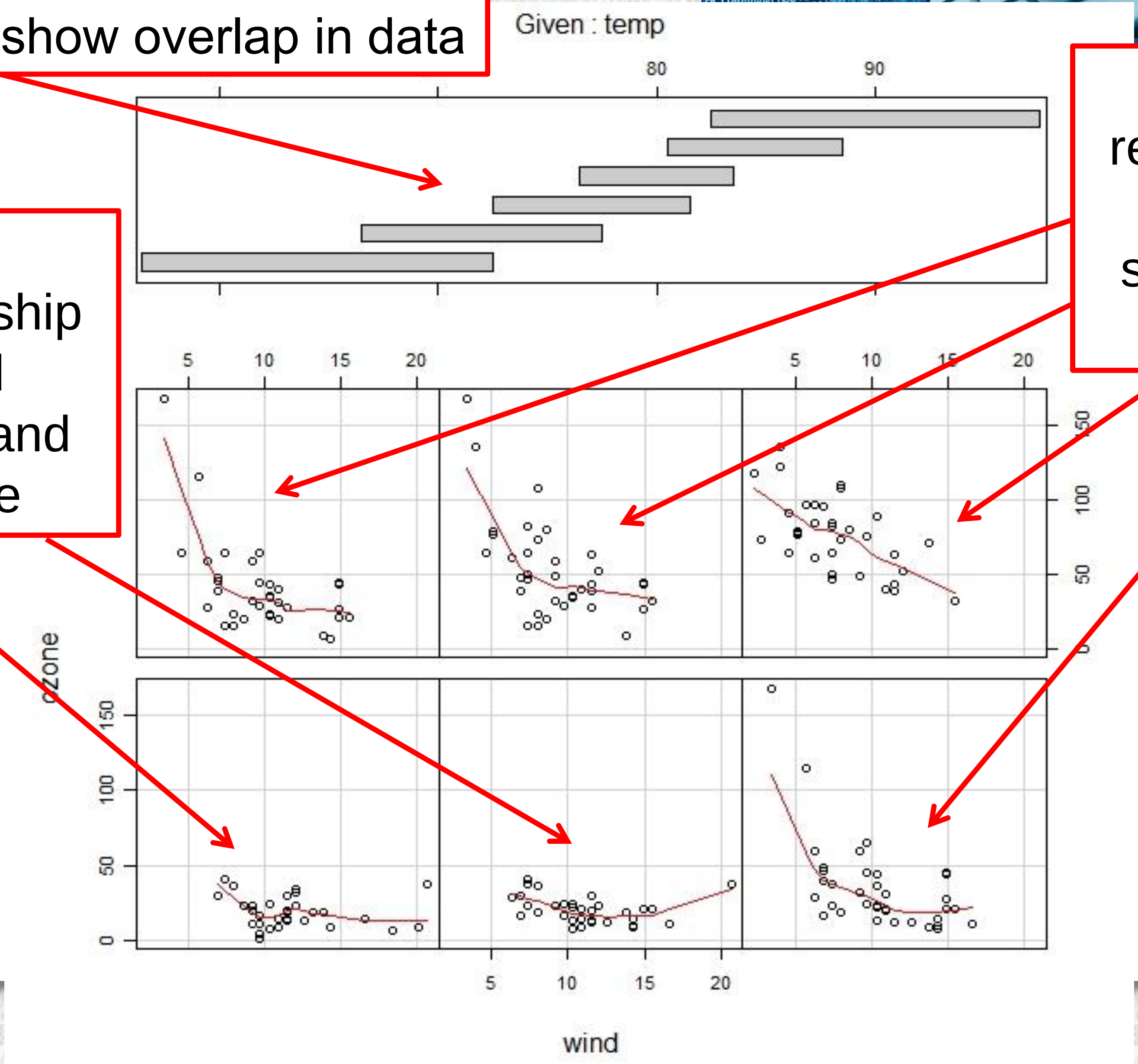


The `coplot()` Function

- A problem with multivariate data is that the relationship between two variables may be obscured by the effects of other processes:

```
> coplot(ozone~wind |temp,panel=panel.smooth)
```

The `coplot()` Function



‘shingles’ show overlap in data

No relationship wind speed and ozone

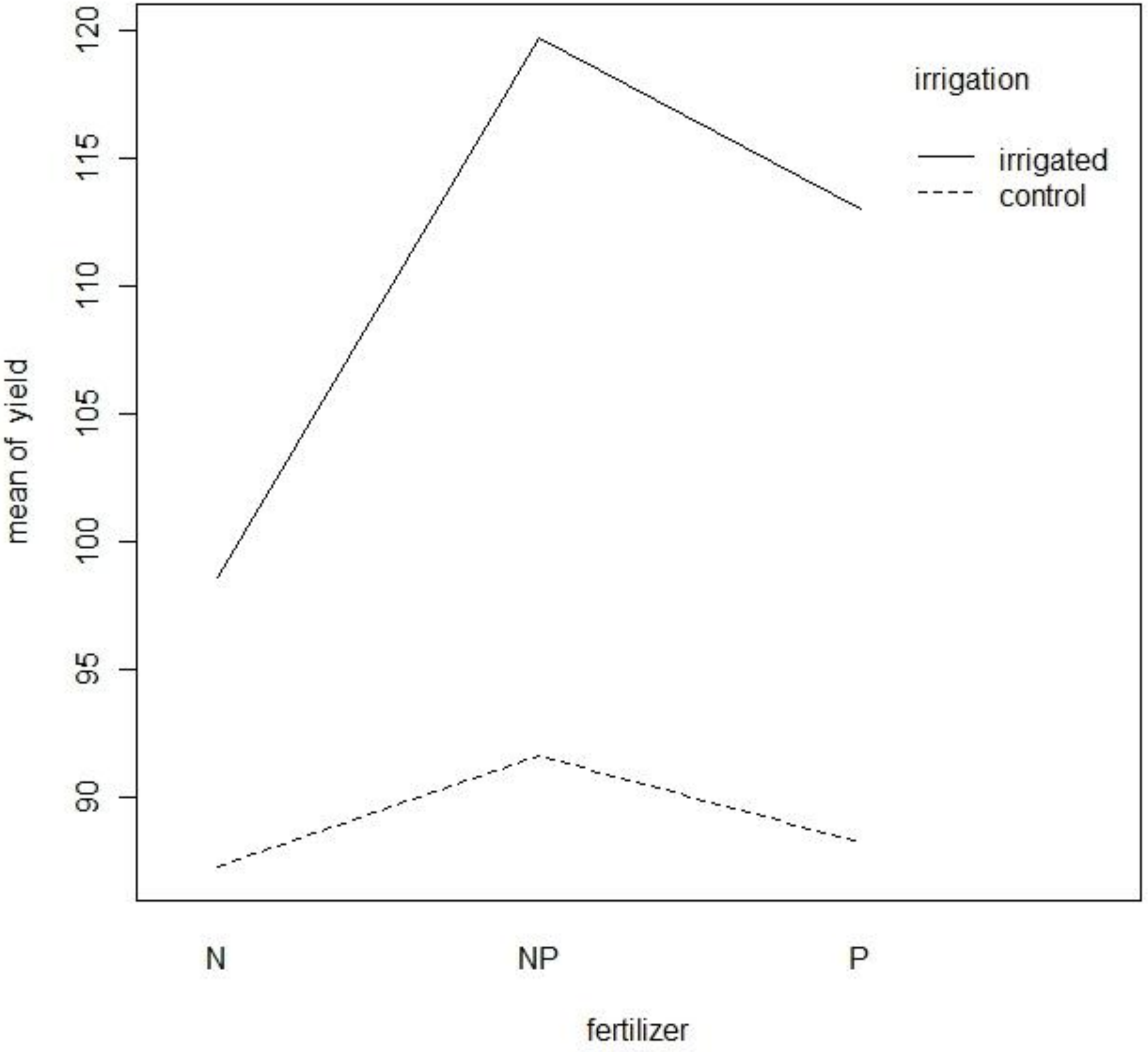
Negative relationship wind speed and ozone

Interaction Plots

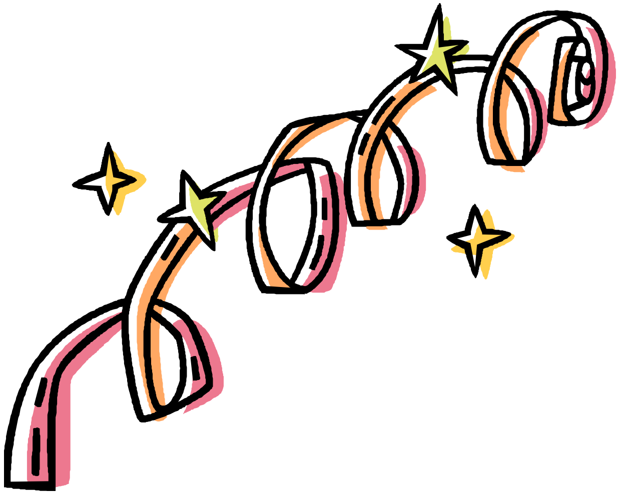
- Useful when the response to one factor depends upon the level of another factor.
- Particularly effective graphical means of interpreting the results of factorial experiments.
- Here is an experiment with grain yields in response to irrigation and fertilizer application

```
> yields <- read.table("c:\\temp\\splityield.txt",header=T)
> attach(yields)
> names(yields)
[1] "yield" “block" "irrigation" "density" "fertilizer"
> interaction.plot(fertilizer,irrigation,yield)
```

Interaction Plots



Special Plots



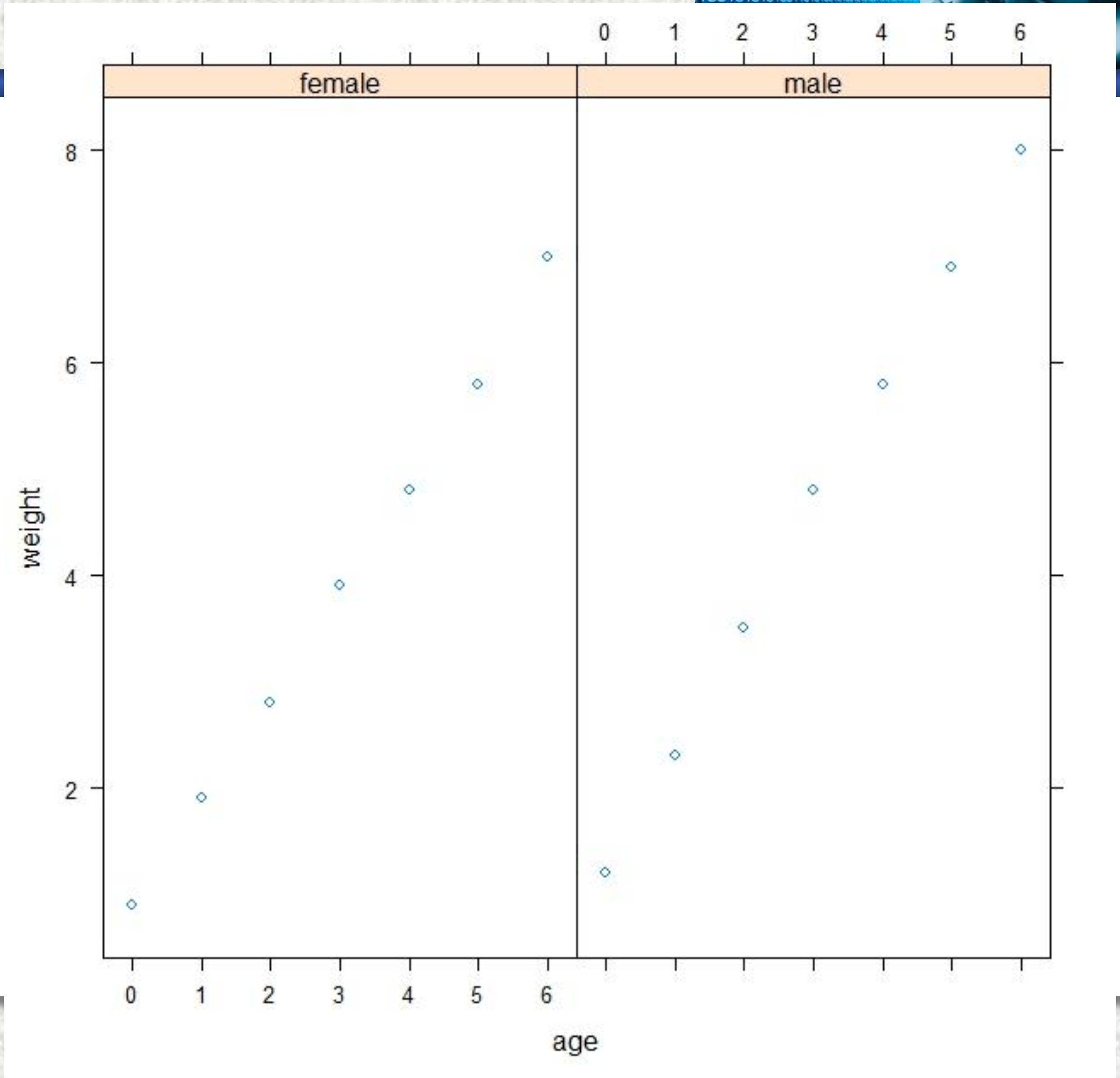
Trellis Graphics

- **Trellis plot** of weight against age by gender:

```
> data <- read.table("c:\\temp\\panels.txt",header=T)
> attach(data)
> names(data)
[1] "age"    "weight" "gender"
> library(lattice)
> xyplot(weight ~ age | gender)
```

Trellis Graphics: `xyplot()`

```
xyplot(weight ~ age | gender
```



High Level Trellis Functions

- **`barchart()`** for barplots
- **`bwplot()`** for box-and-whisker plots
- **`densityplot()`** for kernel density plots
- **`dotplot()`** for dot plots
- **`histogram()`** for panels of histograms
- **`qqmath()`** for quantile plots against mathematical distributions
- **`stripplot()`** for a one-dimensional scatterplot
- **`qq()`** for a QQ plot for comparing two distributions
- **`xyplot()`** for a scatterplot

High Level Trellis Functions

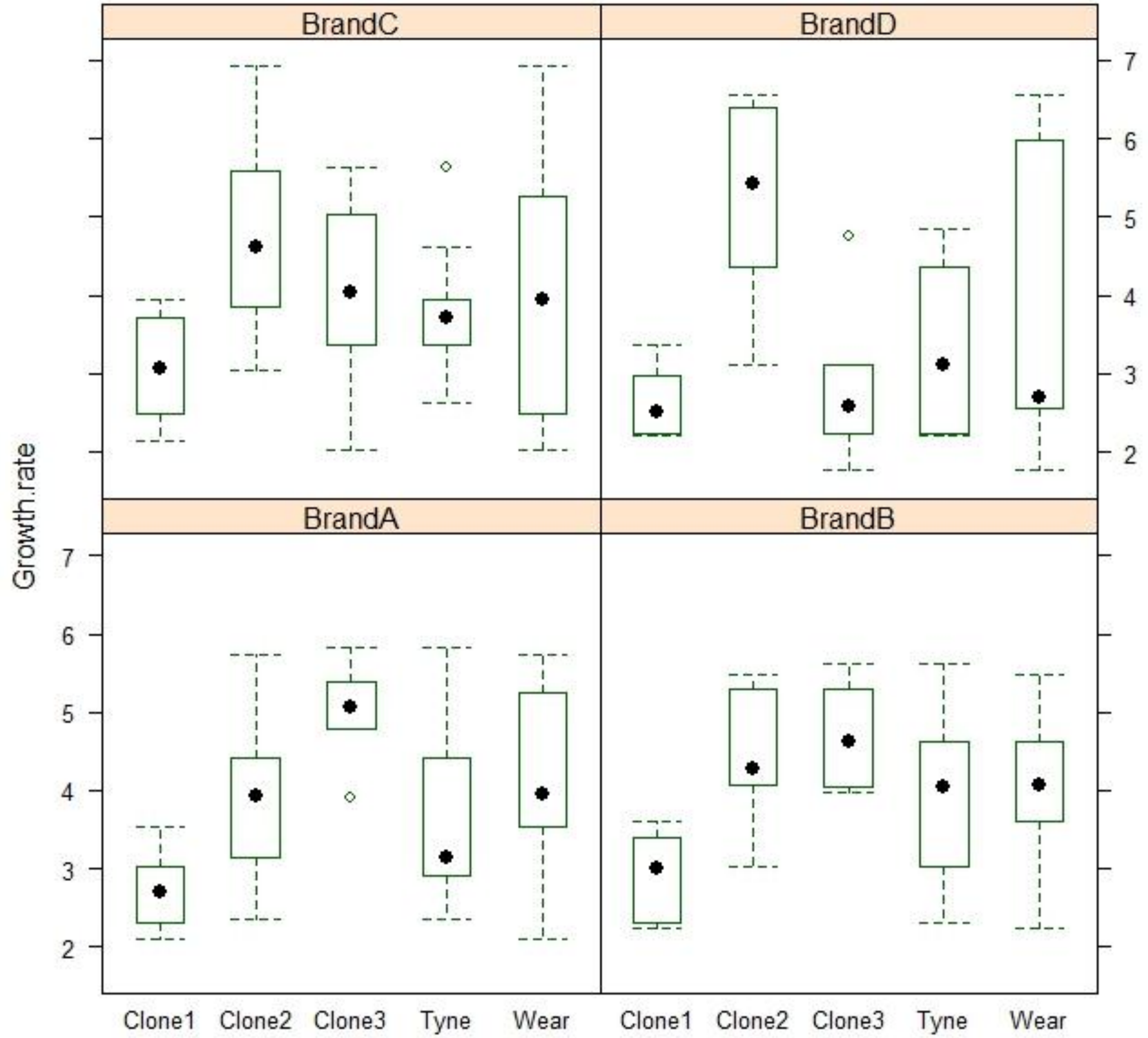
- **`levelplot()`** for creating level plots
- **`contourplot()`** for contour plots
- **`cloud()`** for three-dimensional scatterplots
- **`wireframe()`** for 3D surfaces (similar to **`persp`** plots)
- **`splom()`** for a scatterplot matrix
- **`parallel()`** for creating parallel coordinate plots
- **`rfs()`** to produce a residual and fitted value plot
- **`tmd()`** for a Tukey mean-difference plot

The `bwplot()` for Designed Experiment

- **Trellis plot** to interpret designed experiment where all explanatory variables are categorical:

```
> data <- read.table("c:\\temp\\daphnia.txt",header=T")
> attach(data)
> names(data)
[1] "Growth.rate" "Water"  "Detergent"   "Daphnia"
> library(lattice)
> trellis.par.set(col.whitebg())
> bwplot(Growth.rate ~ Water+Daphnia|Detergent)
```

The `bwplot()` for Designed Experiment



Design Plot

- An effective way of visualizing effect sizes in designed experiments is the **`plot.design()`** function which is used just like a model formula:

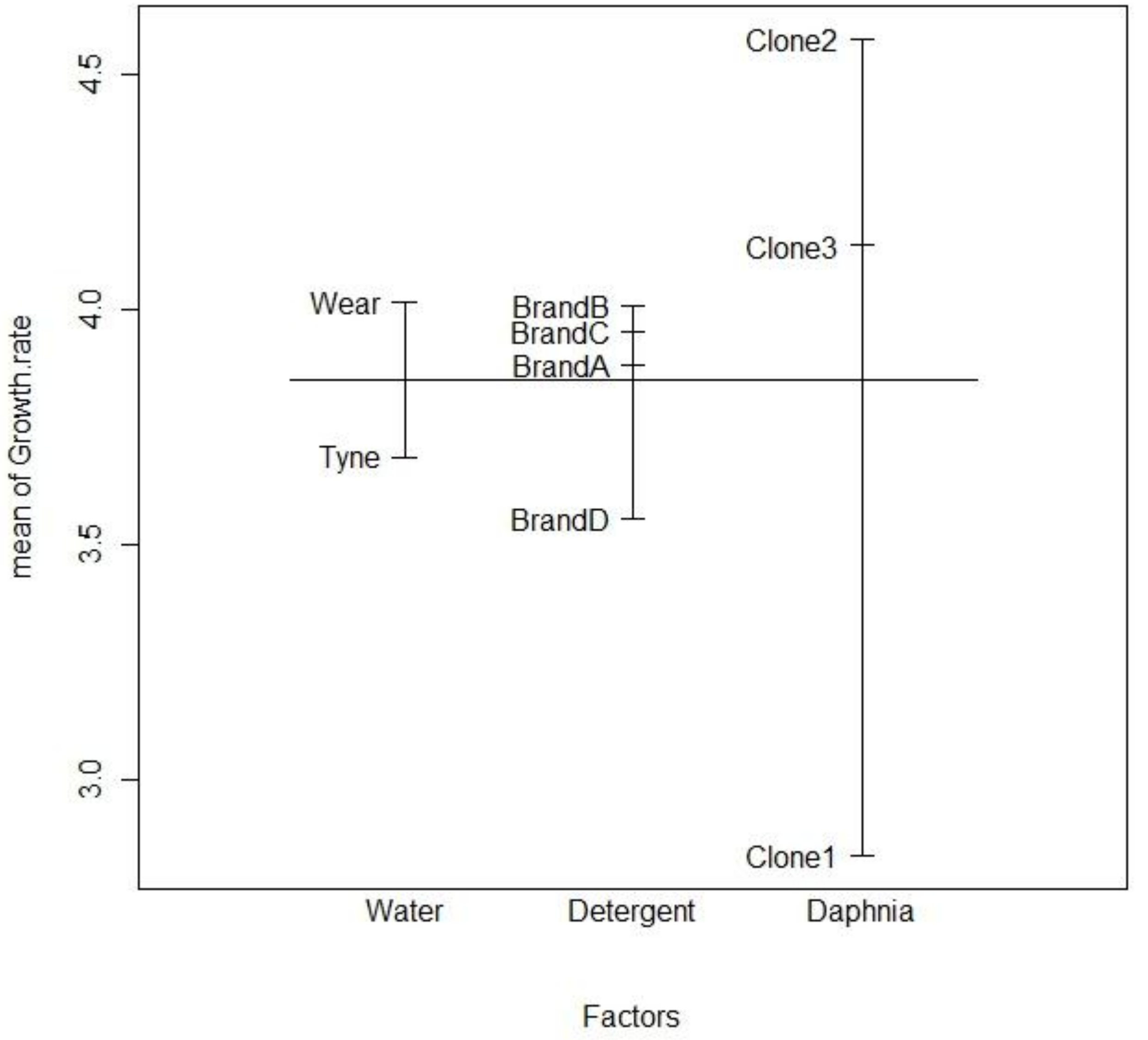
```
> plot.design(Growth.rate~Water*Detergent*Daphnia)
```

Shows the main effect of three factors, drawing attention to the major differences between the detergent brands A, B and C. The default is to plot means, but other functions can be called, such as median, var or sd.

```
> plot.design(Growth.rate~Water*Detergent*Daphnia,fun=“sd”)
```

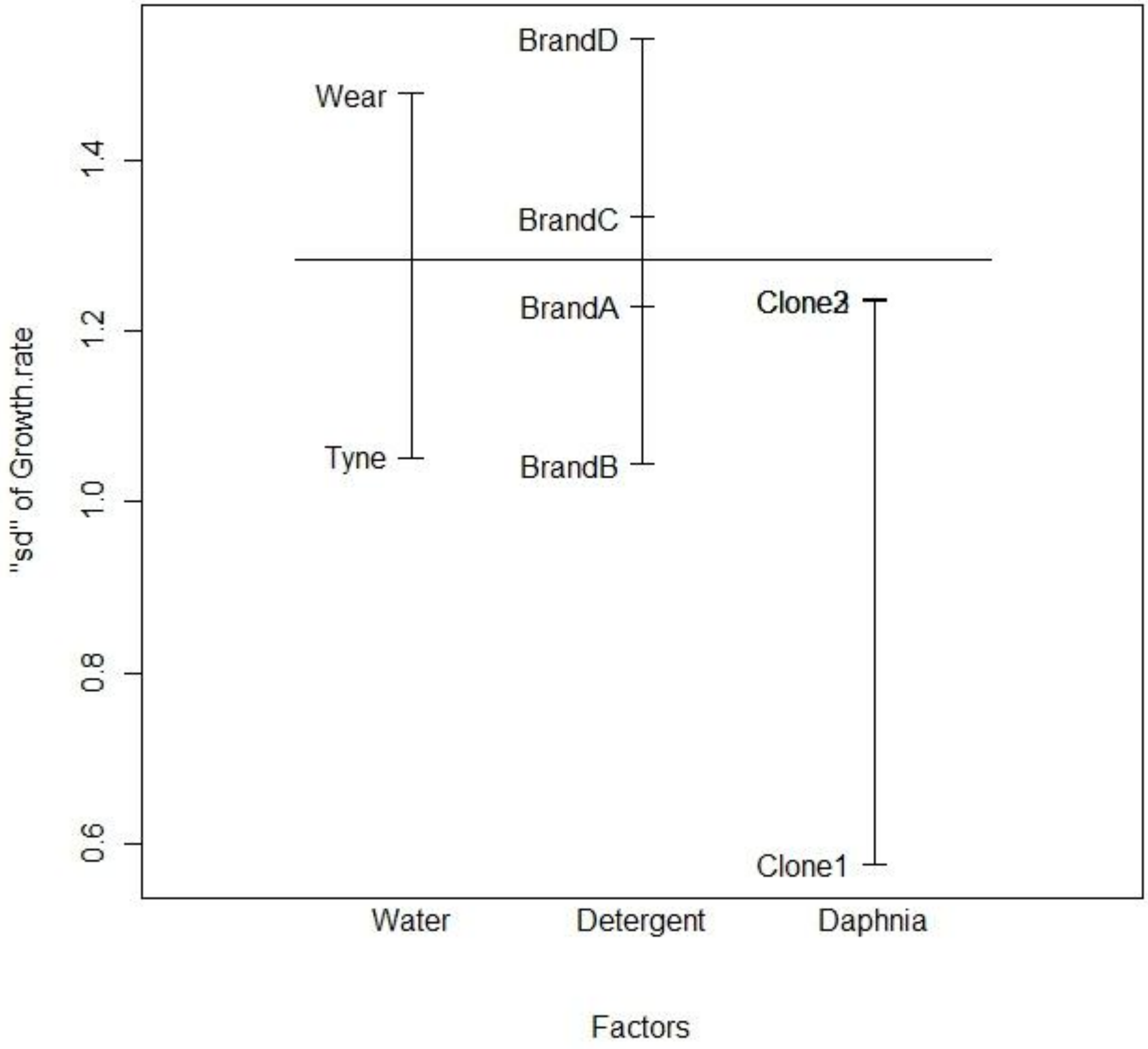
Design Plots

```
> plot.design(Growth.rate~Water*Detergent*Daphnia)
```



Design Plots

```
> plot.design(Growth.rate~Water*Detergent*Daphnia,fun="sd")
```

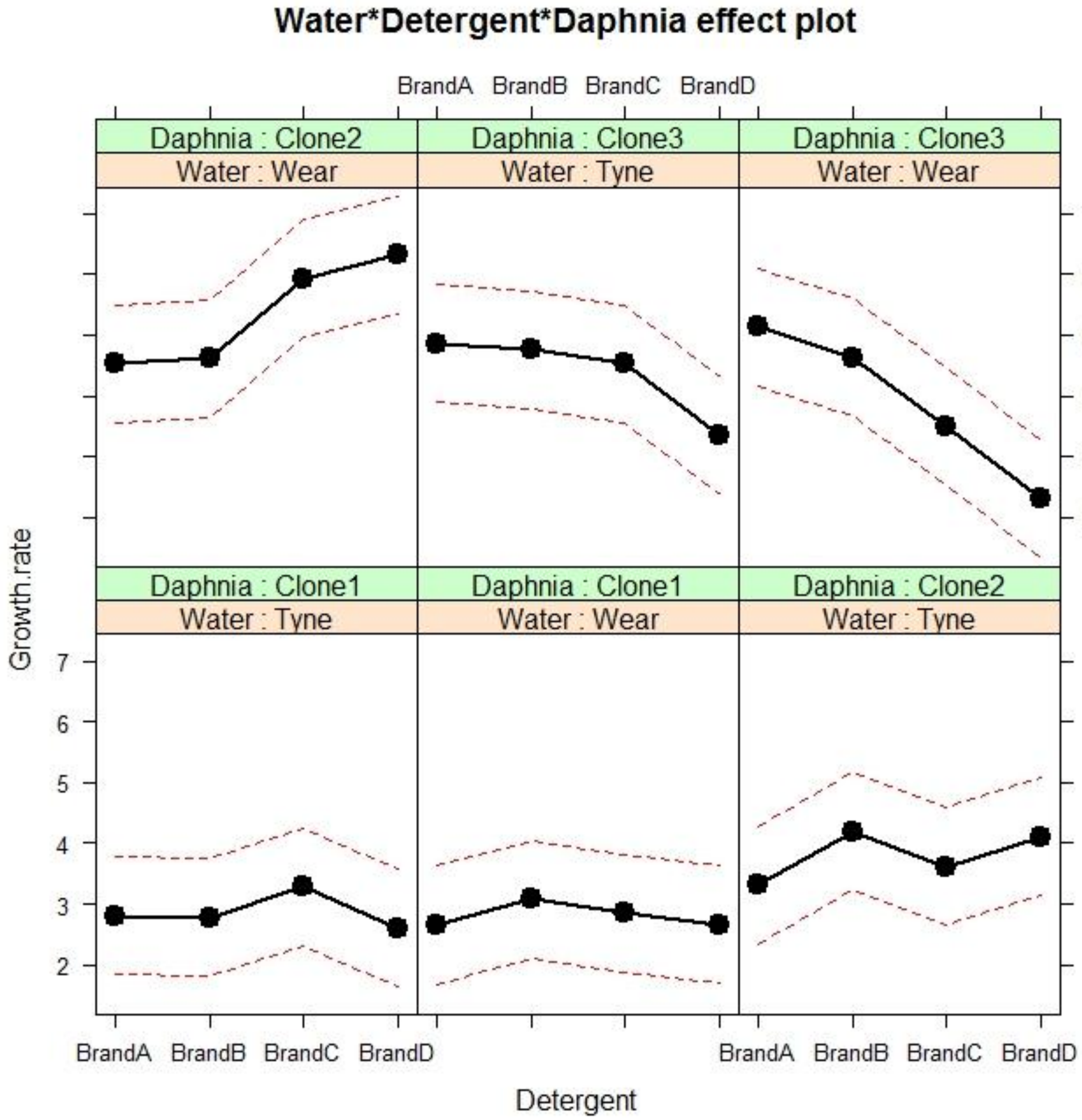


Effect Sizes

- An alternative is to use the **`effects`** package which takes a model object (a linear model or a generalized linear model) and provides trellis plots of specified effects:

```
> install.packages("effects")
> library(effects)
> model <- lm(Growth.rate~Water*Detergent*Daphnia)
# First calculate all effects, then plot:
> daph.effects <- allEffects(model)
> plot(daph.effects,"Water:Detergent:Daphnia")
```

Effect Sizes



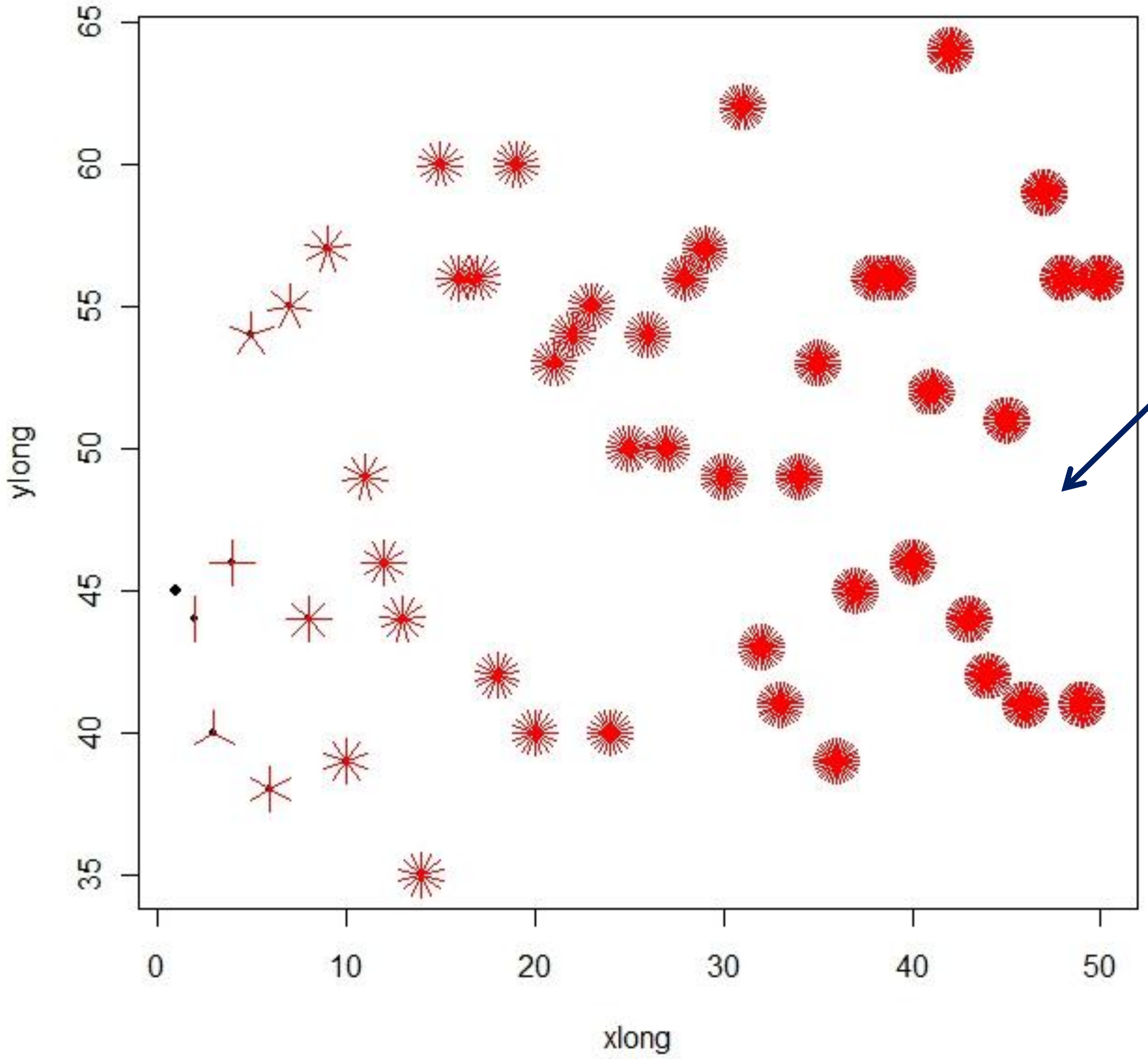
Plots with Identical Values

- Sometimes have (especially with count data) **two or more points** that fall in exactly the same location in a scatterplot, burying one repeated value beneath the other.
- **`sunflowerplot()`** function:

```
> numbers <- read.table("c:\\temp\\longdata.txt",header=T)
> attach(numbers)
> names(numbers)
[1] "xlong" "ylong"

> sunflowerplot(xlong,ylong)
```

Plots with Identical Values



Replication at each point increases as *x* increases