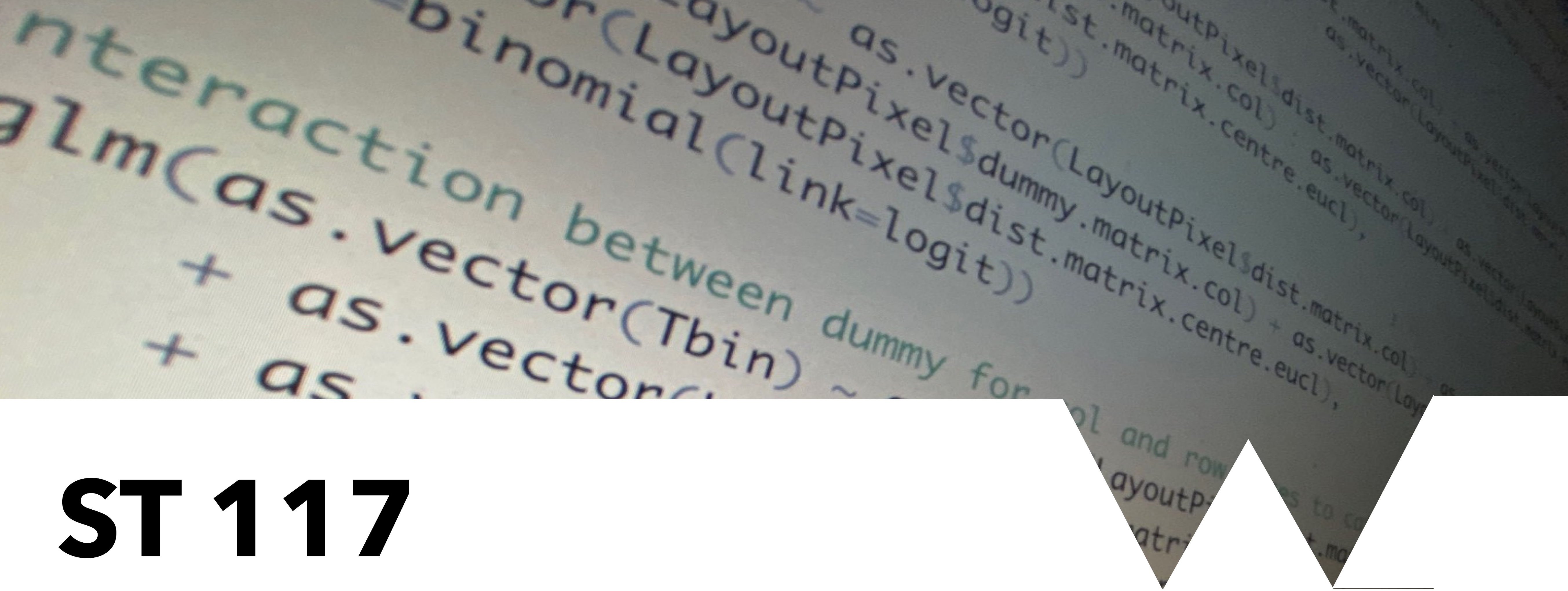


ST 117
2. Basic R

WARWICK

Lecture 5 & 6
(Week 2)

Sampling random variables
Data frames
Plotting
Basic R programming (functions and flow)

Sequences of random variables

General task: Let $X_1, X_2, X_3, \ldots, X_5$ be random variables on a probability space $(\Omega, \mathscr{A}, P)$. Assume they are independent and identically distributed (i.i.d.) with probability distribution P. Create realisations of $X = (X_1, X_2, X_3, \ldots, X_5)$.

P is the **uniform distribution on** $[0,1]$:

```
> X = runif(5)
> X
[1] 0.03184086 0.73297235 0.94056130 0.23931732 0.97944579
```

P is the **uniform distribution on** $[-2,2]$:

```
> X = runif(5,-2,2)
> X
[1] -1.6956848  1.9053933  0.6403827 -0.3197902 -1.3378172
```

Sequences of random variables

General task: Let $X_1, X_2, X_3, \ldots, X_5$ be random variables on a probability space $(\Omega, \mathscr{A}, P)$. Assume they are independent and identically distributed (i.i.d.) with probability distribution P. Create realisations of $X = (X_1, X_2, X_3, \ldots, X_5)$.

P is the **standard normal distribution**:

```
> X = rnorm(5)
> X
[1] -0.9889306  0.7816926 -0.7506576 -0.8720431  1.5440673
```

P is the **normal distribution with** $\mu = 8$, $\sigma = 2$:

```
> X = rnorm(5,8,2)
> X
[1] 9.366284 6.109285 7.450642 8.315501 7.251779
```

Sequences of random variables

General task: Let $X_1, X_2, X_3, \ldots, X_5$ be random variables on a probability space $(\Omega, \mathscr{A}, P)$. Assume they are independent and identically distributed (i.i.d.) with probability distribution P. Create realisations of $X = (X_1, X_2, X_3, \ldots, X_5)$.

P is the **binomial distribution** $n = 10$ **trials at success probability** $p = 0.5$:

```
> X = rbinom(5,10,0.5)
> X
[1] 5 3 5 4 7
```

Interpretation: For each X_i $(i = 1,2,\ldots,5)$ we toss 10 fair coins and count the number of successes.

P is the **binomial distribution** $n = 10$ **trials at success probability** $p = 0.1$:

```
> X = rbinom(5,10,0.1)
> X
[1] 0 0 1 0 2
```

P is the **binomial distribution** $n = 1000$ **trials at success probability** $p = 0.1$:

```
> X = rbinom(5,1000,0.1)
> X
[1]  91  99 107  99 100
```

Data frames

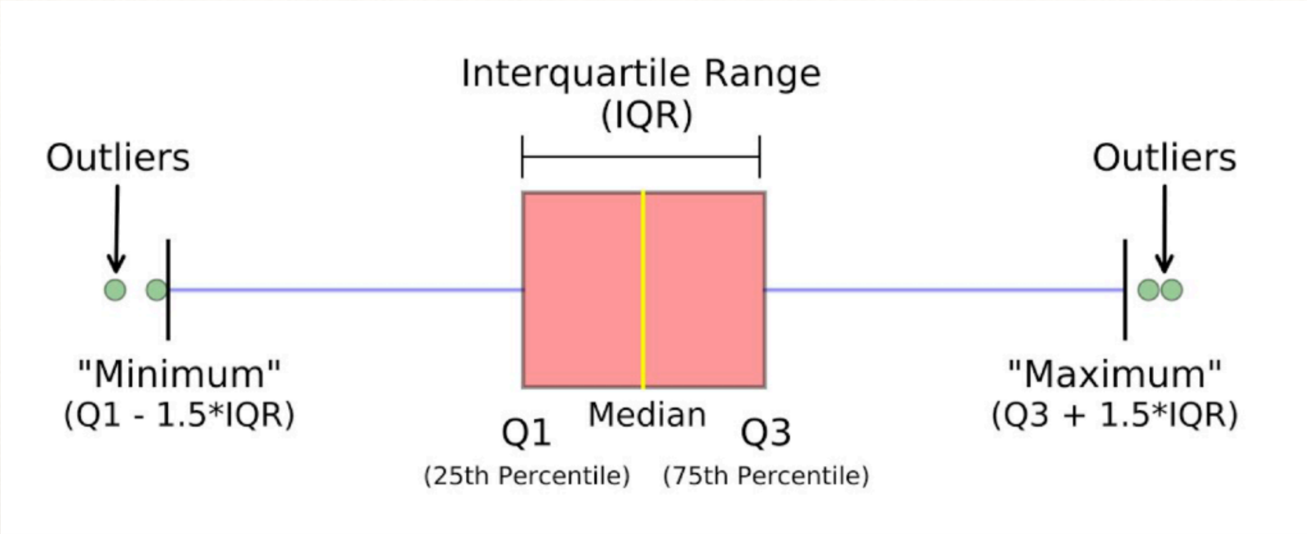
- Data frames are lists, but like arrays all items are the same length.
- Each (named) column is a statistical variable.
- Each row is a single subject. Different variables may have different types: integer, floating point, Boolean, factor, string.
- Individual variables are isolated using `$` notation: `dataframe$variable` is a vector.

```
> D <- data.frame(weight=rnorm(5,2,1),
                  height=rnorm(5,10,3),
                  colour=c("red", "blue", "blue", "red", "blue"))
> D$weight
[1] 2.3188167 0.9109947 2.1228736 2.7731489 0.9810314
```

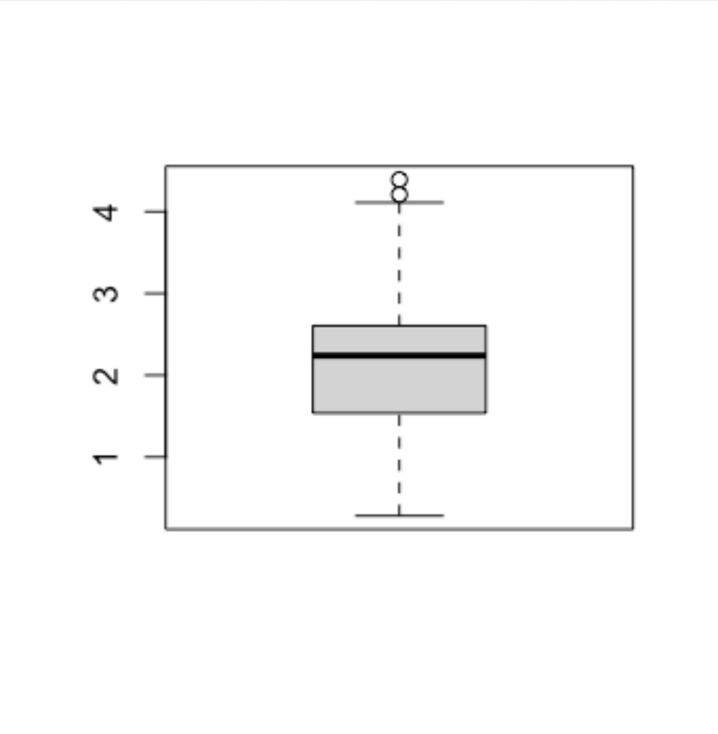
```
> D
     weight    height colour
1 2.3188167 10.257222    red
2 0.9109947  8.760244   blue
3 2.1228736 12.650058   blue
4 2.7731489  9.064683    red
5 0.9810314 10.473973   blue
```

Data visualisation: boxplot

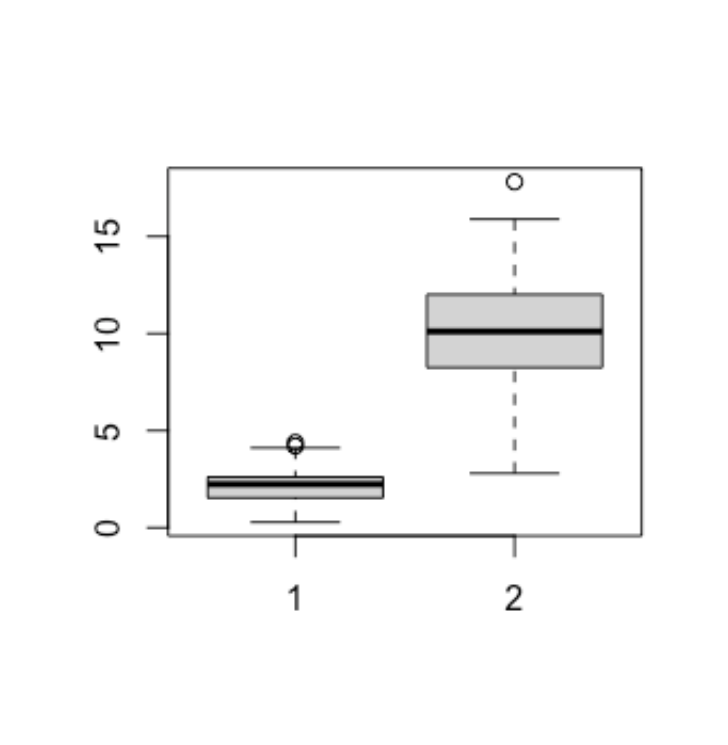
```
D <- data.frame(weight=rnorm(100,2,1),
                height=rnorm(100,10,3))
```




```
boxplot(D$weight)
```



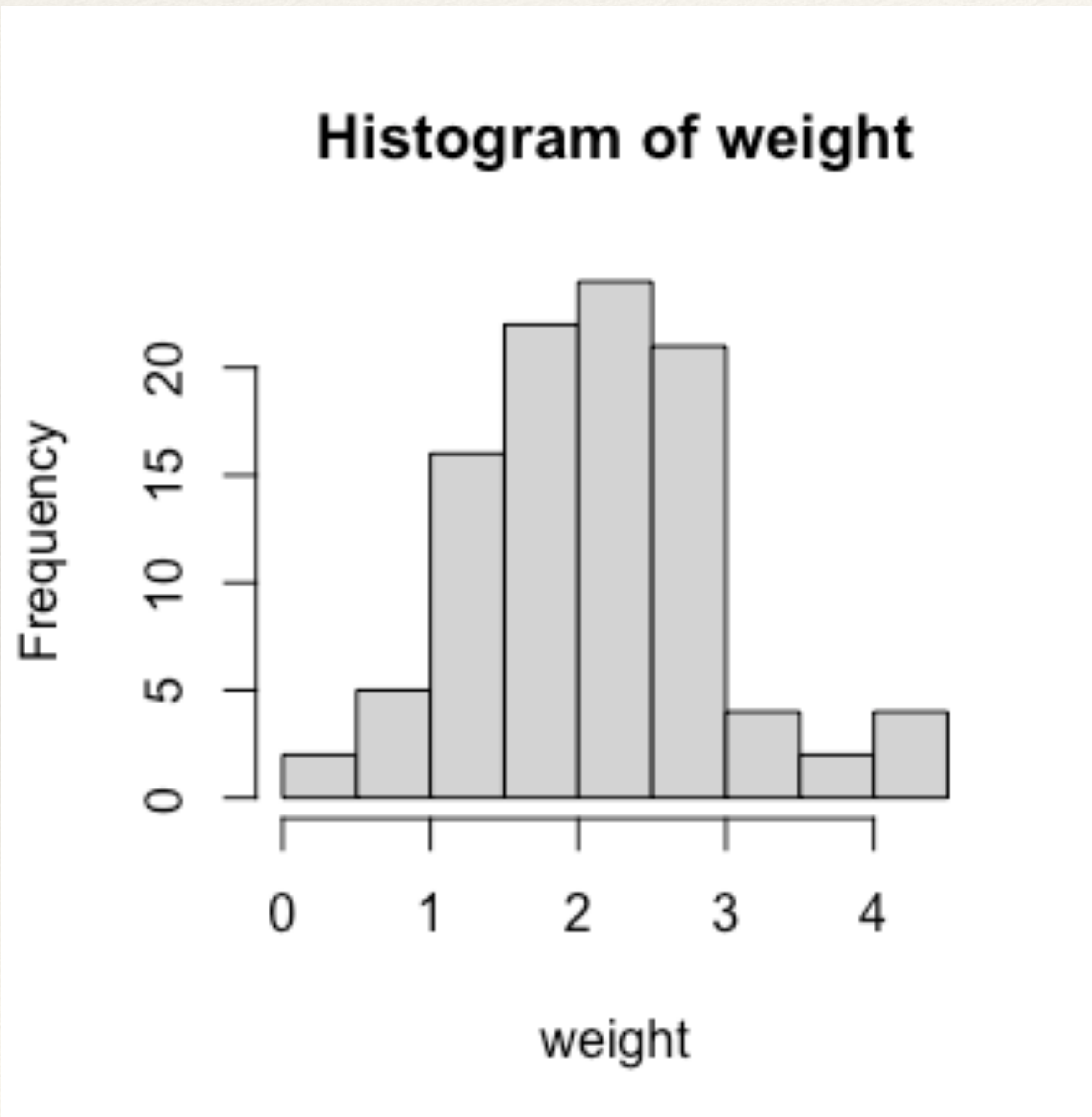

```
boxplot(D$weight,D$height)
```

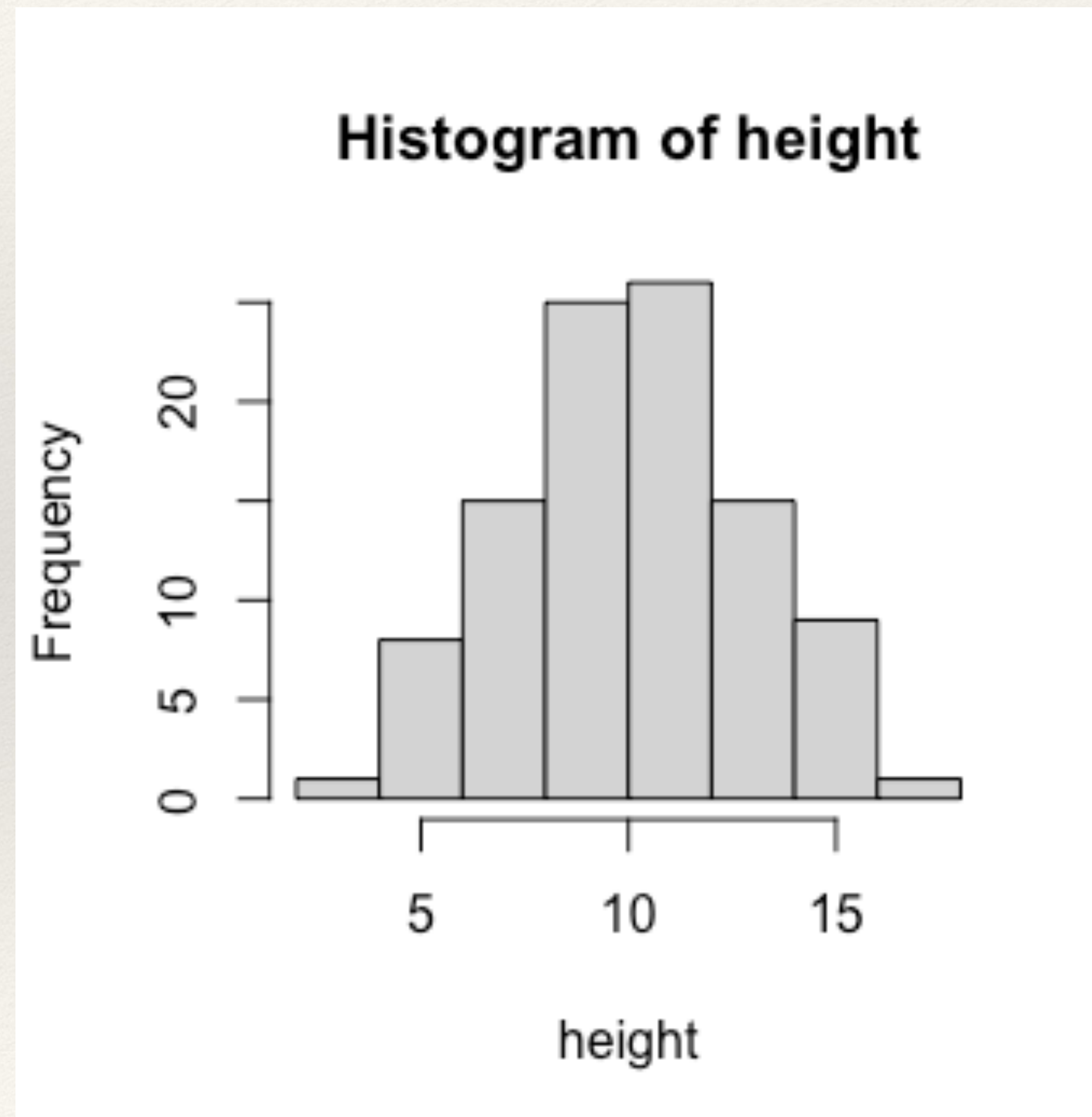


Data visualisation: histogram

```
hist(D$weight, xlab="weight", main="Histogram of weight")
```

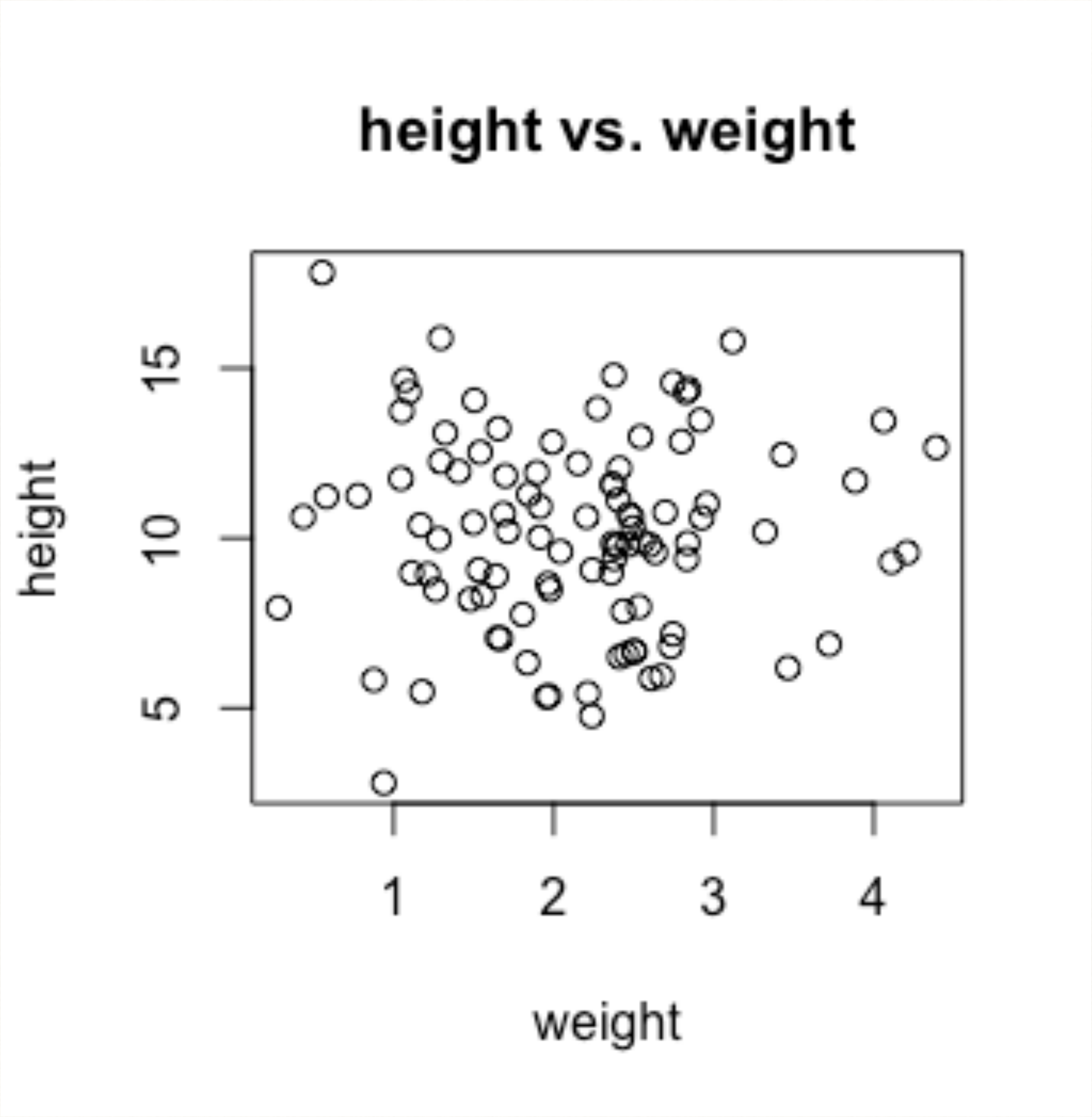
```
hist(D$height, xlab="height", main="Histogram of height")
```





Data visualisation: scatterplot

```
plot(D$weight, D$height,
     main="height vs. weight", xlab="weight", ylab="height")
```

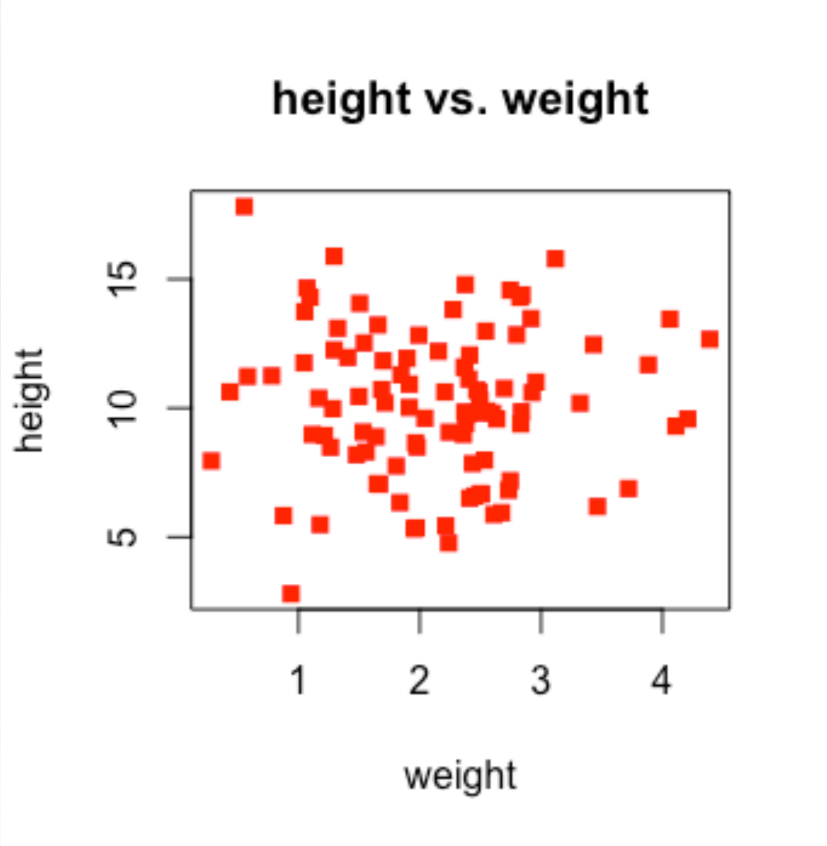


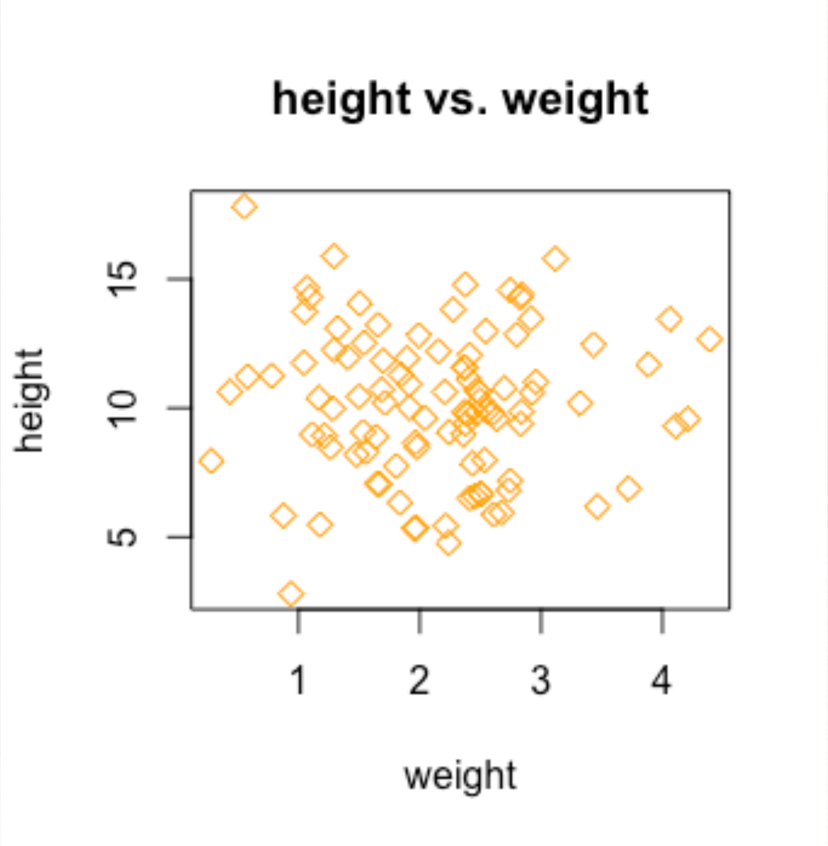
Data visualisation: scatterplot

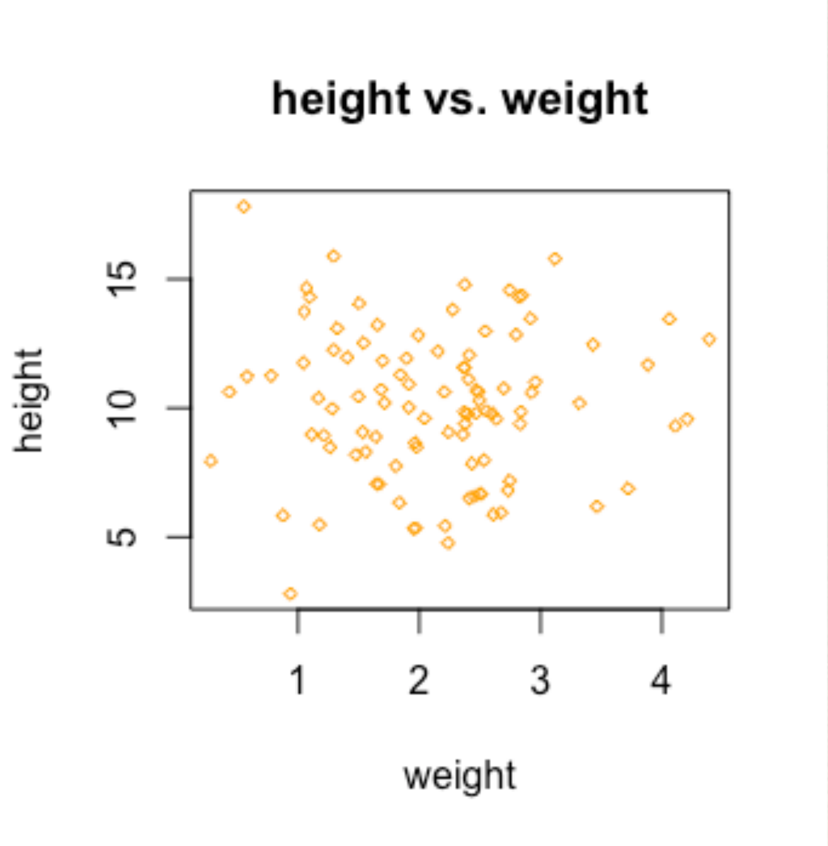
```
plot(D$weight, D$height, main="height vs. weight", xlab="weight",
ylab="height", pch=15, col="red")
```

```
plot(D$weight, D$height, main="height vs. weight", xlab="weight",
ylab="height", pch=5, col="orange")
```

```
plot(D$weight, D$height, main="height vs. weight", xlab="weight",
ylab="height", pch=5, col="orange", cex=0.5)
```



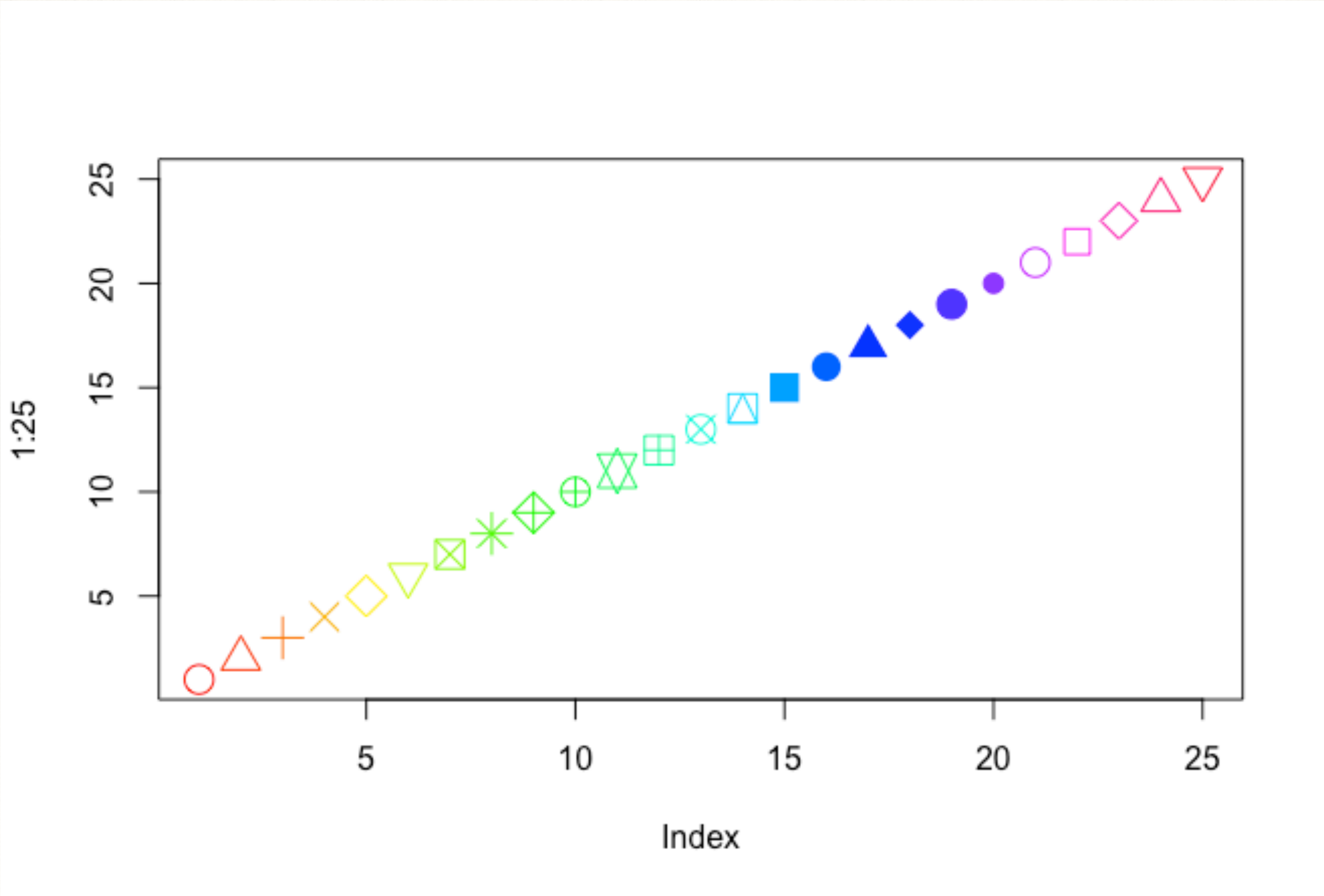




Data visualisation: scatterplot

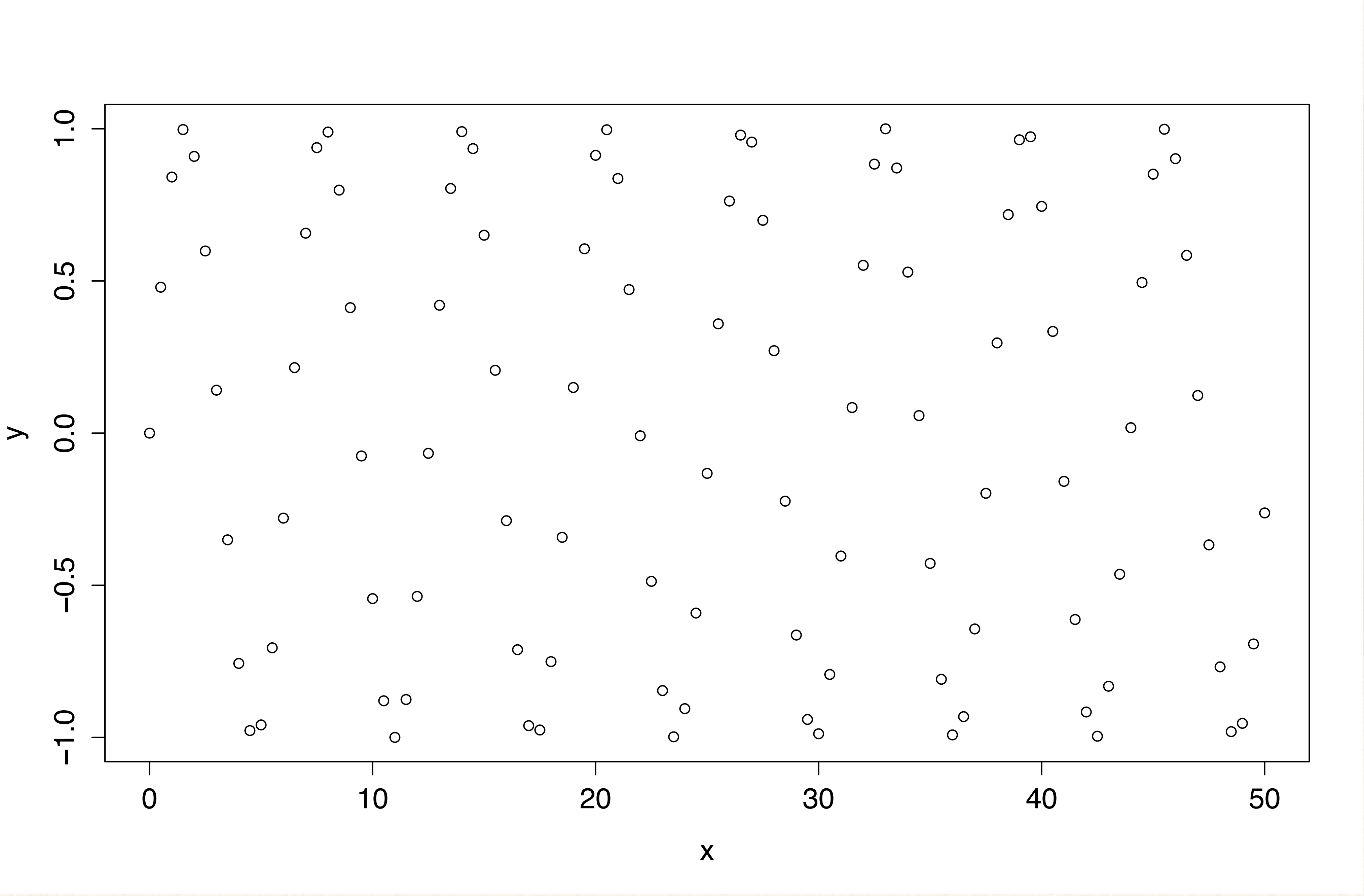
Plotting characters overview

```
plot(1:25, pch=1:25, col=rainbow(25), cex=3)
```

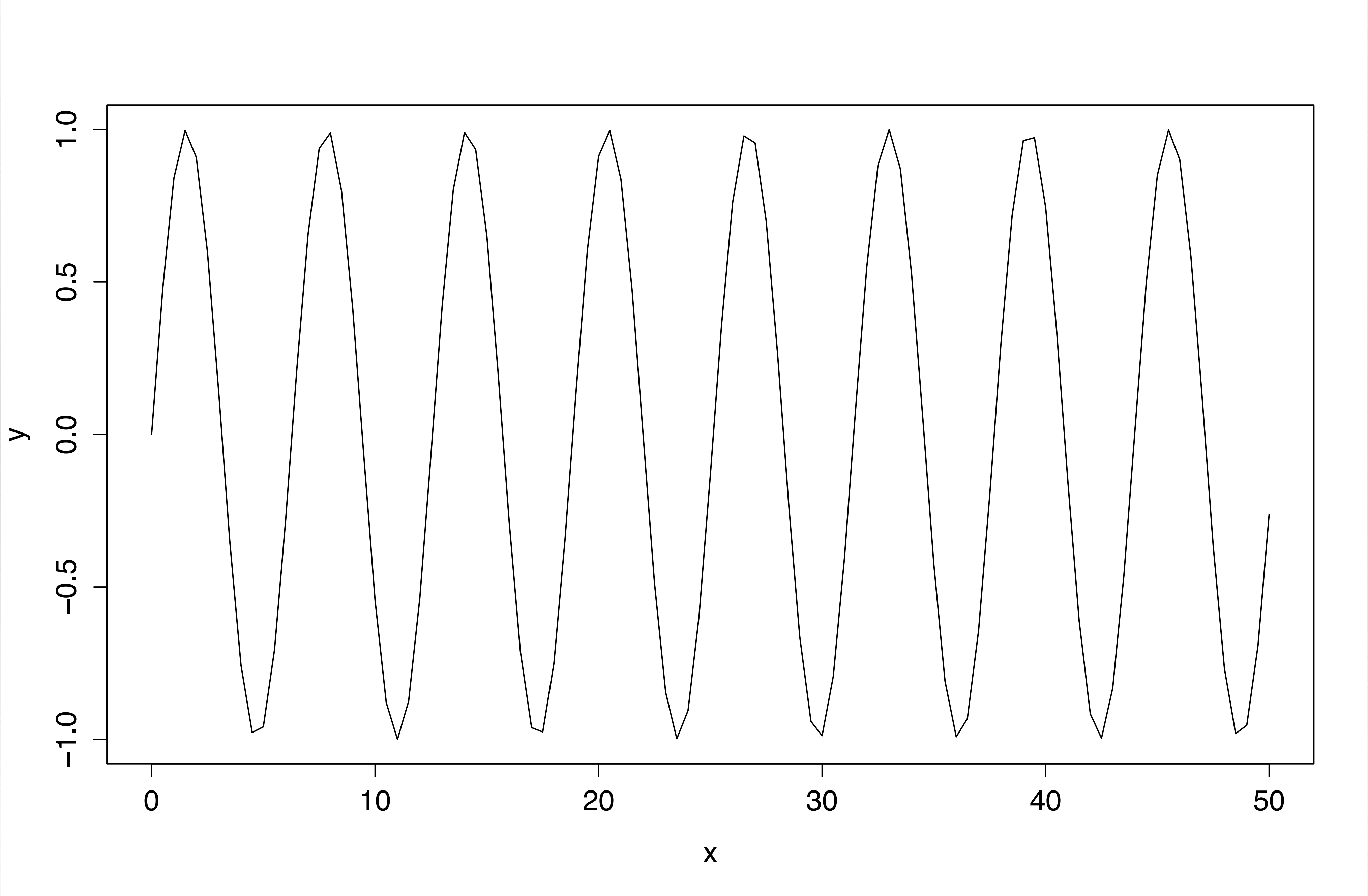


Data visualisation: line graph

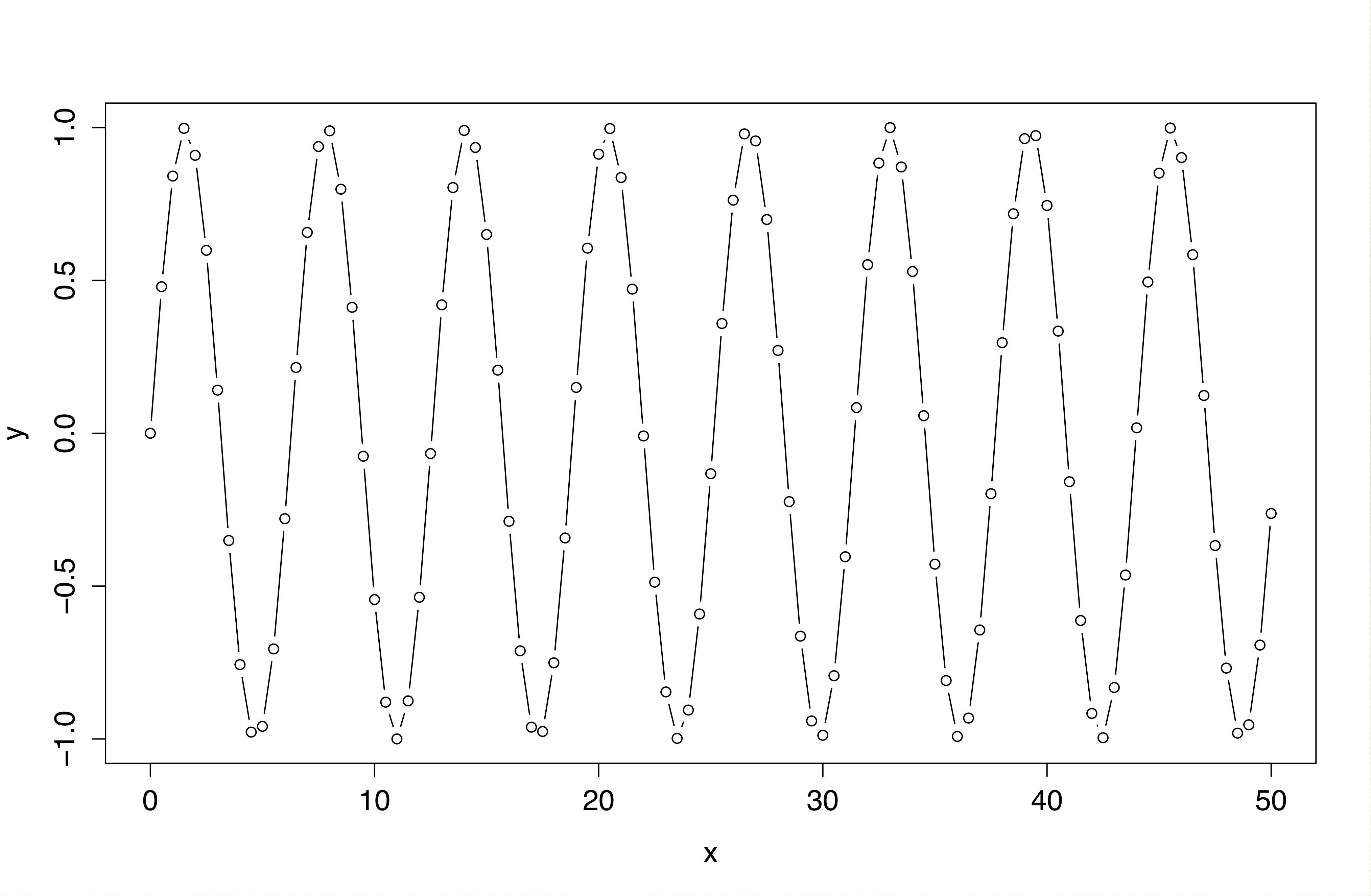
```
x=seq(0,50,.5)
y=sin(x)
plot(x,y)
```



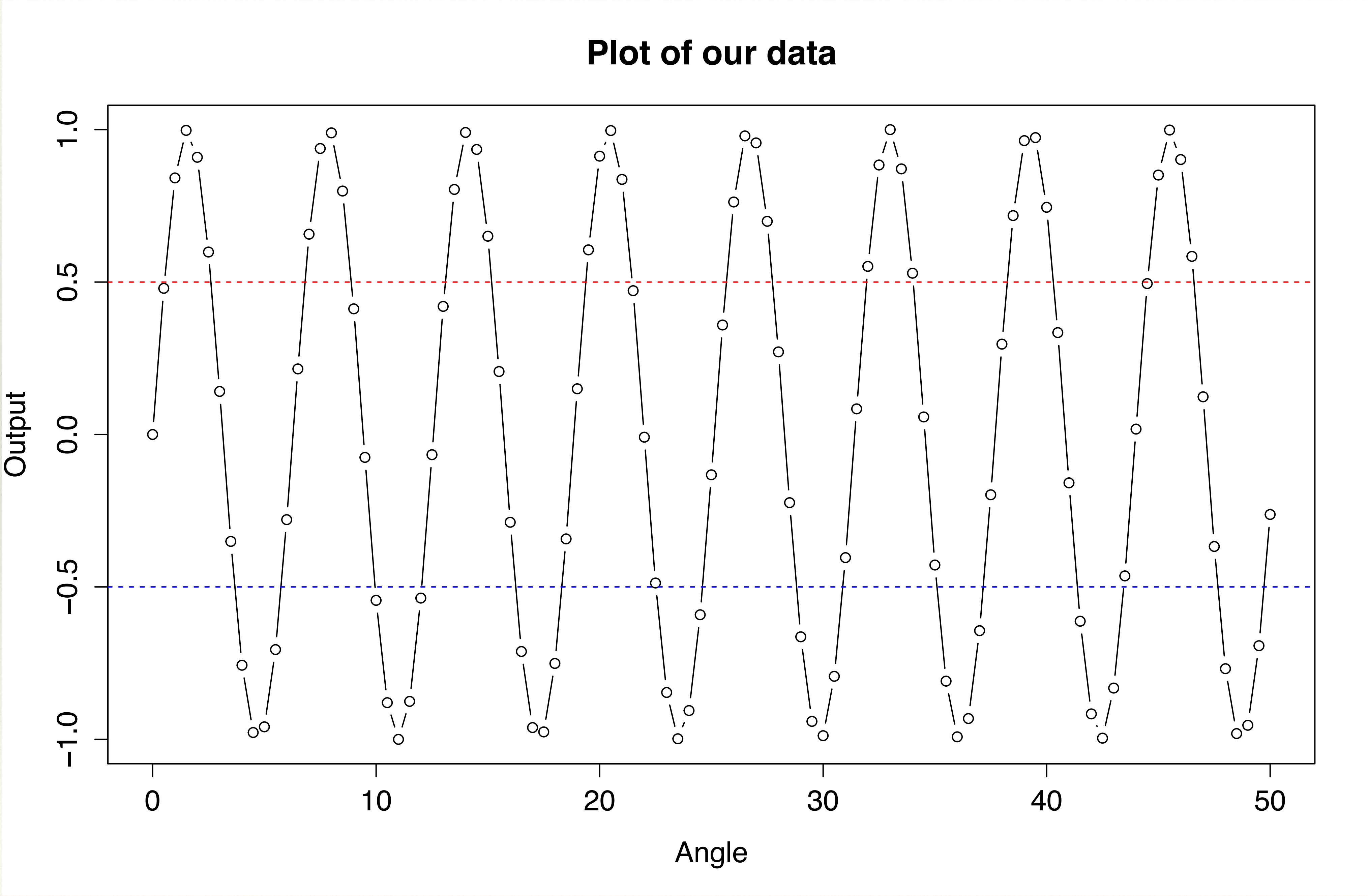
```
plot(x,y,type='l')
```



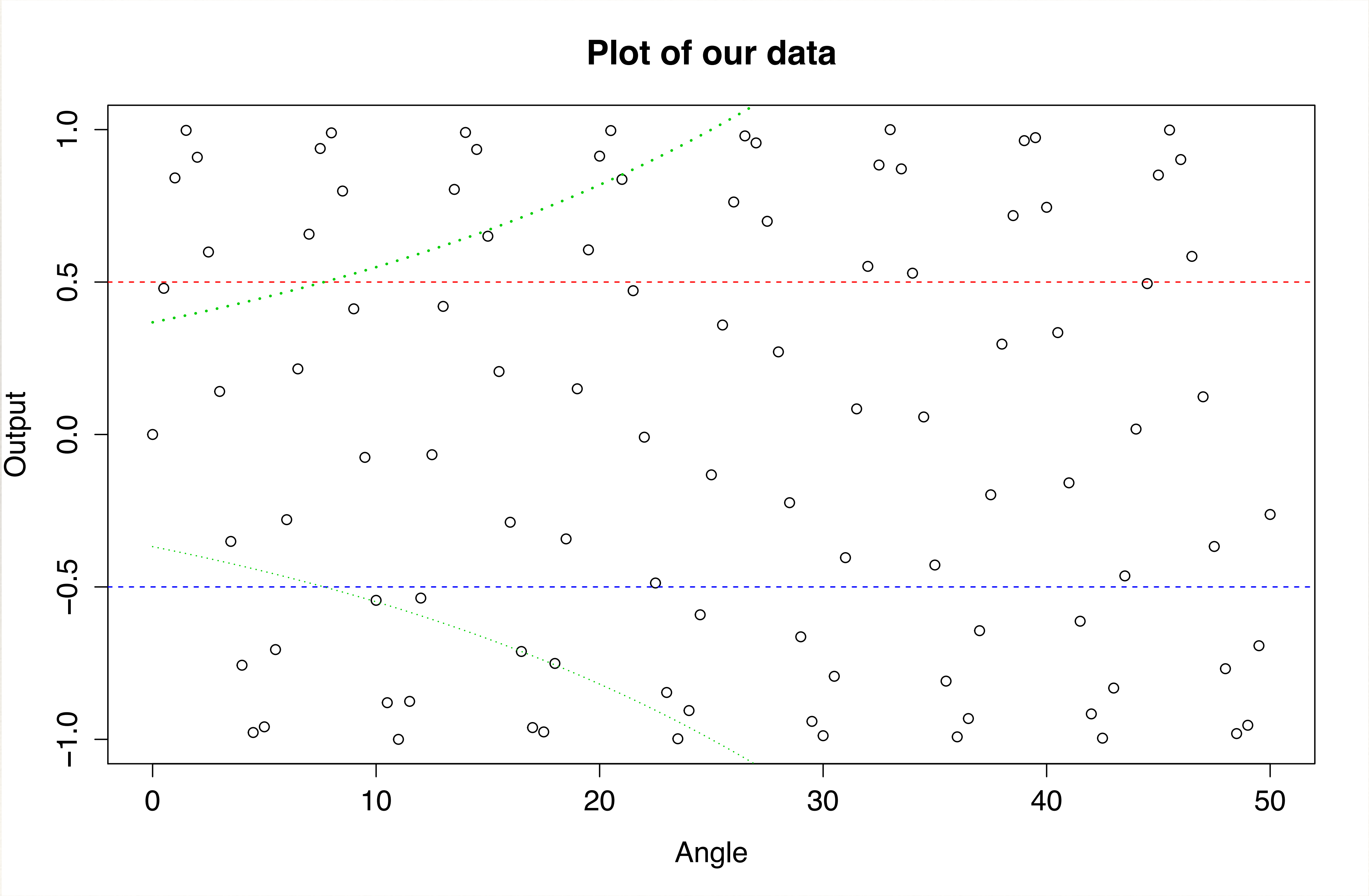
```
plot(x,y,type='b')
```



```
plot(x,y,xlab='Angle',ylab='Output',main='Plot of our data',type='b')
abline(h=.5,lty=2,col=2)
abline(h=-.5,lty=2,col='blue')
```

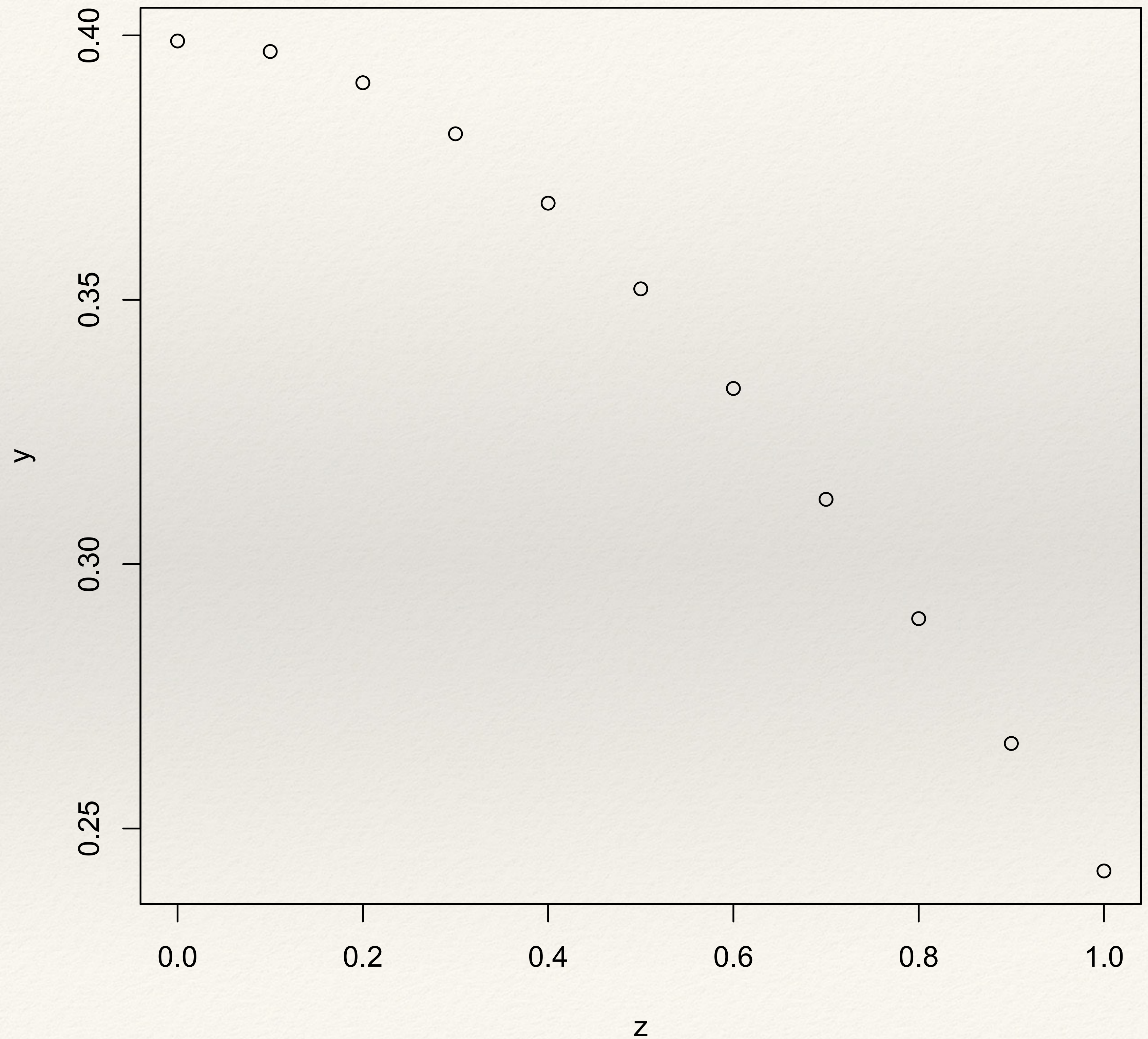


```r
plot(x,y,xlab='Angle',ylab='Output',main='Plot of our data')
abline(h=.5,lty=2,col=2)
abline(h=-.5,lty=2,col='blue')
points(x,-exp(x/25-1),lty=3,col=3,type='l')
points(x,exp(x/25-1),lty=3,col=3,type='l',lwd=2)
```



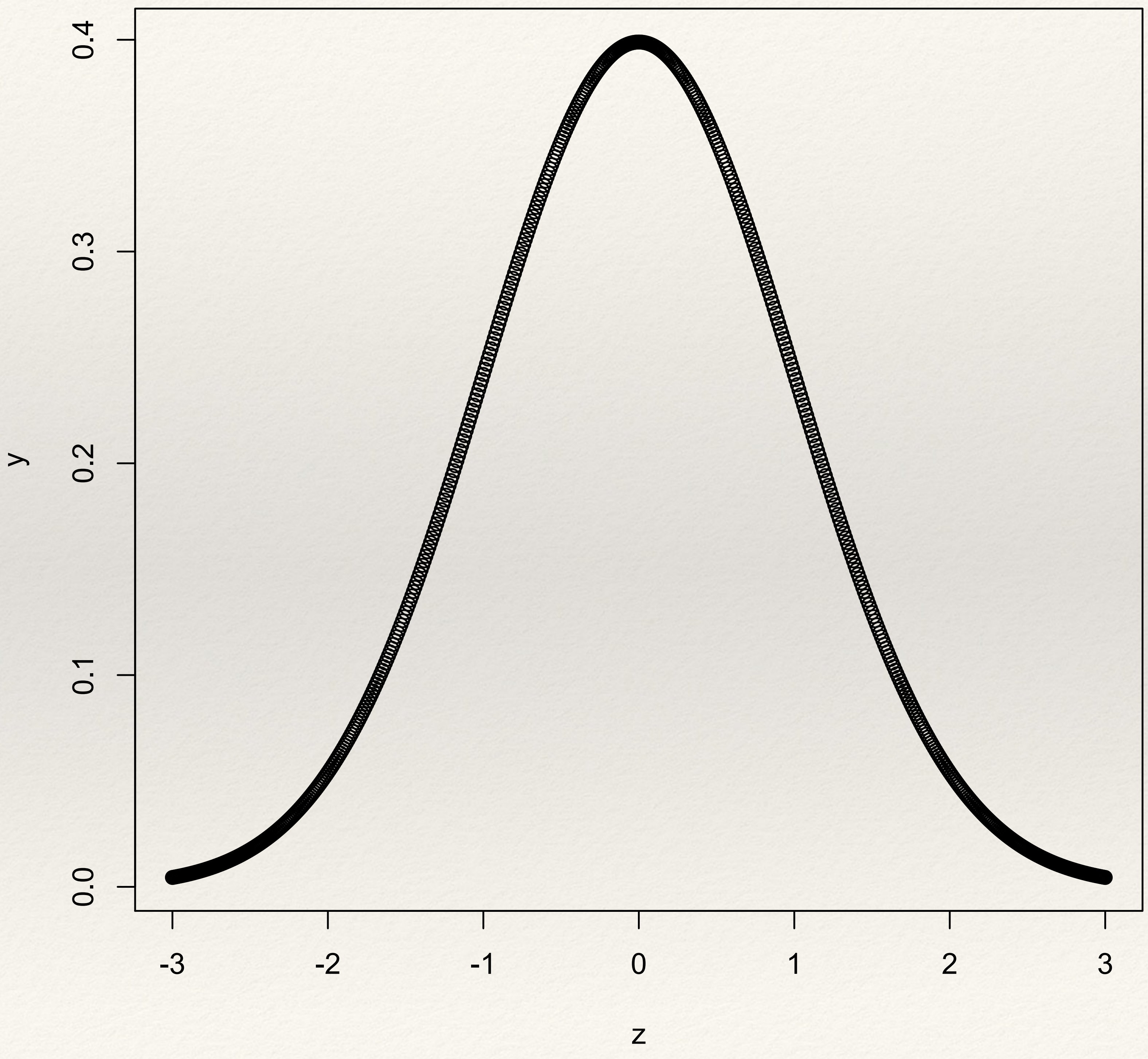
Plotting normal density

```
> z=seq(0,1,.1)
> y=dnorm(z)
> plot(z,y)
```



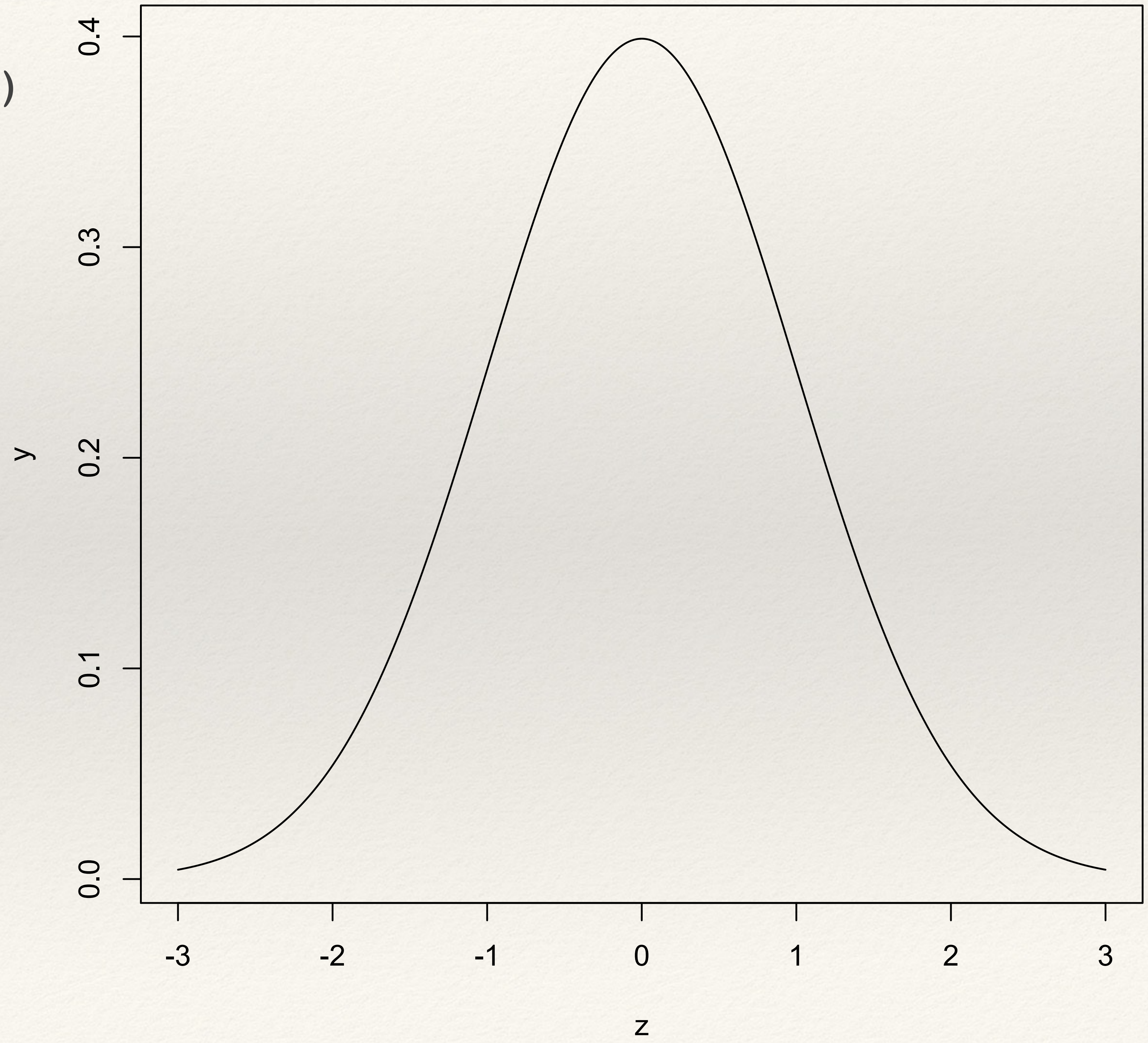
Plotting normal density

```
> z=seq(-3,3,.01)
> y=dnorm(z)
> plot(z,y)
```



Plotting normal density

```
> z=seq(-3,3,.01)
> y=dnorm(z)
> plot(z,y,type='l')
```



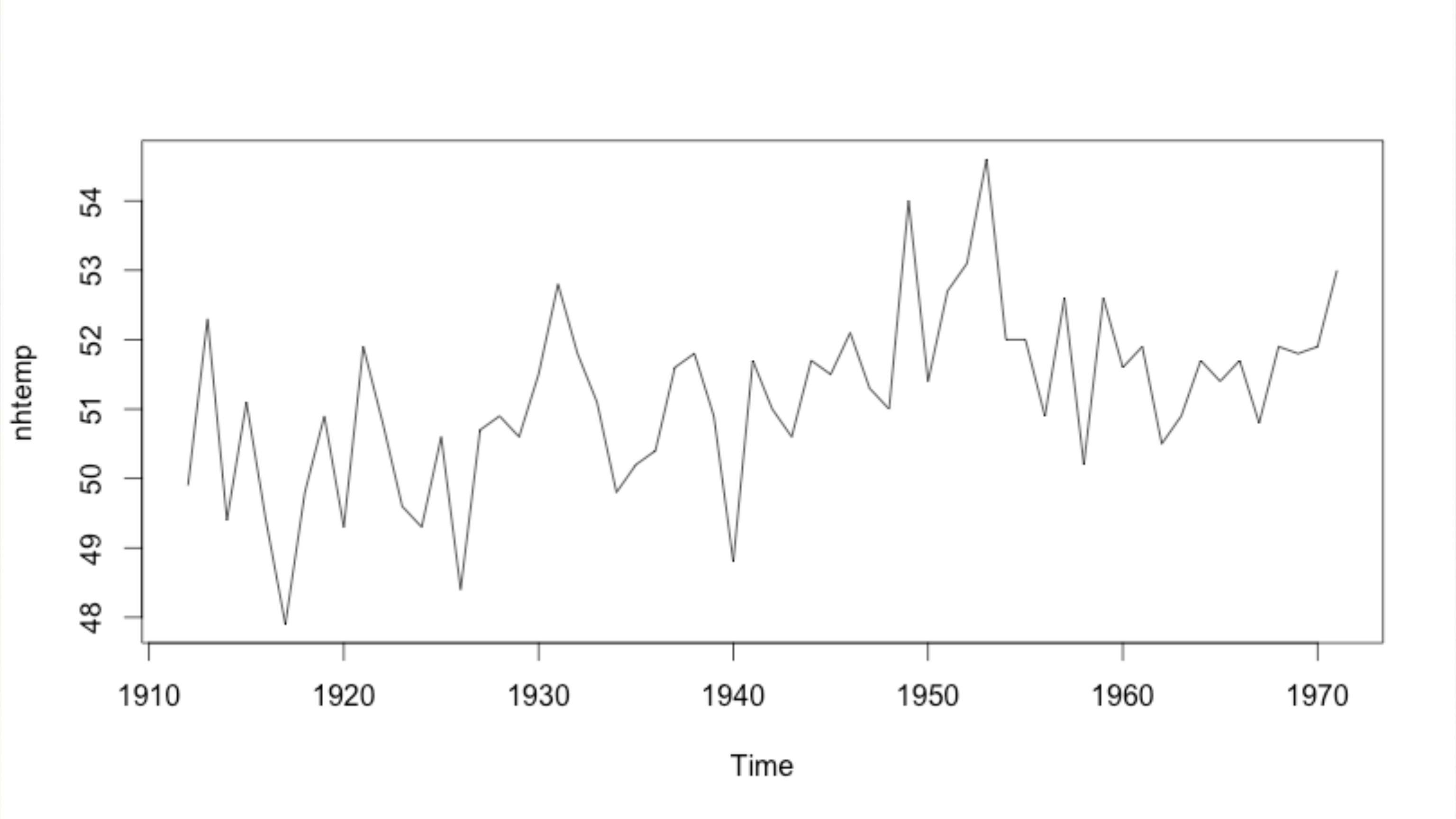
Real data: annual temperatures

`plot` command gives type-appropriate exploratory graphics

nhtemp is a time series of annual mean temperatures in New Haven, Connecticut

```
> data(nhtemp)
> summary(nhtemp)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  47.90   50.58   51.20   51.16   51.90   54.60 
> class(nhtemp)
[1] "ts"
> attributes(nhtemp)
$tsp
[1] 1912 1971    1

$class
[1] “ts"
> plot(nhtemp)
```



Real data: volcano

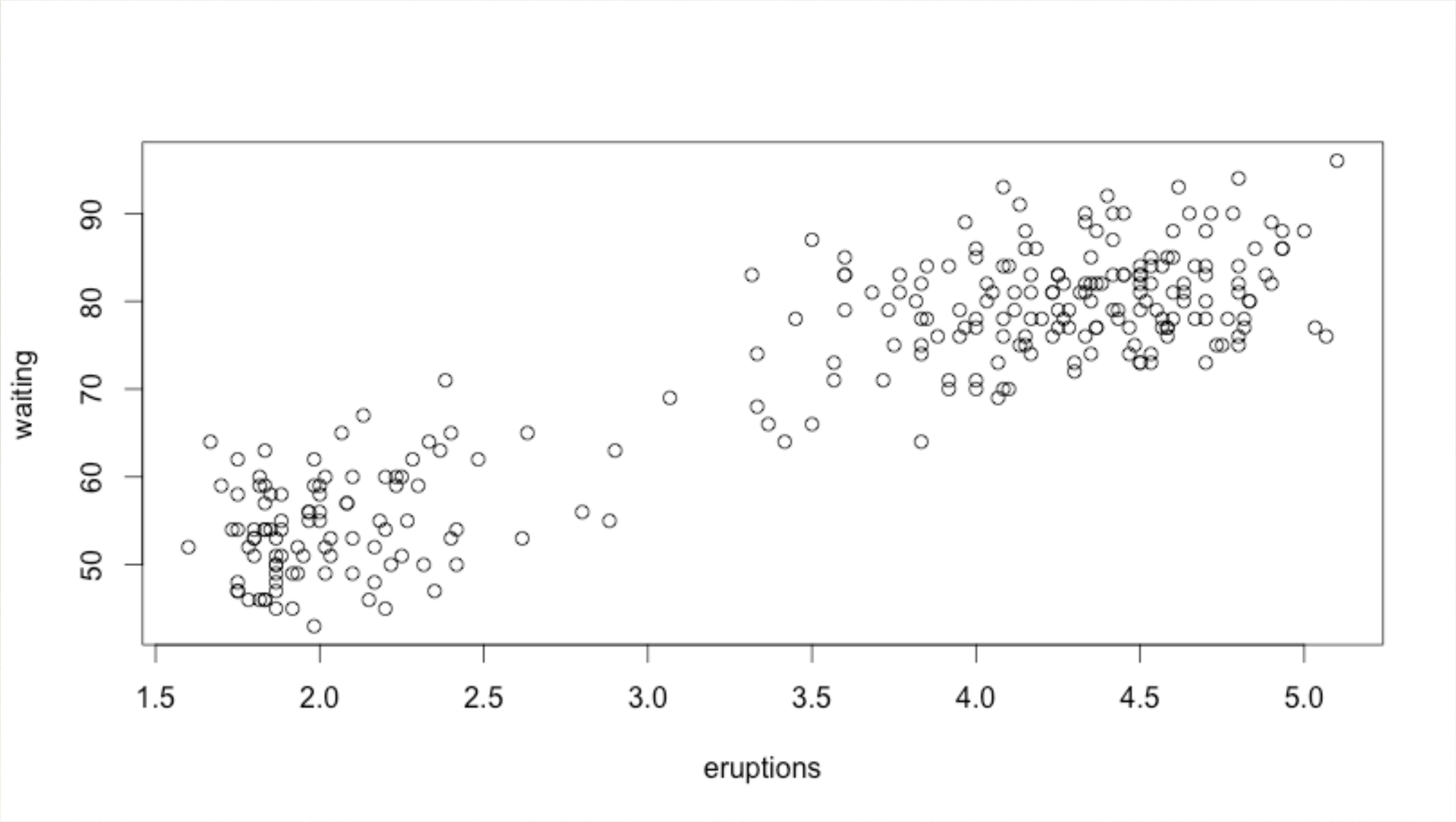
`faithful` is a data.frame giving times between eruptions and duration of eruptions of `the Yellowstone geyser Old Faithful`

```
> data(faithful)
> class(faithful)
[1] "data.frame"
```

```
plot(faithful)
```

Two quantitative variables, produce <u>scatterplot</u> (built in).

```
> summary(faithful)
   eruptions        waiting
 Min.   :1.600   Min.   :43.0
 1st Qu.:2.163   1st Qu.:58.0
 Median :4.000   Median :76.0
 Mean   :3.488   Mean   :70.9
 3rd Qu.:4.454   3rd Qu.:82.0
 Max.   :5.100   Max.   :96.0
```



Manual command:

```
plot(faithful[,1], faithful[,2], xlab="eruptions", ylab="waiting")
```

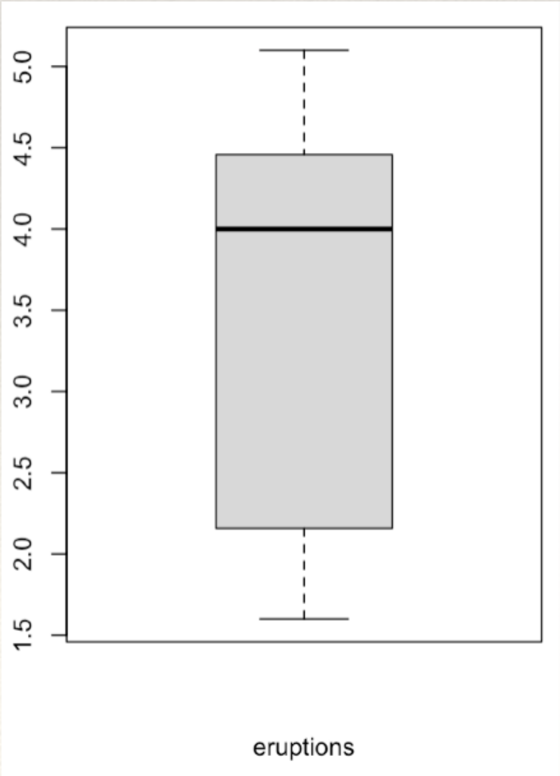
`faithful` is a data.frame giving times between eruptions and duration of eruptions of the Yellowstone geyser Old Faithful

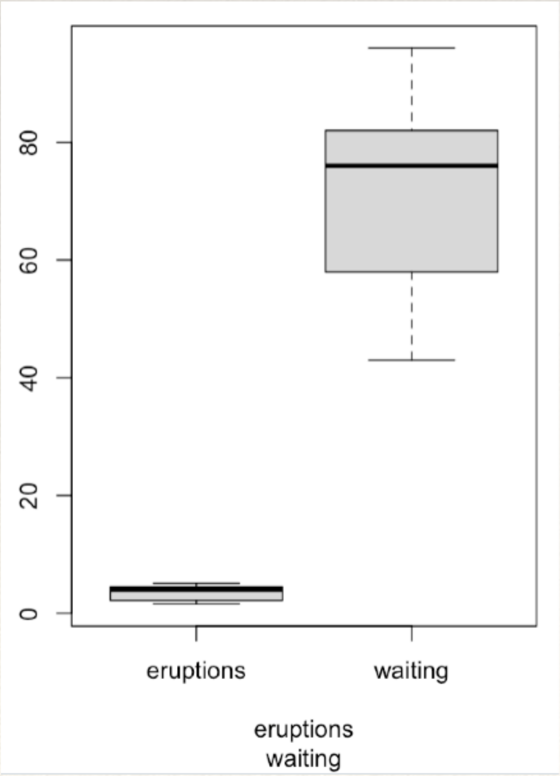
```
> data(faithful)
> class(faithful)
[1] "data.frame"
```

```
> summary(faithful)
   eruptions        waiting
 Min.   :1.600   Min.   :43.0
 1st Qu.:2.163   1st Qu.:58.0
 Median :4.000   Median :76.0
 Mean   :3.488   Mean   :70.9
 3rd Qu.:4.454   3rd Qu.:82.0
 Max.   :5.100   Max.   :96.0
```

Alternatively, make boxplots

```
boxplot(faithful[,1], xlab="eruptions")
boxplot(faithful[,1:2], xlab=c("eruptions", "waiting"))
```





Real data: student's eye colour

`EyeHairColor` is a contingency table of eye colour, hair colour, and sex for 592 students.

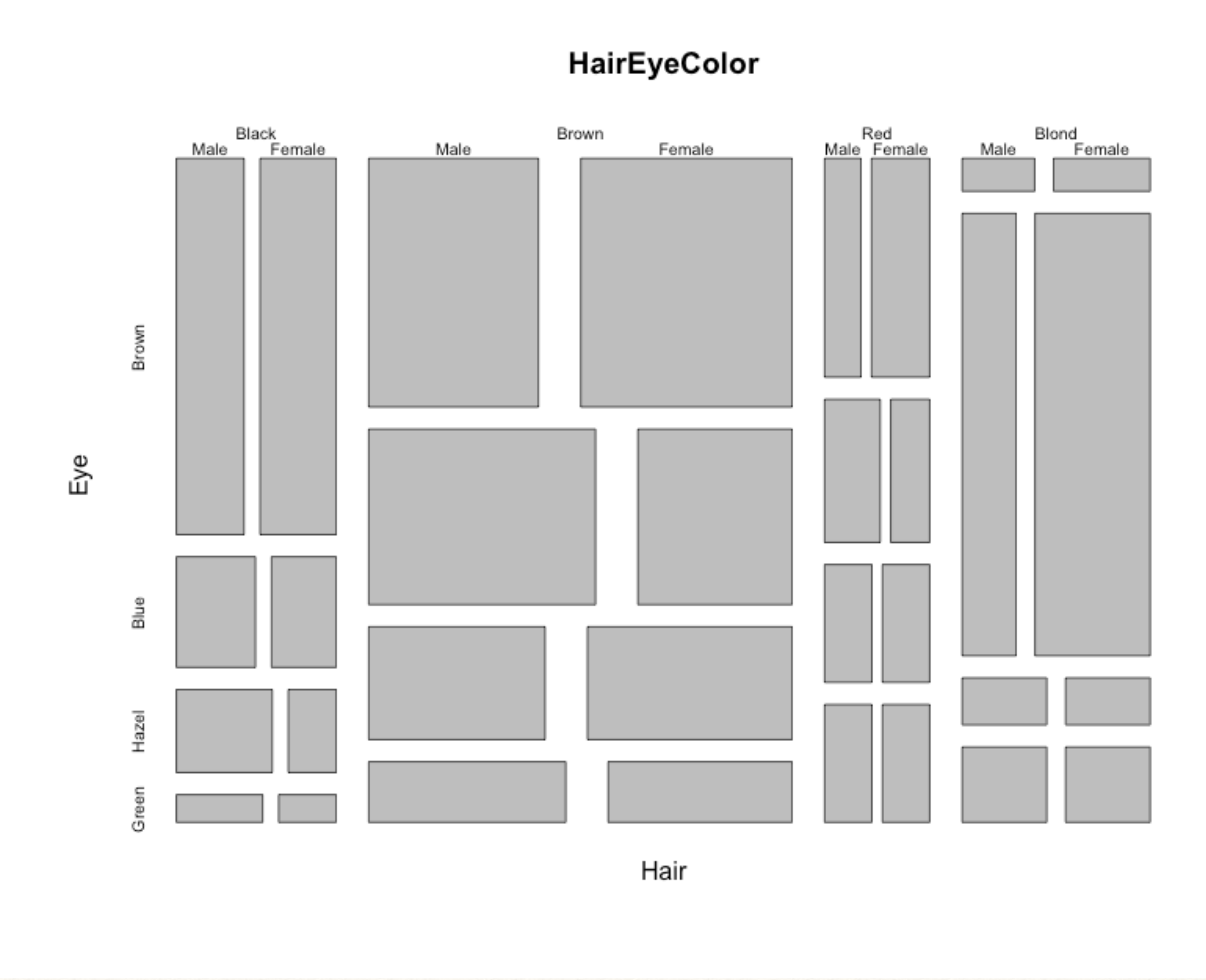
```
> class(HairEyeColor)
[1] "table"
> HairEyeColor
, , Sex = Male

       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    53   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8

, , Sex = Female

       Eye
Hair    Brown Blue Hazel Green
  Black    36    9     5     2
  Brown    66   34    29    14
  Red      16    7     7     7
  Blond     4   64     5     8
```

```
plot(HairEyeColor)
```



Reading data from files

You will need this in the lab!

The previous examples used data build into R.
Now read data from a csv file `mydata.csv from.`
(csv=comma separated values, can be created e.g. in Excel)

Put your file my data into your working directory.
What is that?

```
getwd()
```

Find the current working directory (where inputs are found and outputs are sent).

`setwd('C://file/path')` Change the current working directory.

Read the data into a variable in your R session.

```
T <- read.table("mydata.csv", sep=",")
```

Check out help file of `read.table()`, because there are many options and most dataset have their own formatting issues…

Reading and writing data in files: example

```
> D <- data.frame(weight=rnorm(5,2,1),
+                 height=rnorm(5,10,3),
+                 colour=c("red", "blue", "blue", “red", "blue"))
> write.table(D, "mydata.csv")
> write.table(D, "mydata.csv", sep=",")
> T <- read.table("mydata.csv", sep=",")
> T
    weight    height colour
1 3.455603 11.177369    red
2 1.392296  4.133113   blue
3 3.414024  6.453028   blue
4 2.459340 12.682613    red
5 2.437456 14.932696   blue
```

Create your own questions and tasks!

- In the last lectures I suggested short practical exercise for each topic we studied. These questions were created to help you reviewing, understanding, and remembering the material. Going forward, can generate such questions and tasks yourself? This will make the material more memorisable and lead to a deeper understanding.
- Some ways of *generating* questions and tasks:
  - For which data types does this command work/not work? Why/why not?
  - Can I do something like it was done in a different dimension? Or involving different functions?
  - Pick a few lines of R code from the slides at random and copy them onto a different page. Without looking at the original context, can you say what their output would be and what they do?
  - For data visualisations: Why is this an effective way to communicate the message in the data? Why not? How could you improve it?
  - Try to "break" some R code from the lecture slides. What can you change about the commands or the arguments so it results in an error message?

This afternoon: Basic R programming

Functions: In addition to built in functions like sin(), sort(), runif() you can create you own functions.

Controlling flow: Manage which parts of your R script are being executed and when.