Chapter 5 Graphics; Chapter 6 Tables

Qianqian Shan

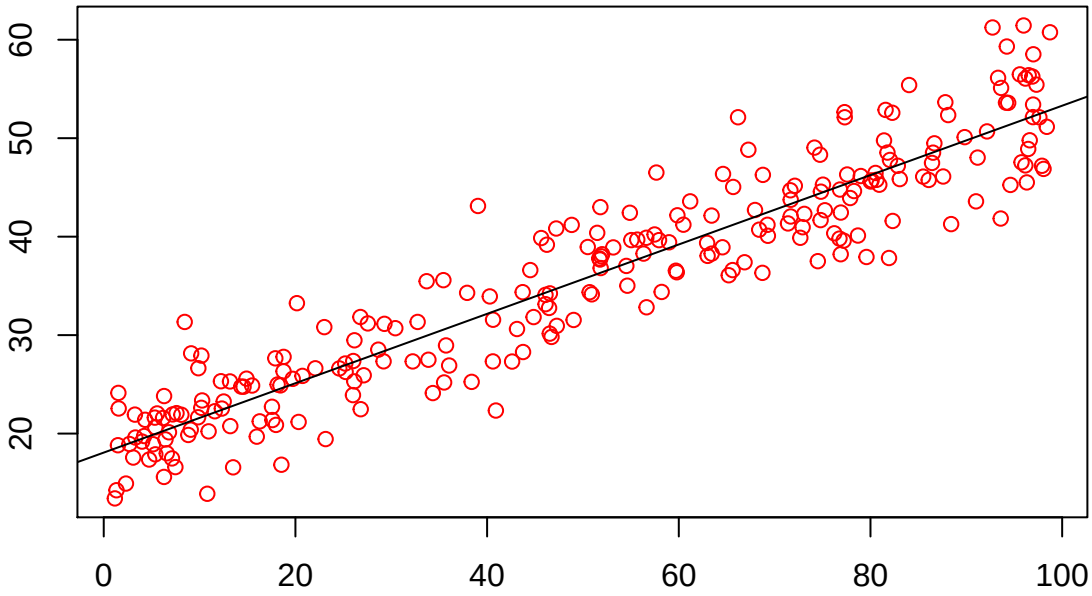
May 22, 2017

Plot with two variables

1. **Scatterplot** : when explanatory variable is continuous
2. **box-and-whisker plot** : when explanatory variable is categorical
3. **barplot** :for categorical variable with more emphasis on effect sizes

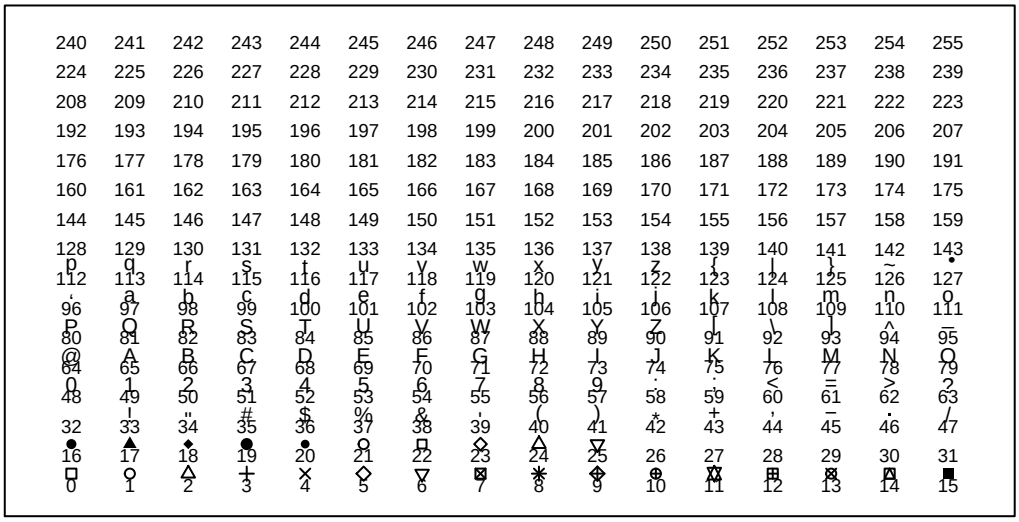
Plotting with two continuous variables : `Scatterplots`

```
data2 <- read.table("scatter2.txt", header = TRUE)
plot(data2$xv2, data2$ys2, col = "red", xlab = "", ylab = "")
abline(lm(ys2 ~ xv2, data = data2))
```




```
# legend(locator(1), c("treatment"), col = "red", pch = 1)


# pch
plot(0:10, 0:10, xlim = c(0, 32), ylim = c(0, 40), type = "n", xaxt = "n", yaxt = "n",
     xlab ="", ylab = "")
x <- seq(1, 31, 2)
s <- -16
f <- -1
for(y in seq(2, 40, 2.5)){
  s <- s + 16
  f <- f + 16
  y2 <- rep(y, 16)
  points(x, y2, pch = s:f, cex = 0.7)
```

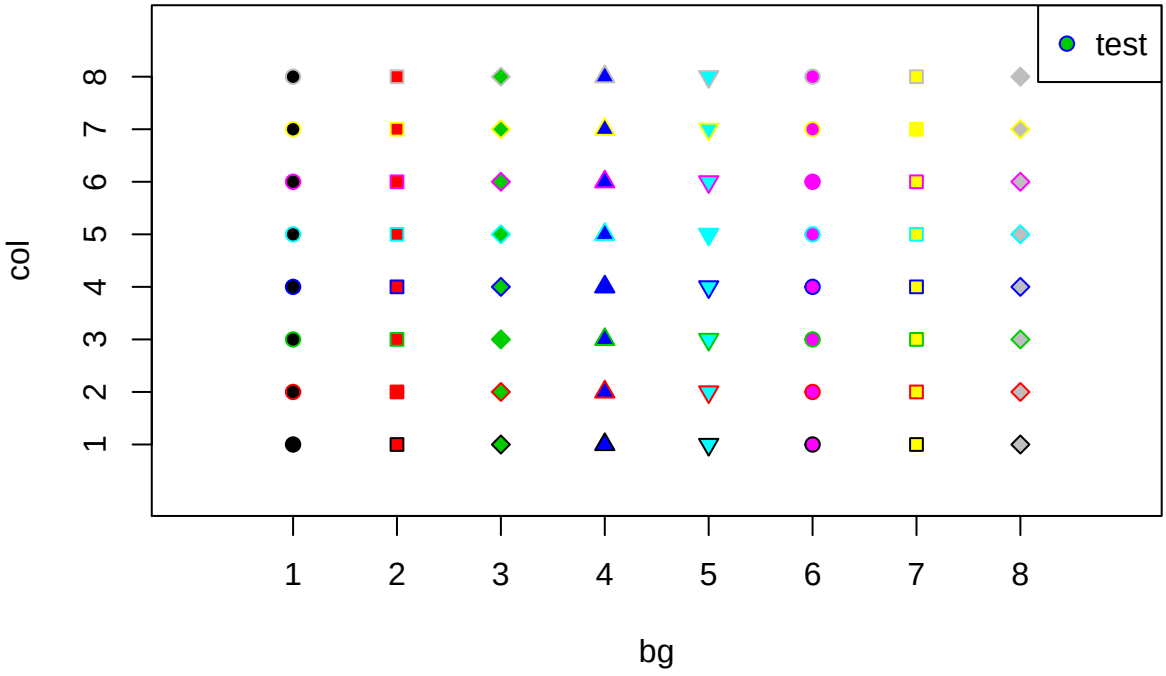
```
  text(x, y - 1, as.character(s:f), cex = 0.6)
}
```



color for symbols

pch 21 - 25 allows to specify the background(fill) color and the border color separately.

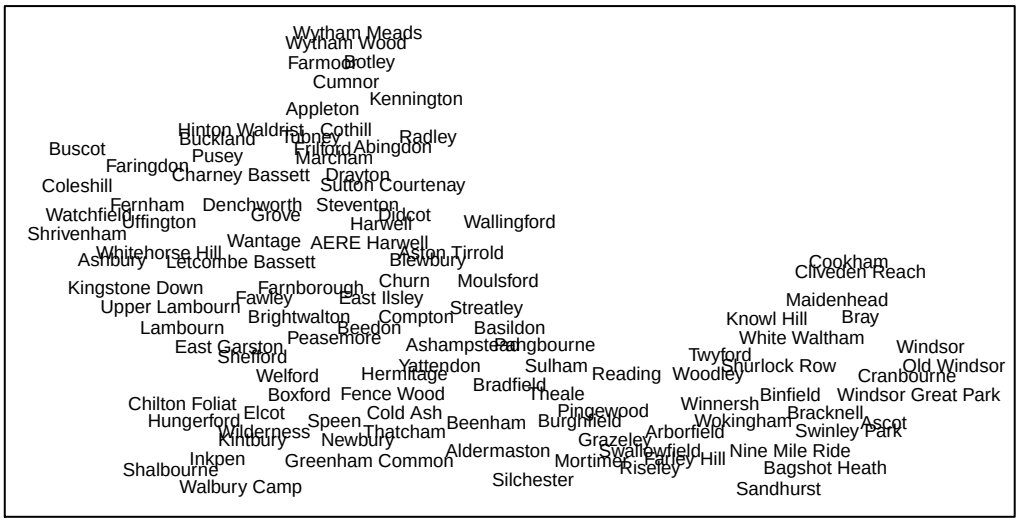
```
plot(0:9, 0:9, type = "n", xaxt = "n", yaxt = "n", ylab = "col", xlab = "bg")
axis(1, at = 1:8)
axis(2, at = 1:8)
# bg: color for background
# col: color for border
for(i in 1:8) points(1:8, rep(i, 8), pch = c(21, 22, 23, 24, 25), bg = 1:8, col = i)
legend("topright", c("test"), pch = 21, pt.bg = 3, col = 4)
```



```
# pt.bg is used for specifying the fill color
```

Adding text to scatterplots

```
map.place <- read.csv("map.places.csv")
map.data <- read.csv("bowens.csv")
map.data$nn <- ifelse(map.data$north < 60, map.data$north + 100, map.data$north)

attach(map.place)
attach(map.data)
# produce a map with places names with corresponding locations
plot(c(20, 100), c(60, 110), type = "n", xlab = "", ylab = "", xaxt = "n", yaxt = "n")
for(i in 1:length(map.place$wanted)){
  ii <- which(map.data$place == as.character(map.place$wanted[i]))
  text(east[ii], nn[ii], as.character(place[ii]), cex = 0.6)
}
```

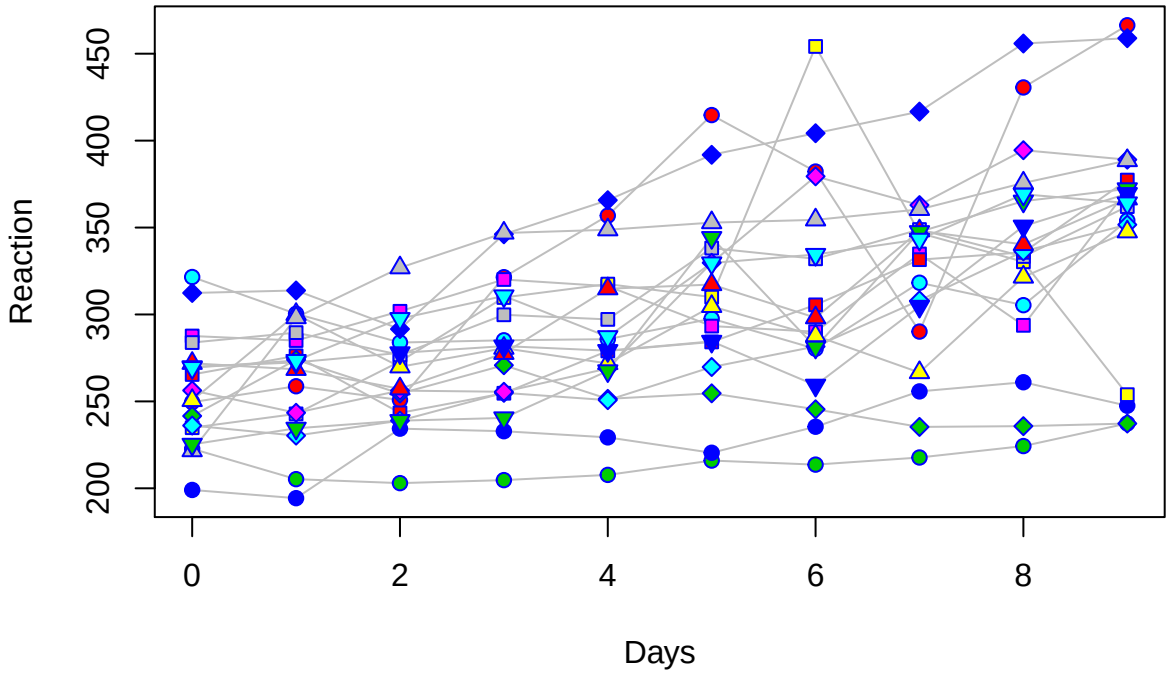



```
detach(map.place)
detach(map.data)
```

Identifying individuals in scatterplots

```
data <- read.table("sleep.txt", header = TRUE)
attach(data)
s <- as.numeric(factor(Subject))
plot(Days, Reaction, type = "n")
for(k in 1:max(s)){
  x <- Days[s == k]
  y <- Reaction[s == k]
  lines(x, y, type = "l", col = "gray")
}

sym <- rep(c(21, 22, 23, 24, 25), c(4, 4, 4, 3, 3))
bcol <- c(2:8, 2:8, 2:5)
```

```
for(k in 1:max(s)){
  points(Days[s == k], Reaction[s == k], pch = sym[k], bg = bcol[k], col = 4)
}
```



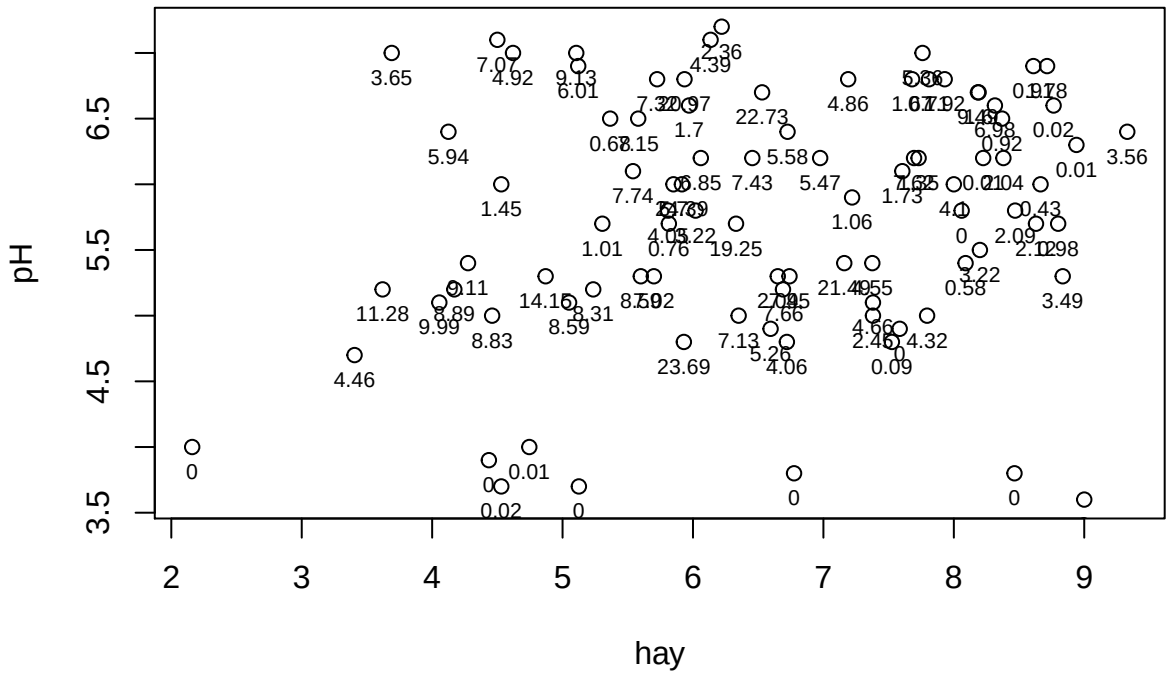

```
detach(data)
```

Using a third variable to label a scatterplot `text()`

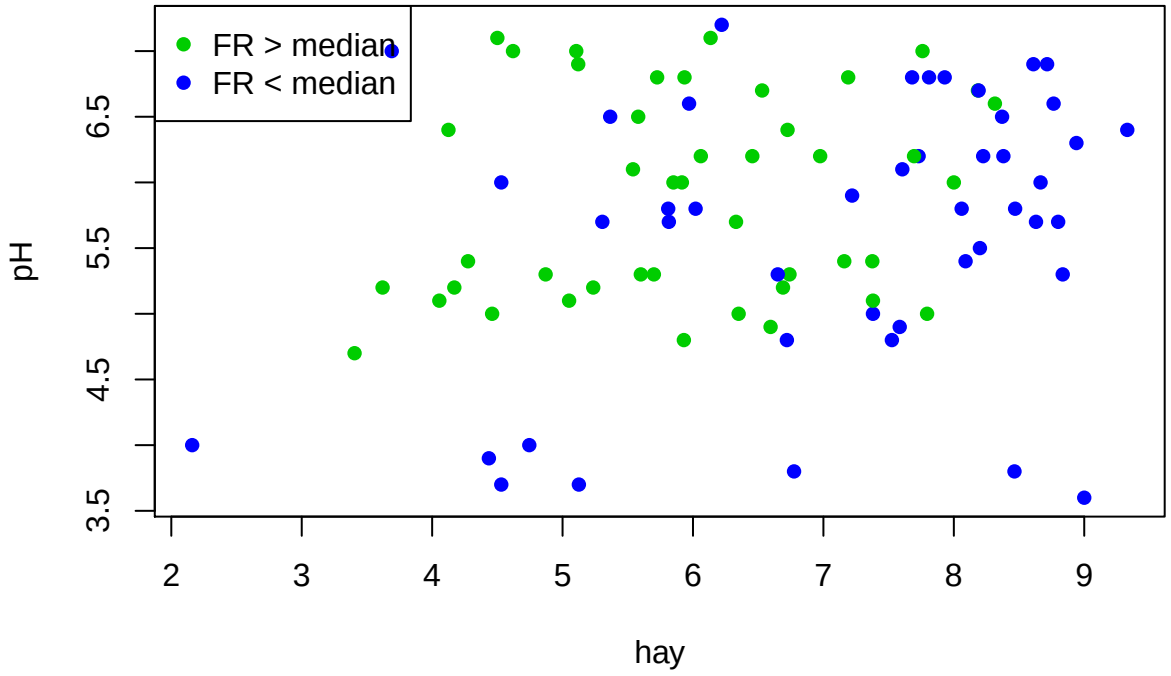
```
data <- read.table("pgr.txt", header = TRUE)
names(data) # label with FR
```

```
## [1] "FR"  "hay" "pH"
```

```
attach(data)
plot(hay, pH)
text(hay, pH, labels = round(FR, 2), pos = 1, offset = 0.5, cex = 0.7)
```




```
# pos = 1 : labels are centered
# offset = 0.5 : when pos is specified, this value gives the offset of the label from the specified
# coordinate in fractions of a character width.


# use the third variable to choose the color of points
plot(hay, pH, pch = 16, col = ifelse(FR > median(FR), 3, 4))
legend("topleft", c("FR > median", "FR < median"), col = c(3, 4), pch = 16)
```



```
detach(data)
```

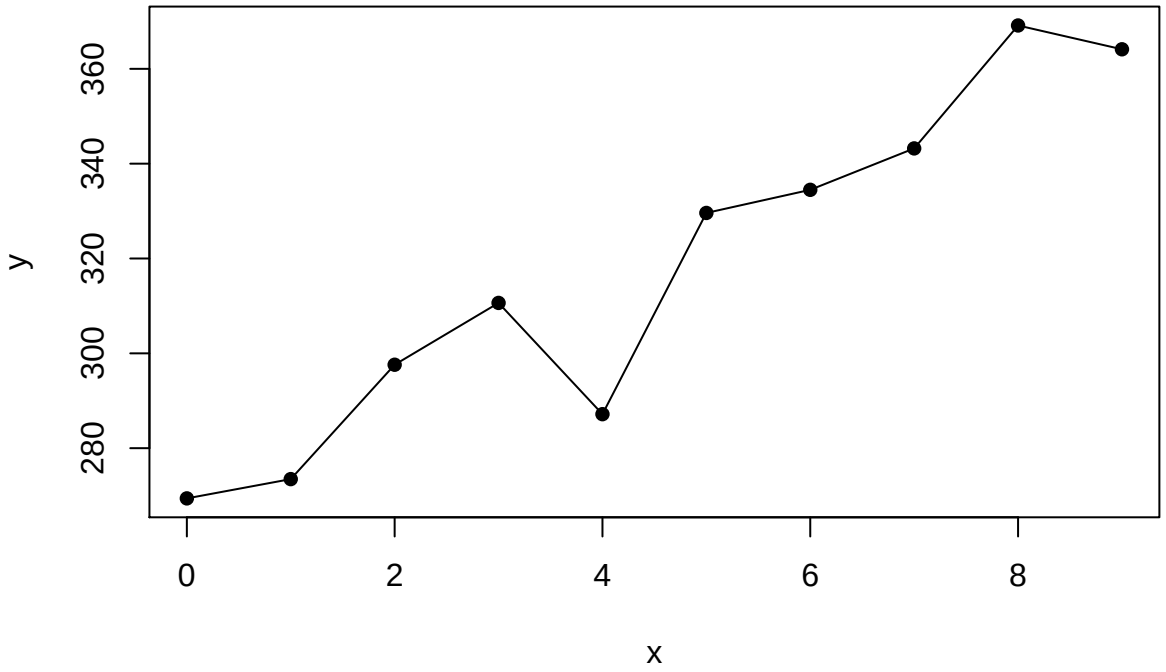
Joining the dots

Need to first order the points on hte x axis in order to connect by lines

```
smooth <- read.table("smoothing.txt", header = TRUE)
attach(smooth)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     x, y
```

```
sequence <- order(x)
plot(x, y, pch = 16)
lines(x[sequence], y[sequence])
```




```
detach(smooth)
```

Plotting stepped lines

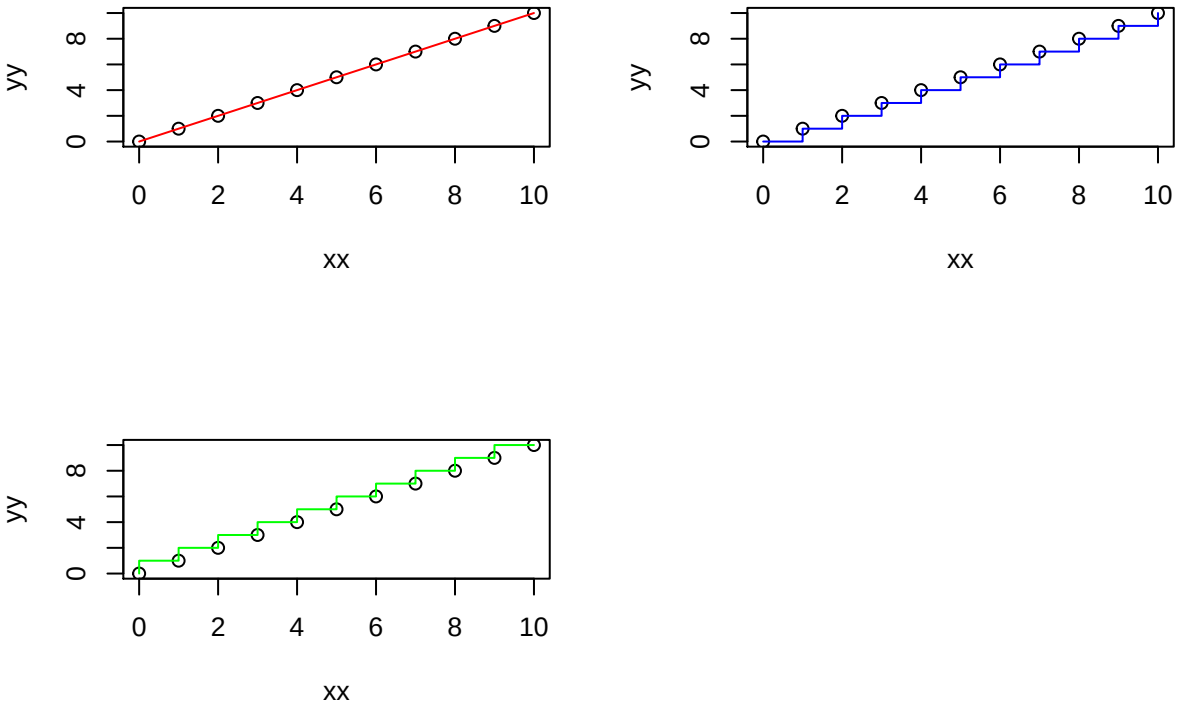
type : 1-character string giving the type of plot desired.

The following values are possible, for details, see plot:

1. “p” for points, “l” for lines,
2. “b” for both points and lines,
3. “c” for empty points joined by lines,
4. “o” for overplotted points and lines,
5. “s” and “S” for stair steps and

6. “h” for histogram-like vertical lines.
7. Finally, “n” does not produce any points or lines

```
xx <- 0:10
yy <- 0:10
par(mfrow = c(2, 2))
plot(xx, yy)
lines(xx, yy, col = "red")

plot(xx, yy)
lines(xx, yy, col = "blue", type = "s")

plot(xx, yy)
lines(xx, yy, col = "green", type = "S")
par(mfrow = c(1 ,1))
```



Adding other shapes to a plot

Special objects: * rect : rectangular

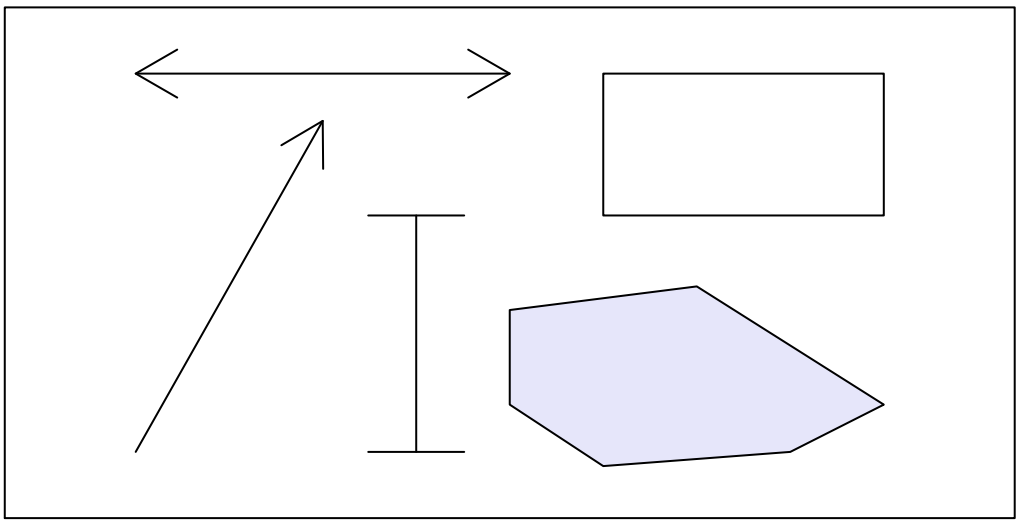
- arrows
- polygon

```
plot(0:10, 0:10, xlab = "", ylab = "", xaxt = "n", yaxt = "n", type = "n")
rect(6, 6, 9, 9)
# rect(xleft, ybottom, xright, ytop, density = NULL, angle = 45,
#      col = NA, border = NULL, lty = par("lty"), lwd = par("lwd"),
#      ...)
corners <- function(){
coos <- c(unlist(locator(1)),unlist(locator(1)))
rect(coos[1],coos[2],coos[3],coos[4])
```

```
}

corners()

arrows(1,1,3,8)

arrows(1,9,5,9,code=3)

arrows(4,1,4,6,code=3,angle=90)


# If we want to use the cursor position for the arrow

# click.arrows <- function(){
# coos <- c(unlist(locator(1)),unlist(locator(1))) arrows(coos[1],coos[2],coos[3],coos[4])
# }

# click.arrows()

# locations <- locator(6)

locations <- vector("list", 2)
class(locations)
```

```
## [1] "list"
```

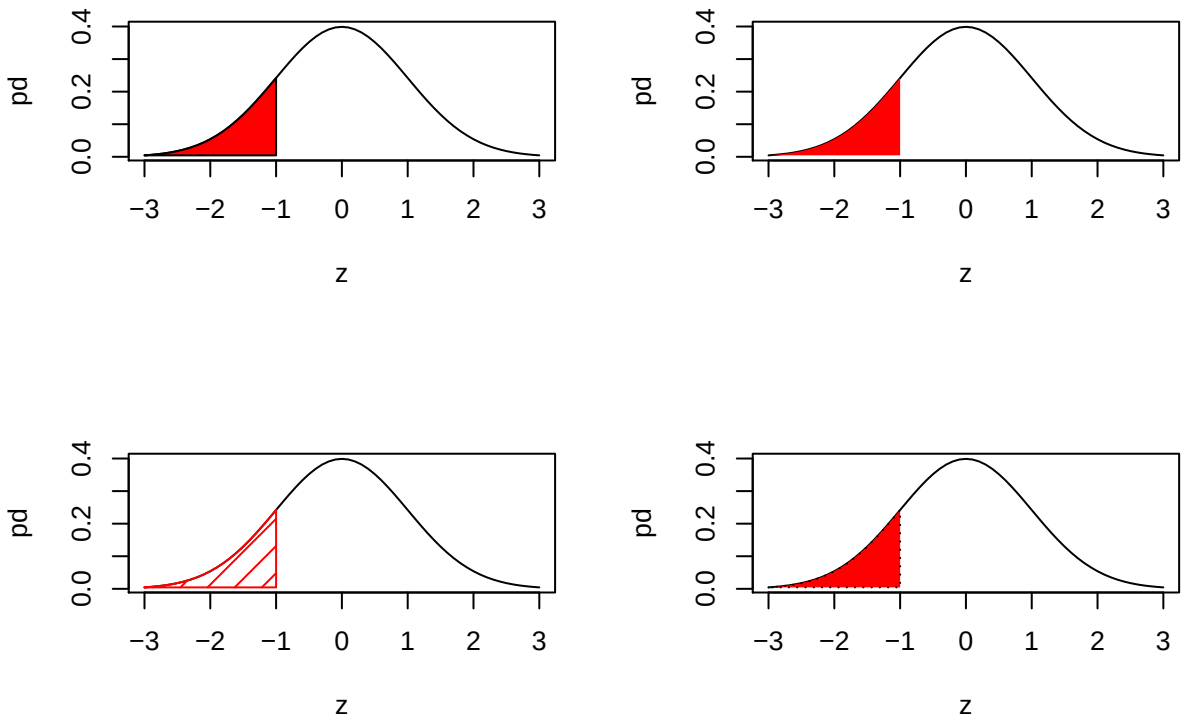
```
names(locations) <- c("x", "y")
locations$x <- c(5, 7, 9, 8, 6, 5)
locations$y <- c(4, 4.5, 2, 1, 0.7, 2)
# polygon draws the polygons whose vertices are given in x and y.
polygon(locations,col="lavender")
```



Draw curverd shapes using `polygon`

```
# shade an area below a curve
z <- seq(-3, 3, 0.1)
```

```r
pd <- dnorm(z)
par(mfrow = c(2, 2))
plot(z, pd, type = "l")
# shade it
polygon(c(z[z <= -1], -1), c(pd[z <= -1], pd[z == -3]), col = "red")


plot(z, pd, type = "l")
# shade it
polygon(c(z[z <= -1], -1), c(pd[z <= -1], pd[z == -3]), col = "red", border = NA)

plot(z, pd, type = "l")
# shade it
polygon(c(z[z <= -1], -1), c(pd[z <= -1], pd[z == -3]), density = 10, col = "red")
# density is in lines per inch

plot(z, pd, type = "l")
# shade it
polygon(c(z[z <= -1], -1), c(pd[z <= -1], pd[z == -3]), col = "red", lty = 3)
```



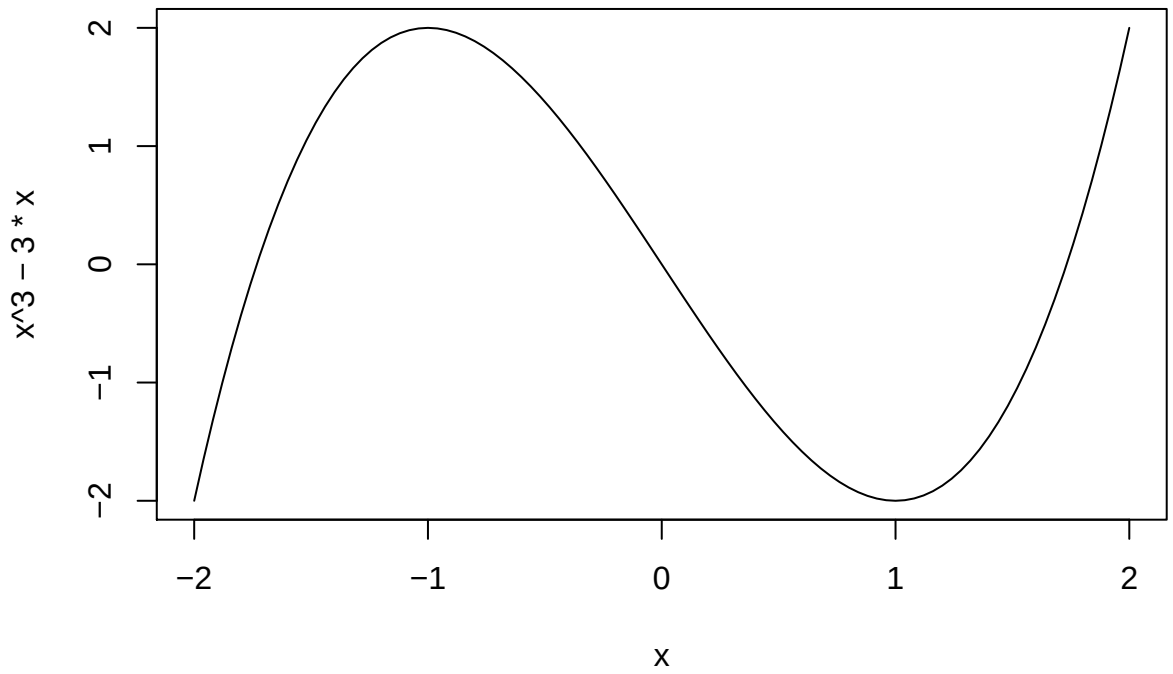

```r
par(mfrow = c(1, 1))
```

Drawing mathematical functions

- Use `curve` to draw funcitons directly
- smoothing lines with `lines`
- non-parametric curves using `lowess`, `loess` , `gam` and `lm`
- `lowess` is curve fitter, eg. , lines(lowess(age, bone))

- `loess` is a modeling tool, eg. loess(bone ~ age)

```
curve(x^3 - 3*x, -2, 2)
```

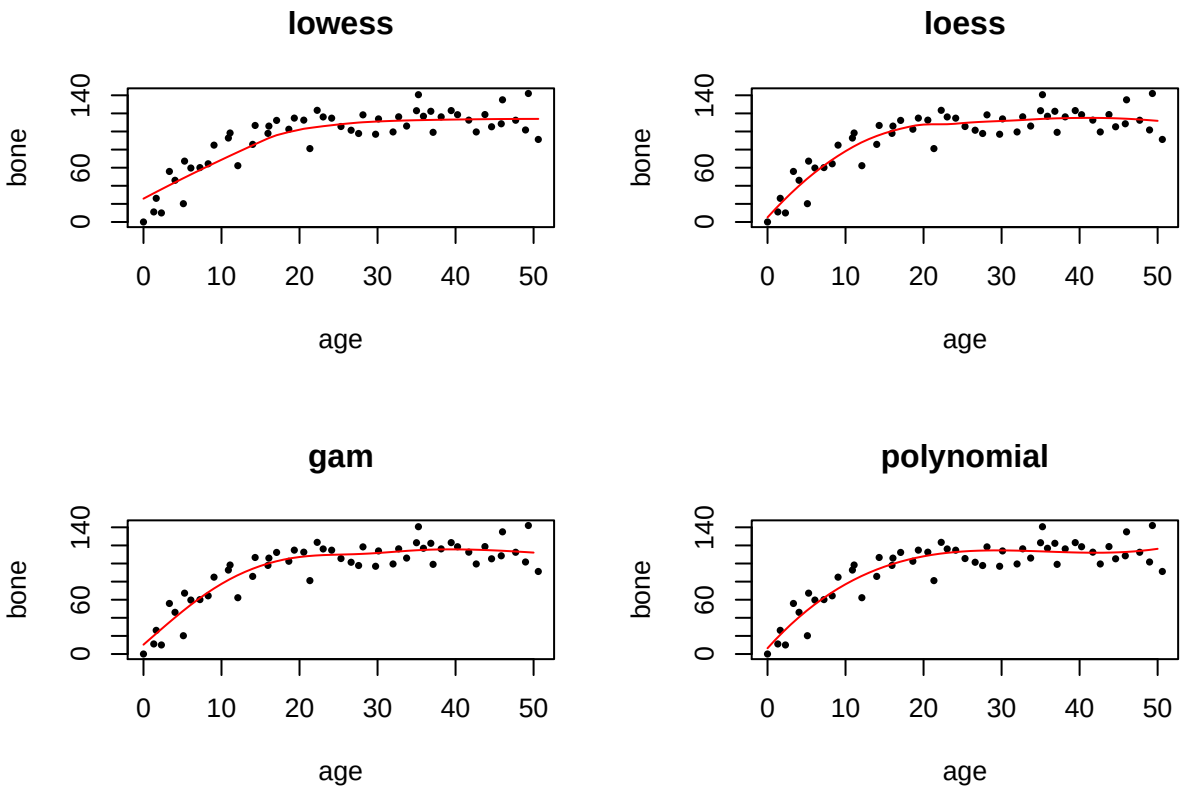



```
# show the effects of lowess, loess, gam and lm
data <- read.table("jaws.txt", header = TRUE)
attach(data)
par(mfrow = c(2, 2))
plot(age, bone, pch = 16, cex = 0.6, main = "lowess")
lines(lowess(age, bone), col = "red")

plot(age, bone, pch = 16, cex = 0.6, main = "loess")
model <- loess(bone ~ age)
xv <- 0:50
yv <- predict(model, data.frame(age = xv))
lines(xv, yv, col = "red")
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-16. For overview type 'help("mgcv-package")'.
```

```
plot(age, bone, pch = 16, cex = 0.6, main = "gam")
model <- gam(bone ~ s(age))
xv <- 0:50
yv <- predict(model, data.frame(age = xv))
lines(xv, yv, col = "red")


plot(age, bone, pch = 16, cex = 0.6, main = "polynomial")
model <- lm(bone ~ age + I(age^2) + I(age^3))
xv <- 0:50
yv <- predict(model, data.frame(age = xv))
lines(xv, yv, col = "red")
```




```
par(mfrow = c(1, 1))
# call the "s" function from package "mgcv"
# s: Function used in definition of smooth terms within gam model formulae. The function does
# not evaluate a (spline) smooth - it exists purely to help set up a model using spline based smooths
detach(data)
```

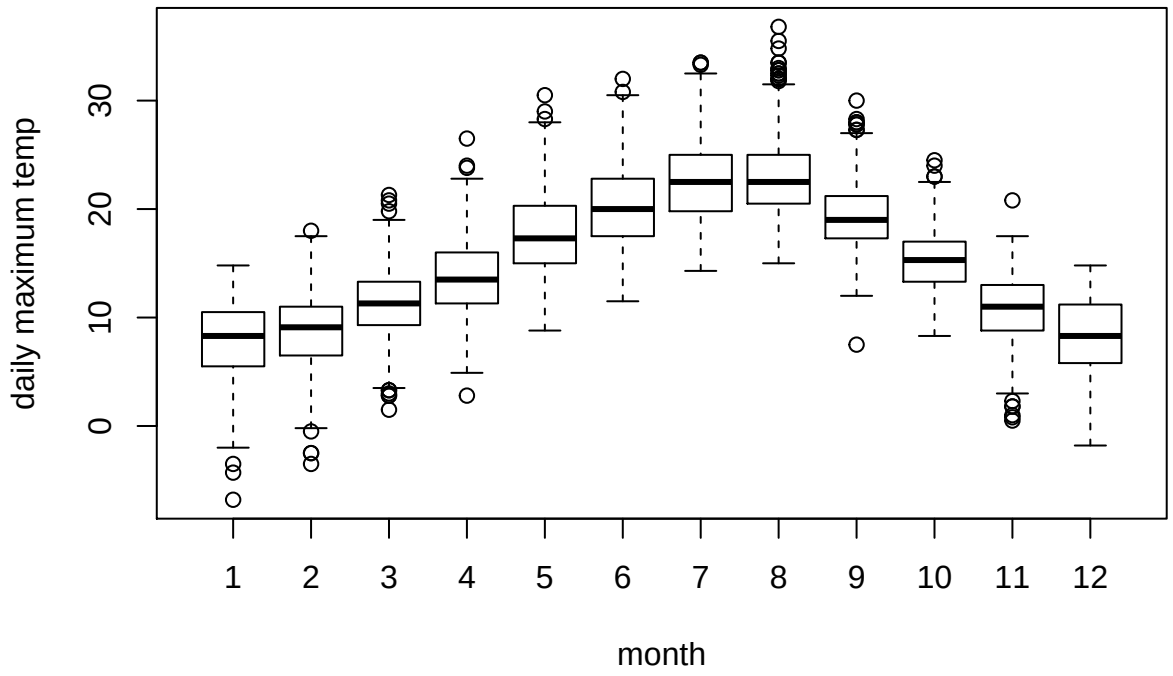
Plotting with categorical explanatory variables

- **box and whisker plot** : the upper whisker is the largest **data point** that is less than the 1.5 interquantile range above the 75th percentile. While the **interquantile** is the difference in the response variable between the first and third quantile.
  `notches` $= \pm 1.58 \frac{IQR}{\sqrt{n}}$, with n the sample size. It's based on the assumptions of asymptotic normality of the median and roughly **equal** sample size for two medians being compared. The **idea** is to give roughly 95% confidence intervals for the difference in two medians.
- barplot

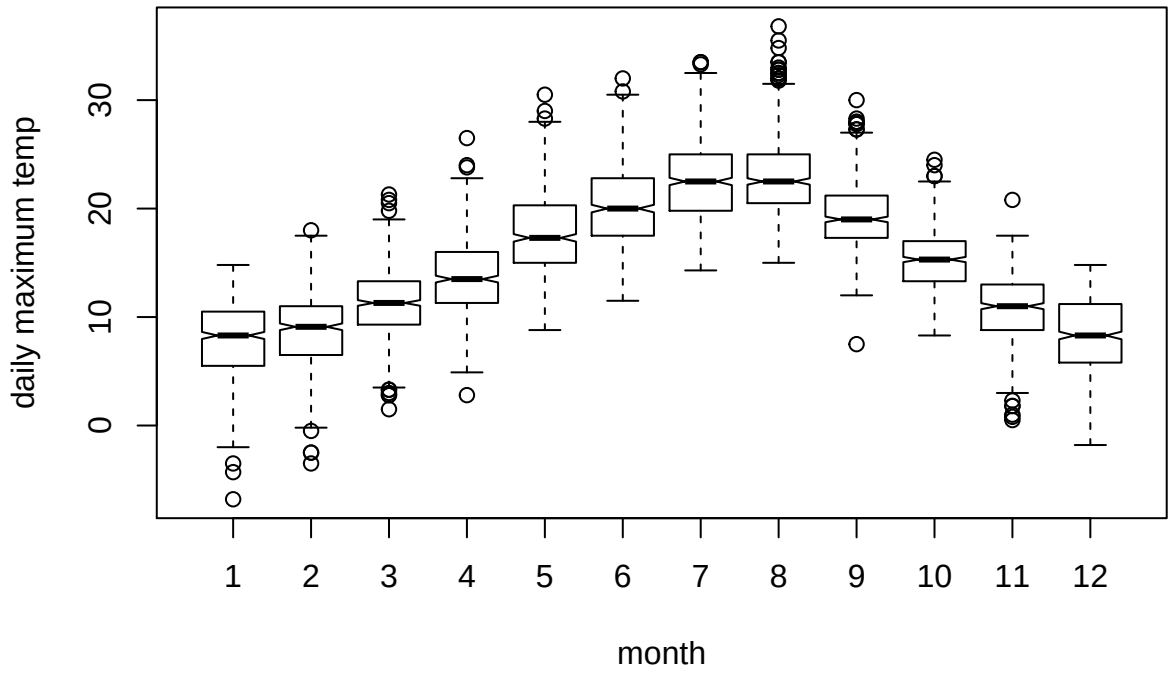
```
weather <- read.table("SilwoodWeather.txt", header = TRUE)
str(weather)
```

```
## 'data.frame':    6940 obs. of  5 variables:
##  $ upper: num  10.8 10.5 7.5 6.5 10 8 5.8 2.8 -0.8 1.5 ...
##  $ lower: num  6.5 4.5 -1 -3.3 5 3 -3.3 -5.5 -4.8 -1 ...
##  $ rain : num  12.2 1.3 0.1 1.1 3.5 0.1 0 0 0 0 ...
##  $ month: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ yr   : int  1987 1987 1987 1987 1987 1987 1987 1987 1987 1987 ...
```

```
attach(weather)
month <- factor(month)
plot(month, upper, ylab = "daily maximum temp", xlab = "month")  # default plot is box plot for categor
```




```
# notches: boxes which have overlapped notches have no significant different medians under appropriate
boxplot(upper ~ month, notch = TRUE, ylab = "daily maximum temp", xlab = "month")
```




```
detach(weather)
```

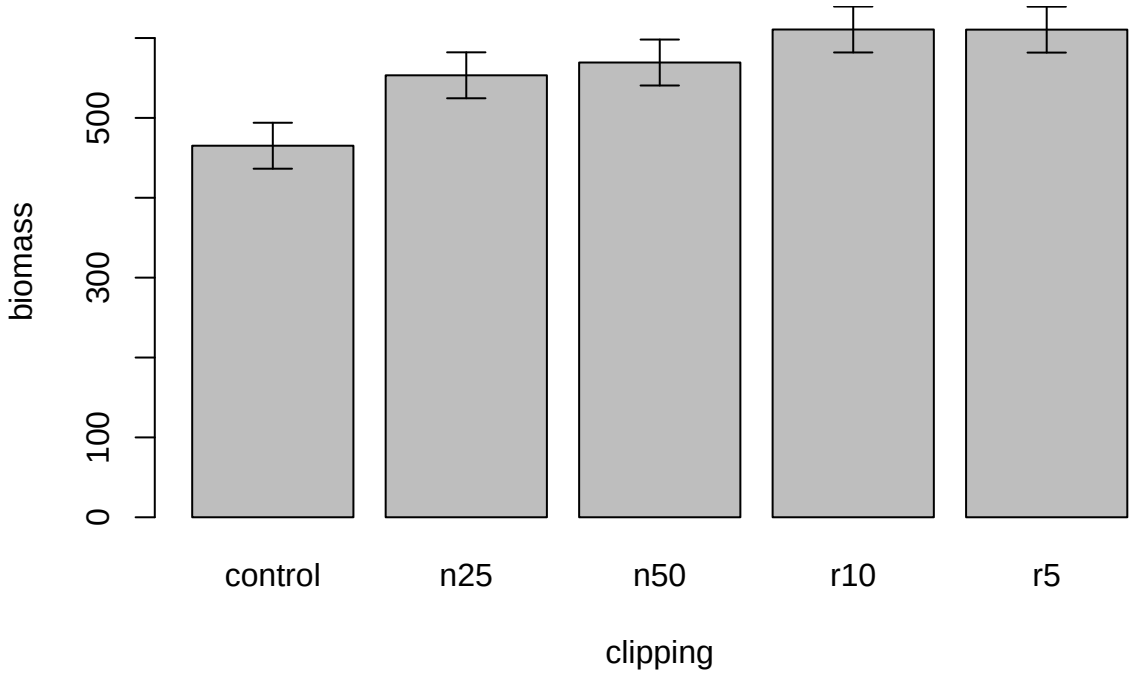
Barplots with error bars

- barplots with error bars showing the uncertainty , similar with Chapter 2.15
- barplots showing the histogram with the help of `table`

```
trial <- read.table("compexpt.txt",header=T)
attach(trial)
names(trial)
```

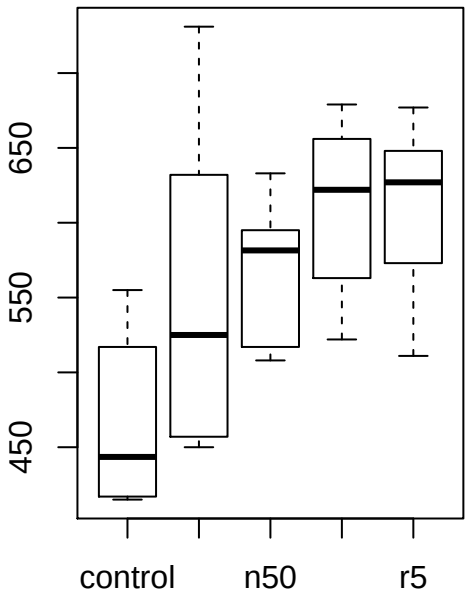
```
## [1] "biomass"  "clipping"
```

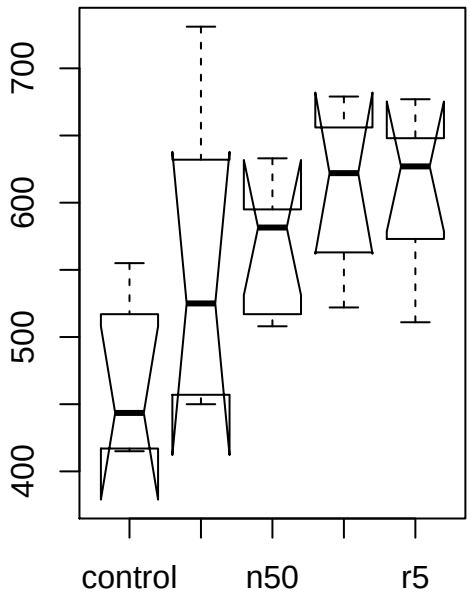
```
# function to draw error bars
seBars <- function(x,y){
model <- lm(y ~ factor(x))
reps <- length(y)/length(levels(x)) # replicates
sem <- summary(model)$sigma/sqrt(reps) # se
m <- as.vector(tapply(y,x,mean))
upper <- max(m) + sem
nn <- as.character(levels(x))
xs <- barplot(m,ylim=c(0,upper),names=nn,
            ylab=deparse(substitute(y)),xlab=deparse(substitute(x)))
for (i in 1:length(xs)) {
arrows(xs[i],m[i]+sem,xs[i],m[i]-sem,angle=90,code=3,length=0.1) } }

# Add error bars
seBars(clipping, biomass)
```



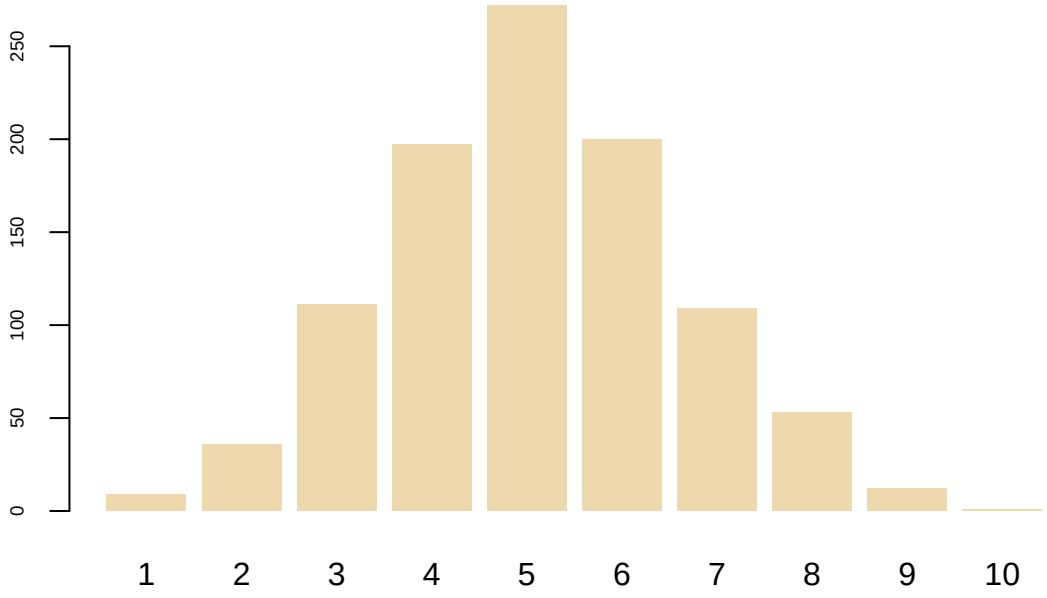

```
par(mfrow=c(1,2))
plot(clipping,biomass)
plot(clipping,biomass,notch=T)
```

```
## Warning in bxp(structure(list(stats = structure(c(415, 417, 443.5, 517, :
## some notches went outside hinges ('box'): maybe set notch=FALSE
```






```
detach(trial)

# produce barplots like "hist"
par(mfrow = c(1, 1))
pois <- rbinom(1000, 10, 0.5)
barplot(table(pois), border = NA, col = "wheat2", cex.axis = 0.6)
```



Plots for multiple comparisons

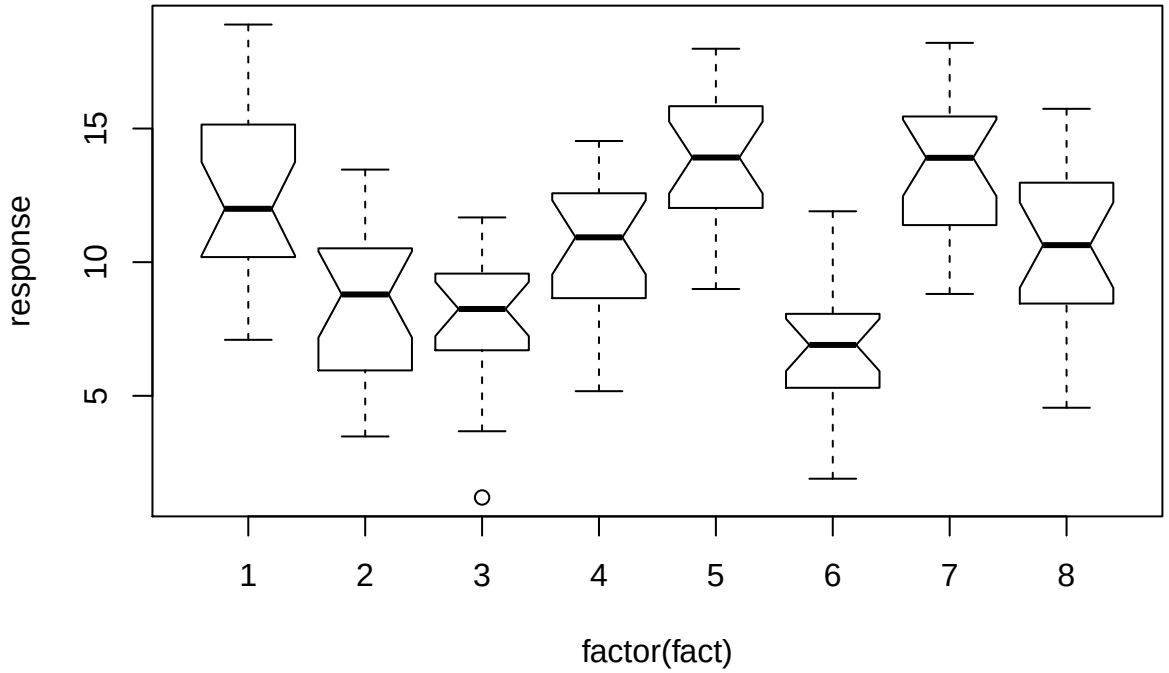
- boxplots with notches,
- Tukey's honest significant difference : the intervals do NOT overlap the vertical dashed line are significantly different.

```
data <- read.table("box.txt", header = TRUE)
attach(data)
```

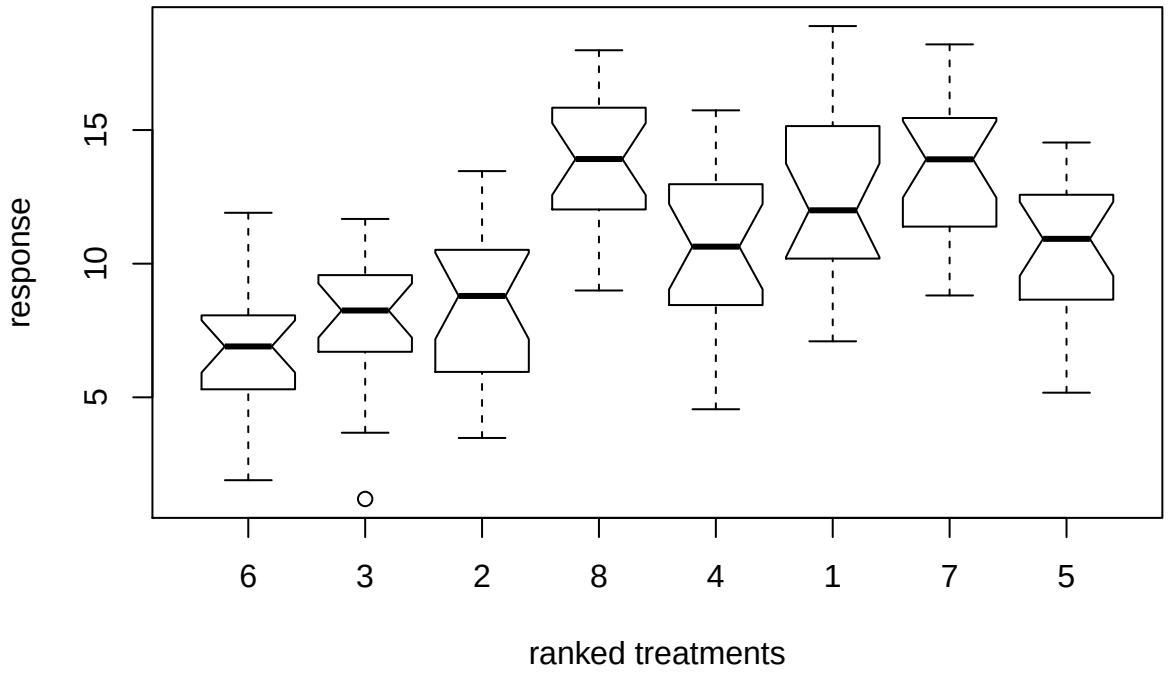
```
names(data)
```

```
## [1] "fact"     "response"
```

```
plot(response ~ factor(fact), notch = TRUE)
```



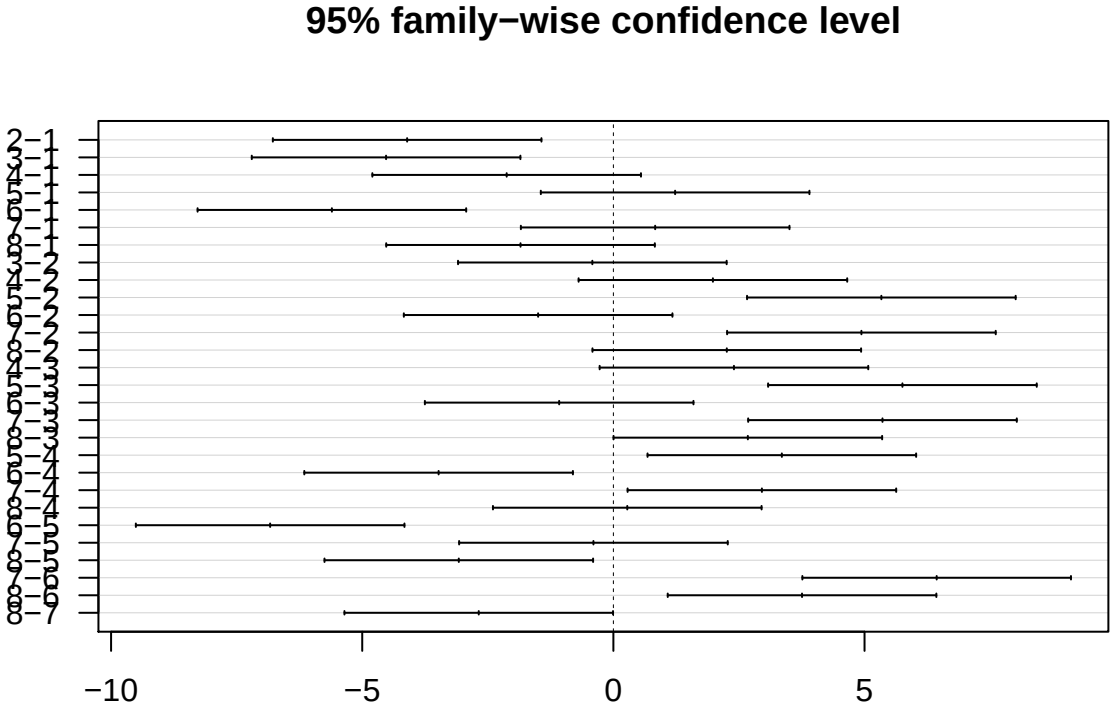

```
# or order the levels for better comparison
index <- order(tapply(response, fact, median))
ordered <- factor(rep(index, rep(20, 8)))
boxplot(response~ordered,notch=T,names=as.character(index),
  xlab="ranked treatments",ylab="response")
```



```
# or use Tukey's honest significant difference
model <- aov(response ~ factor(fact))
summary(model)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## factor(fact)   7  925.7  132.24   17.48 <2e-16 ***
## Residuals    152 1150.1    7.57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(TukeyHSD(model), las = 1)
```




```
# Create a set of confidence intervals on the differences between the means of the levels of
# a factor with the specified family-wise probability of coverage. The intervals are based
# on the Studentized range statistic, Tukey's 'Honest Significant Difference' method.
detach(data)
```

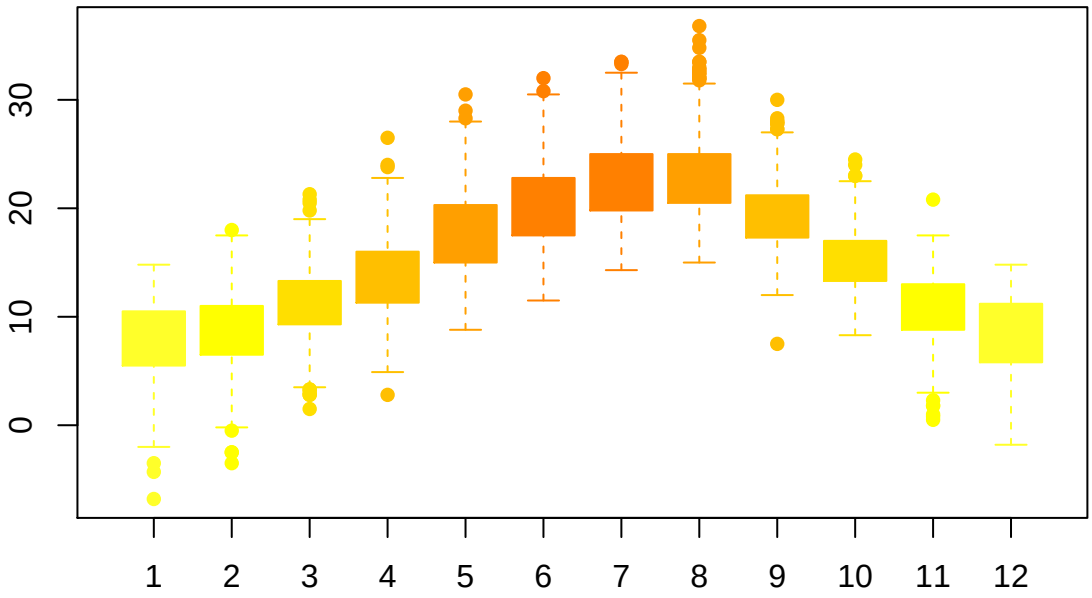
Using color palettes with categorical variables

```
data <- read.table("silwoodweather.txt", header = TRUE)
attach(data)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     month
```

```
month <- factor(month)
season <- heat.colors(12)
```

```
temp1 <- c(rev(5:10), 5:10)
plot(month, upper, col = season[temp1], border = season[temp1], pch = 16)
```




```
detach(data)
```

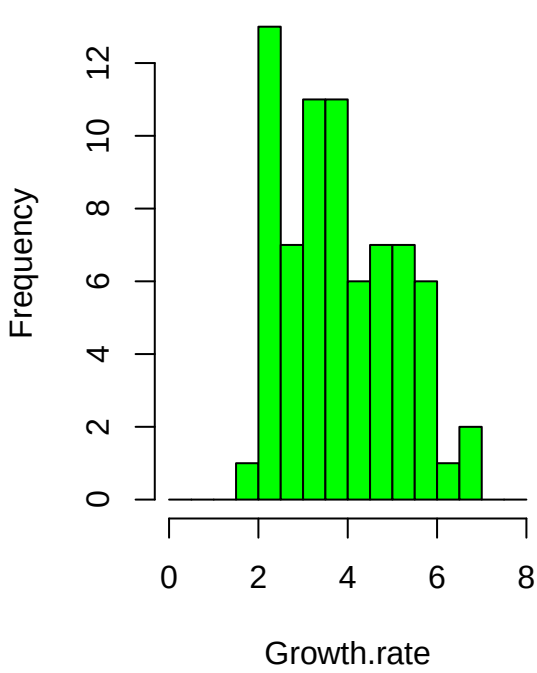
Plots for single samples

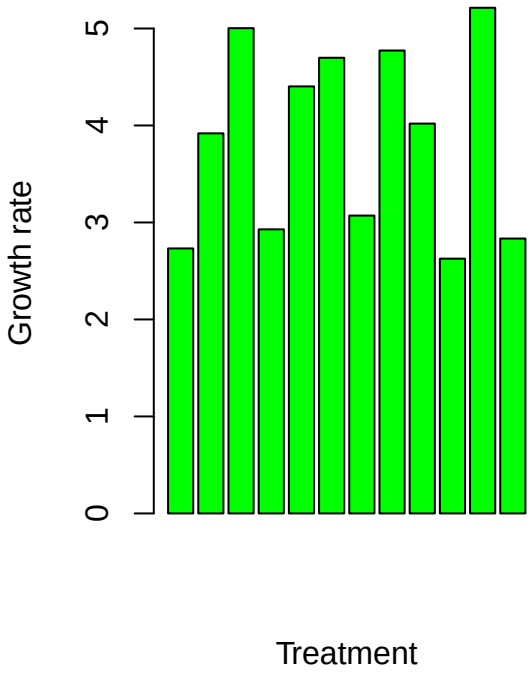
- `hist` for frequency distribution
- `plot` for values in the sequence
- Index plot by `plot` with a single variable
- `plot.ts` or `ts.plot` for time series plots
- `pie` for compositional plots like pie diagrams
- `stripchart` for data with sample size too small that boxplot is not suitable

```
data <- read.table("daphnia.txt", header = TRUE)
attach(data)
names(data)
```

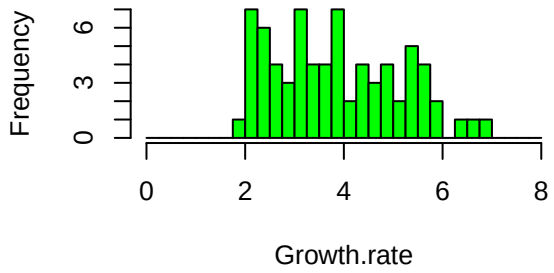
```
## [1] "Growth.rate" "Water"       "Detergent"   "Daphnia"
```

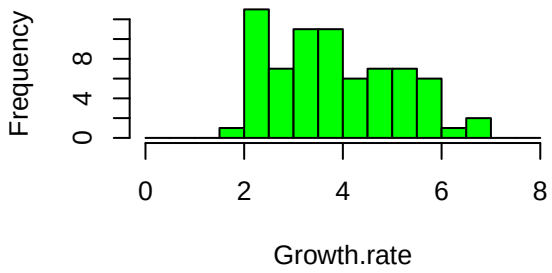
```
# difference of historgram and barplot
par(mfrow = c(1, 2))
hist(Growth.rate, seq(0, 8, 0.5), col = "green", main = "")

y <- as.vector(tapply(Growth.rate, list(Daphnia, Detergent), mean))
barplot(y, col = "green", ylab = "Growth rate", xlab = "Treatment")
```

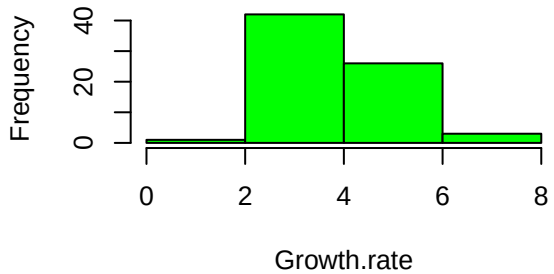


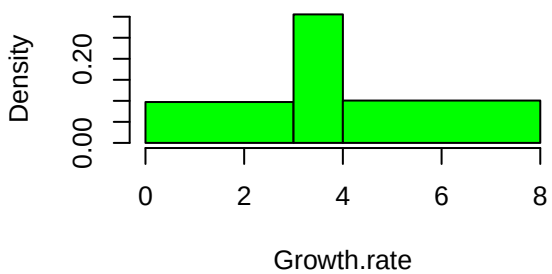



```
# the effects of bins

par(mfrow=c(2,2))
hist(Growth.rate,seq(0,8,0.25),col="green",main="")
hist(Growth.rate,seq(0,8,0.5),col="green",main="")
hist(Growth.rate,seq(0,8,2),col="green",main="")
hist(Growth.rate,c(0,3,4,8),col="green",main="")
```










```
# reproduce the fourth plot by using cut
edges <- c(0,3,4,8)
bin <- cut(Growth.rate,edges)
```

```
bin
```

```
##  [1] (0,3] (0,3] (3,4] (0,3] (3,4] (4,8] (4,8] (3,4] (4,8] (0,3] (3,4]
## [12] (0,3] (3,4] (4,8] (4,8] (4,8] (4,8] (4,8] (0,3] (3,4] (3,4] (3,4]
## [23] (3,4] (3,4] (3,4] (4,8] (4,8] (0,3] (0,3] (3,4] (3,4] (4,8] (4,8]
## [34] (0,3] (3,4] (4,8] (0,3] (0,3] (3,4] (3,4] (3,4] (4,8] (4,8] (4,8]
## [45] (4,8] (3,4] (0,3] (3,4] (4,8] (4,8] (4,8] (3,4] (4,8] (4,8] (0,3]
## [56] (3,4] (0,3] (4,8] (4,8] (4,8] (0,3] (3,4] (4,8] (0,3] (0,3] (0,3]
## [67] (4,8] (4,8] (4,8] (0,3] (0,3] (0,3]
## Levels: (0,3] (3,4] (4,8]
```

```
is.factor(bin)
```

```
## [1] TRUE
```

```
table(bin)
```

```
## bin
## (0,3] (3,4] (4,8]
##    21    22    29
```

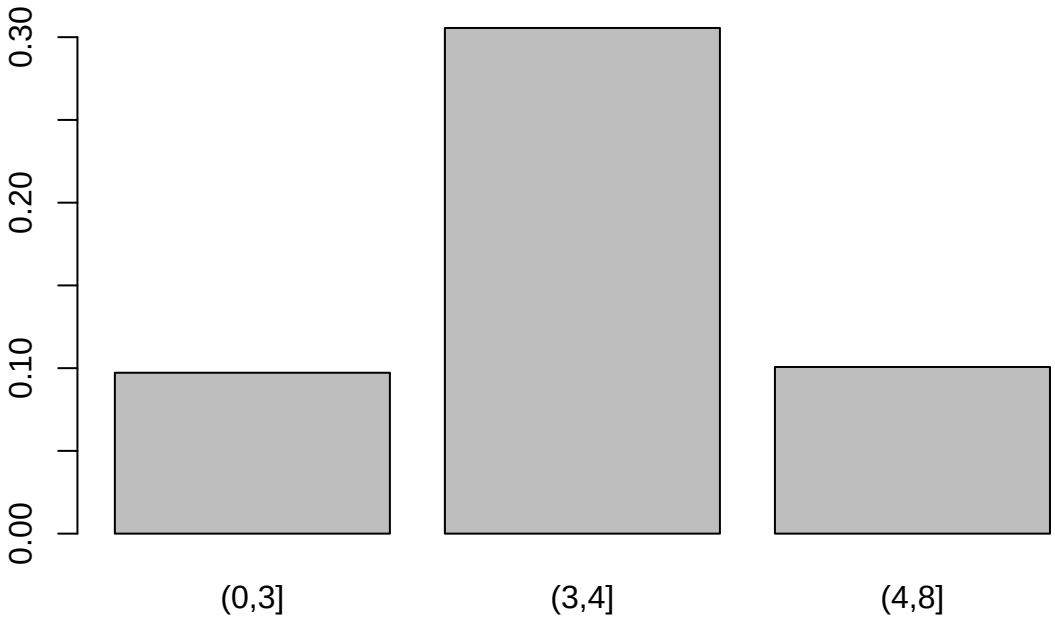
```
diff(edges) # Returns suitably lagged and iterated differences
```

```
## [1] 3 1 4
```

```
(table(bin)/sum(table(bin)))/diff(edges)
```

```
## bin
##      (0,3]      (3,4]      (4,8]
## 0.09722222 0.30555556 0.10069444
```

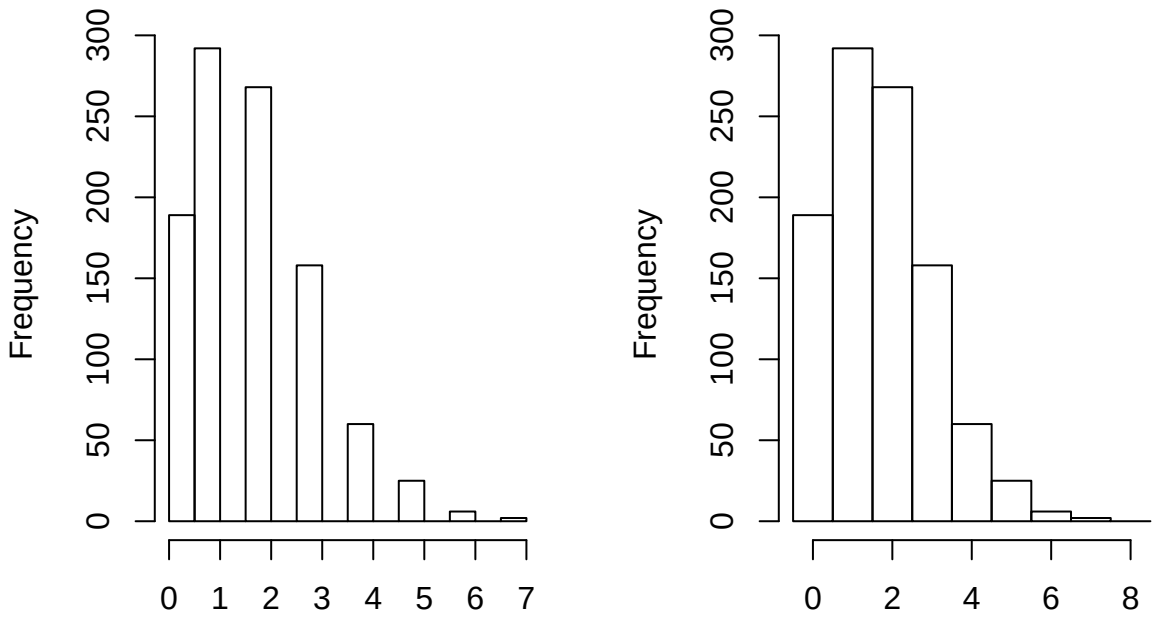
```
par(mfrow = c(1, 1))
barplot((table(bin)/sum(table(bin)))/diff(edges))
```




```
# These are the heights of the three bars in the density plot (bottom right, above).
# They do not add to 1 because the bars are of different widths. It is the total
# area of the three bars that is 1 under this convention.
detach(data)
```

Histograms of integers

```
values <- rpois(1000,1.70)
par(mfrow = c(1, 2))
# before specifying bins
hist(values,main="",xlab="random numbers from a Poisson with mean 1.7")

# after specifying bins to capture frequency of 0
hist(values,breaks = (-0.5:8.5), main="",
            xlab="random numbers from a Poisson with mean 1.7")
```

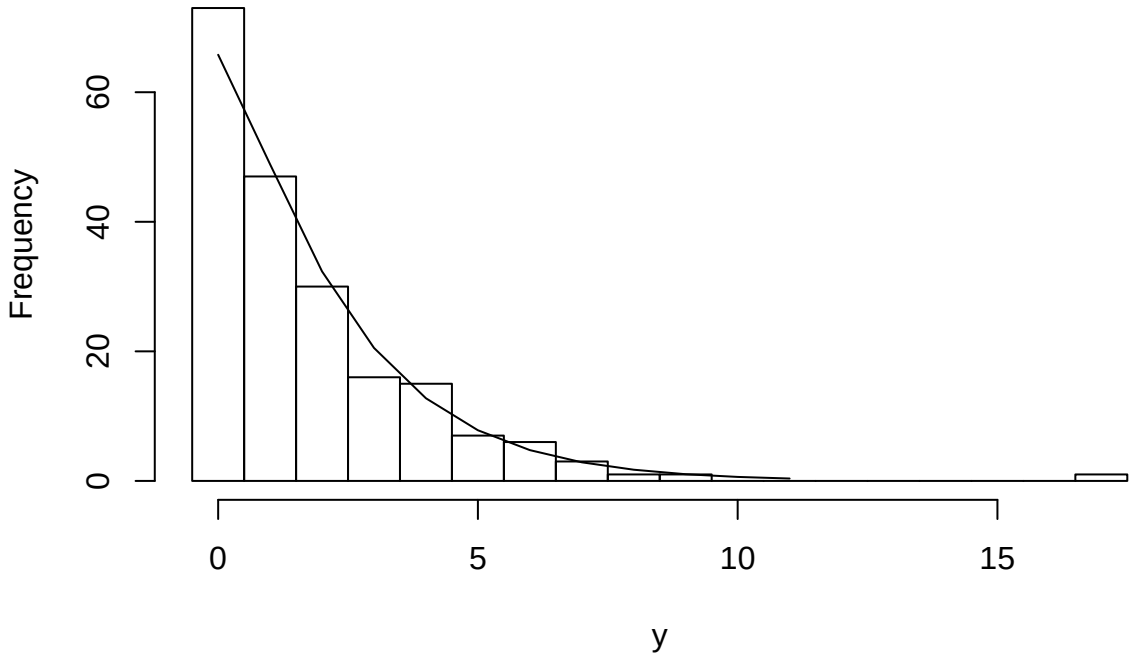



```
par(mfrow = c(1, 1))
```

Overlaying histograms with smooth density functions

- Using `lines` to add estimated lines
- Using `density` function for continuous variables

```
# add lines by "lines"
y <- rnbinom(200, mu = 1.5, size = 1)
bks <-  -0.5:(max(y) + 0.5)
hist(y, bks, main="")
xs <- 0:11
ys <- dnbinom(xs, size=1.2788, mu=1.772)
lines(xs, ys*200)
```




```
# add lines by "density"
library(MASS)
attach(faithful)

# rule of thumb for bankwidth
(max(eruptions)-min(eruptions))/(2*(1+log(length(eruptions),base=2)))
```

```
## [1] 0.192573
```

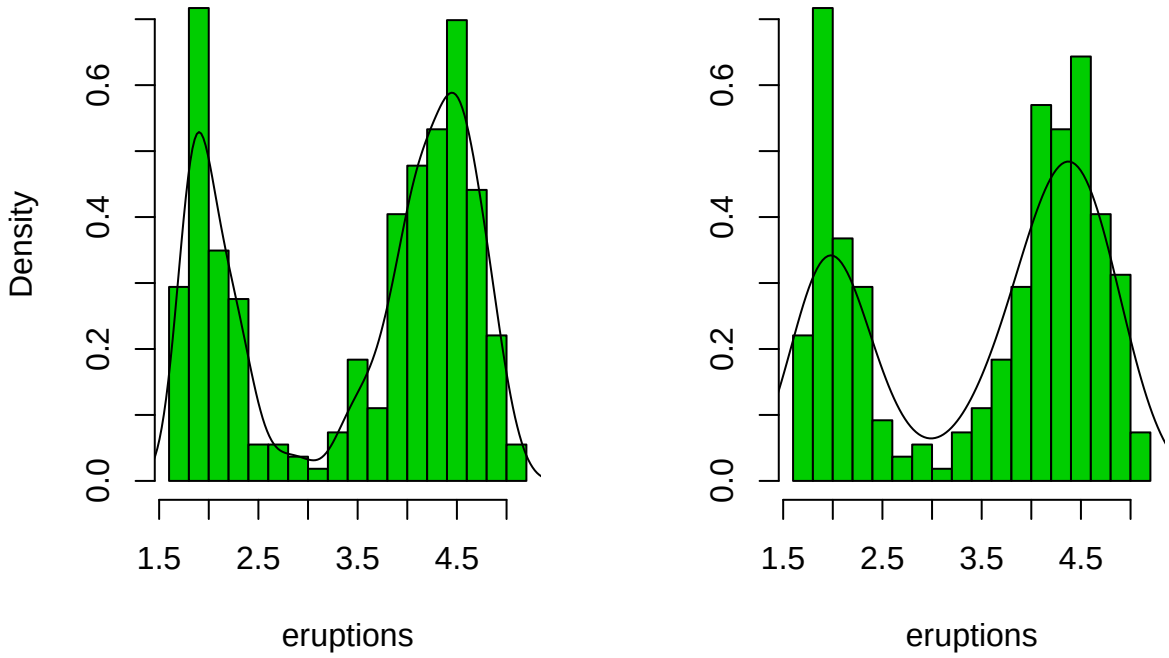
```
# the range
range(eruptions)[2] -range(eruptions)[1]
```

```
## [1] 3.5
```

```
# the ideal number of bins
(range(eruptions)[2] -range(eruptions)[1]) /((max(eruptions)-min(eruptions))/(2*(1+log(length(eruptions)
))
```

```
## [1] 18.17493
```

```
par(mfrow = c(1, 2))
hist(eruptions, 15, freq = FALSE, main = "", col = 27)
lines(density(eruptions, width = 0.6, n = 200)) # specify width = 0.6
truehist(eruptions, nbins = 15, col = 27)
lines(density(eruptions, n = 200))
```



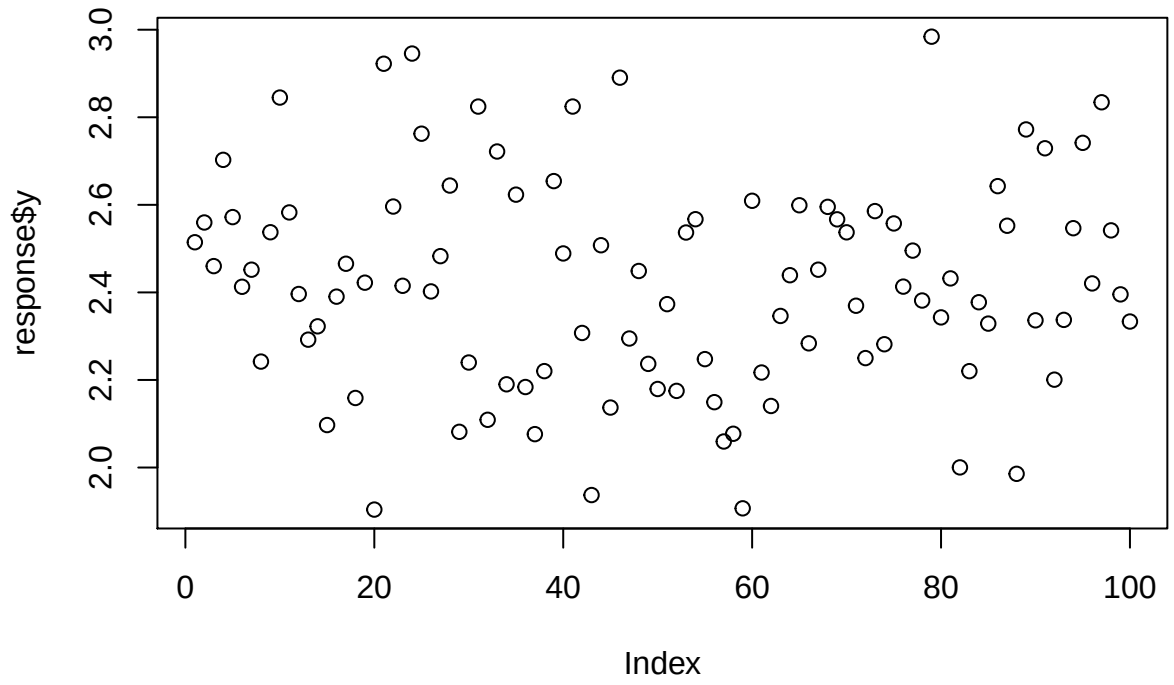

```
par(mfrow = c(1, 1))
detach(faithful)
```

- There are 18 bins even we asked for 15 bins.
- The hists are not extactly the same even for the same data
- The density curve is better when specifying bandwidth = 0.6

Index plot/ Trace plot

Useful for error checking

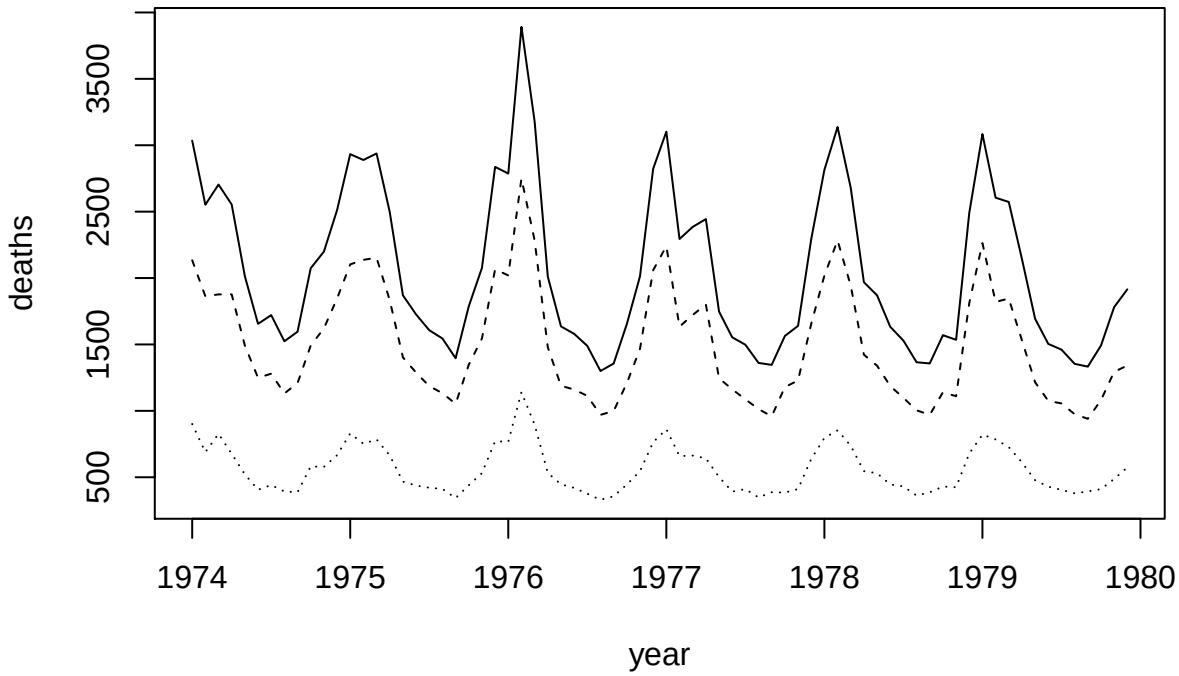
```
response <- read.table("das.txt", header = TRUE)
plot(response$y)
```



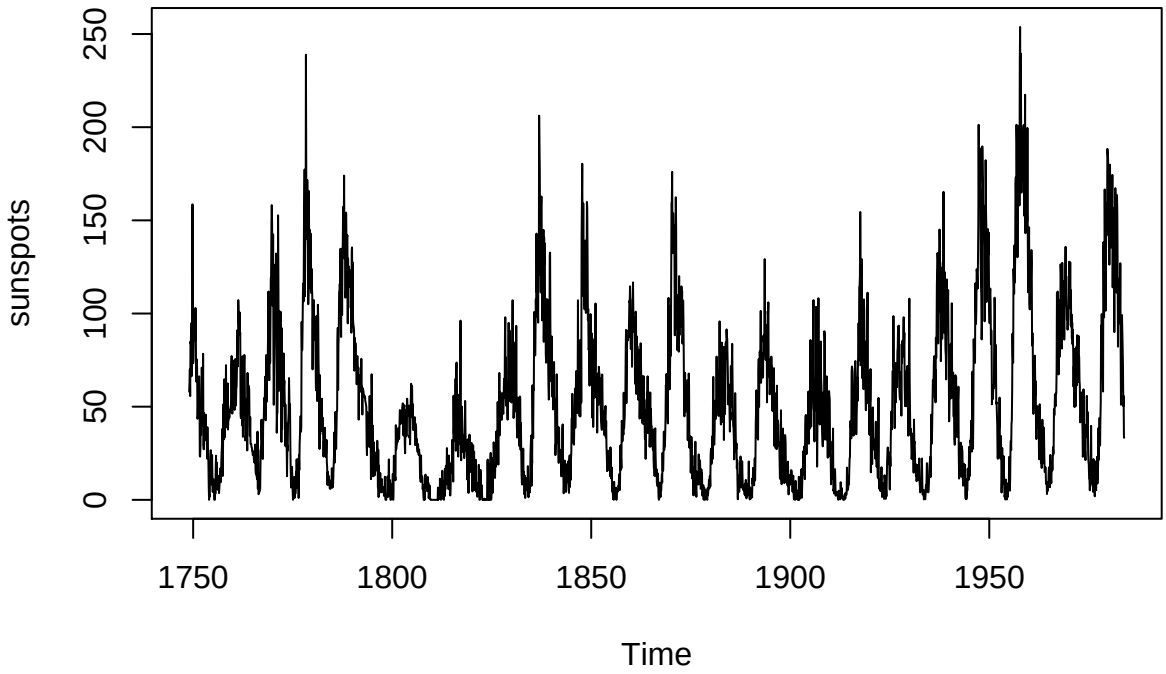
Time series plot

- ts.plot: works for plotting objects inheriting from class = ts

```
# ts.plot
data(UKLungDeaths)
# ts.plot plots several time series on a common plot.
ts.plot(ldeaths, mdeaths, fdeaths, xlab="year", ylab="deaths", lty=c(1:3))
```



```
# produce three time series on the same axes using different line types


# plot.ts
# works for plotting objects inheriting from class = ts
data(sunspots)
plot(sunspots)
```




```
class(sunspots)
```

```
## [1] "ts"
```

```
is.ts(sunspots)
```

```
## [1] TRUE
```

Pie charts

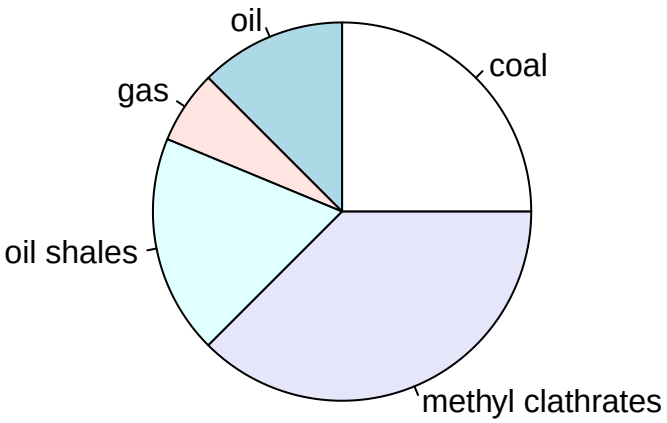
Useful to illustrate the proportional make-up of a sample in presentations.

```
data <- read.csv("piedata.csv")
data
```

```
##               names amounts
## 1              coal       4
## 2               oil       2
## 3               gas       1
## 4        oil shales       3
## 5 methyl clathrates       6
```

```
# pie(x, labels = names(x), edges = 200, radius = 0.8,
#    clockwise = FALSE, init.angle = if(clockwise) 90 else 0,
#    density = NULL, angle = 45, col = NULL, border = NULL,
#    lty = NULL, main = NULL, ...)
```

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



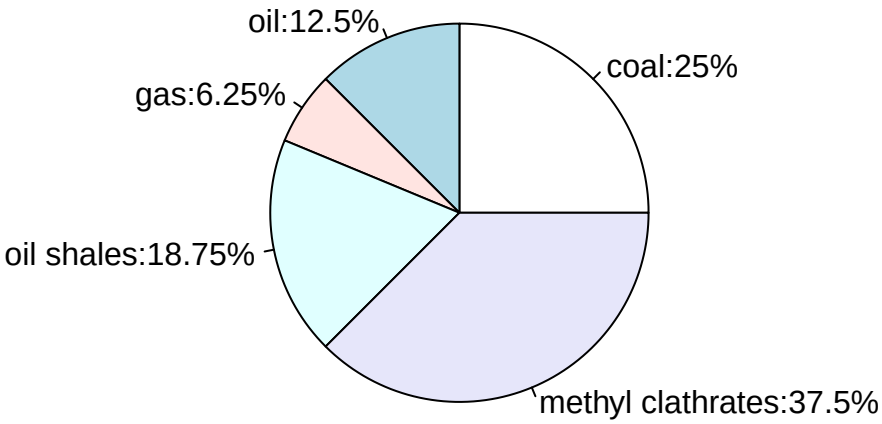

```
data$amounts
```

```
## [1] 4 2 1 3 6
```

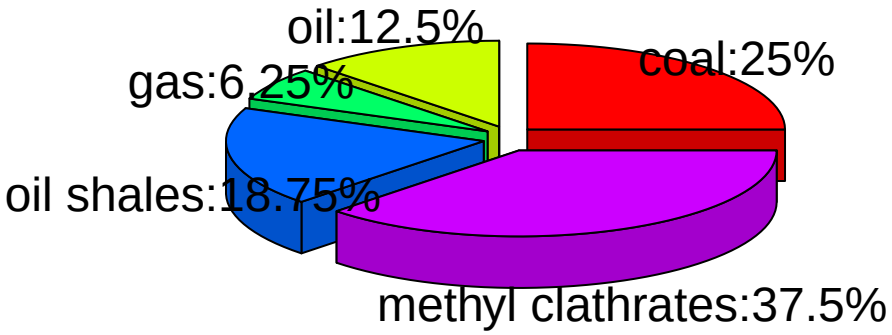
```
sum(data$amounts)
```

```
## [1] 16
```

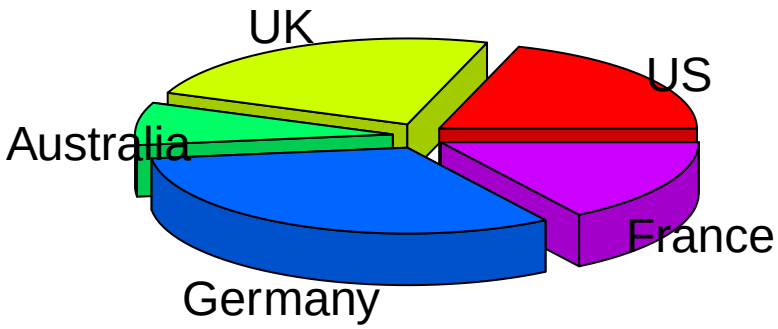
```
pct <- round(data$amounts / sum(data$amounts) *100, 2)
lbs <- paste(data$names, ":", pct, "%", sep = "")
pie(data$amounts, labels = lbs)
```



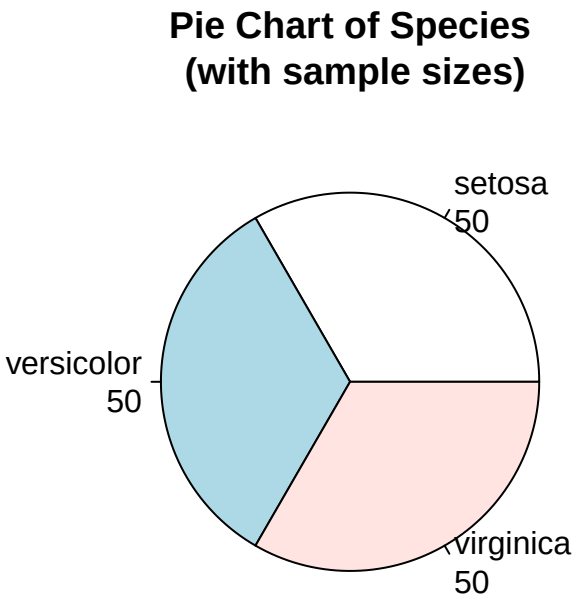

```
# 3D Exploded Pie Chart
library(plotrix)

pie3D(data$amounts, labels = lbs, explode = 0.1)
```




```
slices <- c(10, 12, 4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pie3D(slices,labels=lbls,explode=0.1,
    main="Pie Chart of Countries ")
```



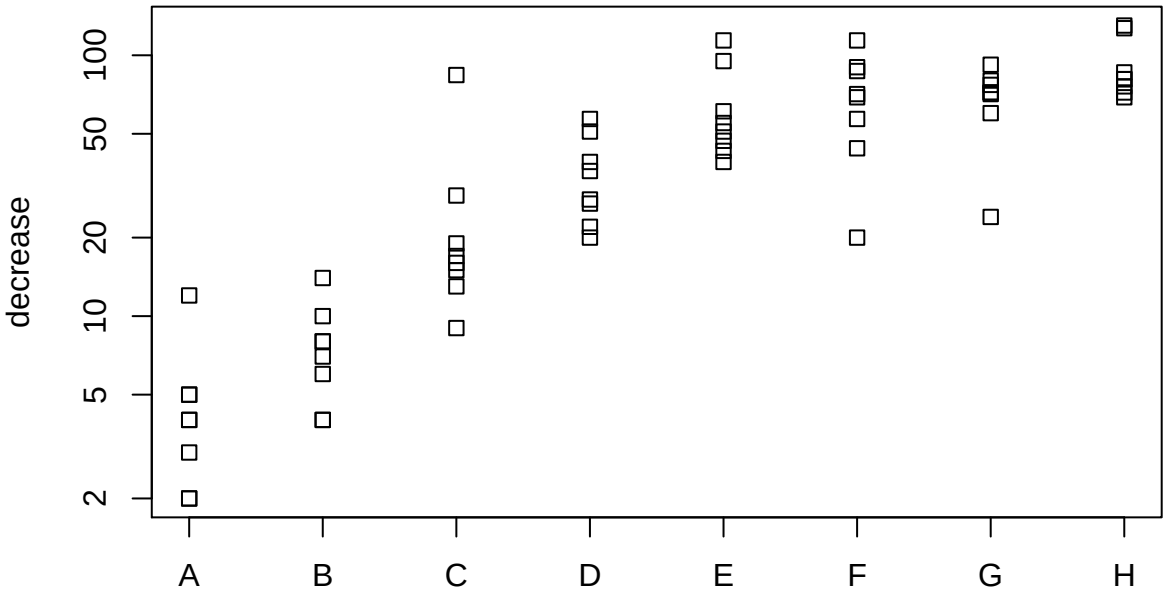

```
# Pie Chart from data frame with Appended Sample Sizes
mytable <- table(iris$Species)
lbls <- paste(names(mytable), "\n", mytable, sep="")
pie(mytable, labels = lbls,
    main="Pie Chart of Species\n (with sample sizes)")
```



stripchart function

`stripchart` produces one dimensional scatter plots (or dot plots) of the given data. These plots are a good alternative to boxplots when sample sizes are small.

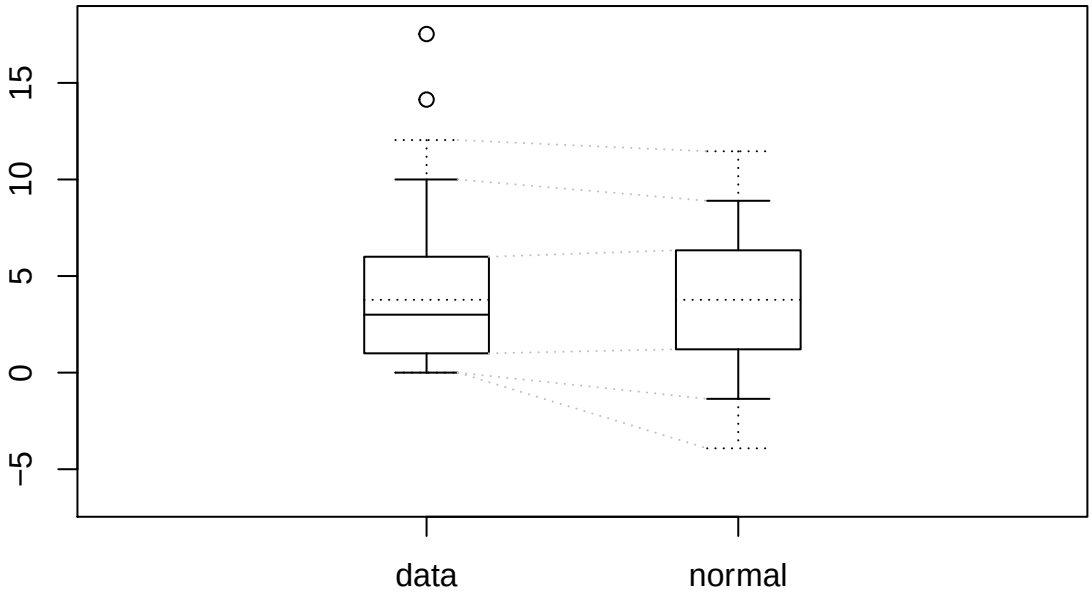
```
data("OrchardSprays")
with(OrchardSprays,
     stripchart(decrease ~ treatment,
                ylab = "decrease", xlab = "",
                vertical = TRUE, log = "y"))
```



A plot to test normality

```
# A function that plots a data set and compares it to a plot of normally distributed
# data with the same mean and sd

normal.plot <- function(y) {
s <- sd(y)
plot(c(0,3),c(min(0, mean(y)-s * 4 * qnorm(0.75)),max(y)),xaxt="n",xlab="",type="n",ylab="")
#   for your data's boxes and whiskers, centred at x = 1
top <- quantile(y,0.75)
bottom <- quantile(y,0.25)
w1u <- quantile(y,0.91)
w2u <- quantile(y,0.98)
w1d <- quantile(y,0.09)
w2d <- quantile(y,0.02)
rect(0.8,bottom,1.2,top)
lines(c(0.8,1.2),c(mean(y),mean(y)),lty=3)
lines(c(0.8,1.2),c(median(y),median(y)))
lines(c(1,1),c(top,w1u))
lines(c(0.9,1.1),c(w1u,w1u))
lines(c(1,1),c(w2u,w1u),lty=3)
lines(c(0.9,1.1),c(w2u,w2u),lty=3)
nou <- length(y[y>w2u])
points(rep(1,nou),jitter(y[y>w2u]))
lines(c(1,1),c(bottom,w1d))
lines(c(0.9,1.1),c(w1d,w1d))
lines(c(1,1),c(w2d,w1d),lty=3)
lines(c(0.9,1.1),c(w2d,w2d),lty=3)
nod <- length(y[y<w2d])
points(rep(1,nod),jitter(y[y<w2d]))
#for the normal box and whiskers, centred at x = 2
n75 <- mean(y)+ s * qnorm(0.75)
n25 <- mean(y)- s * qnorm(0.75)
n91 <- mean(y)+ s * 2* qnorm(0.75)
n98 <- mean(y)+ s * 3* qnorm(0.75)
n9 <- mean(y)- s * 2* qnorm(0.75)
n2 <- mean(y)- s * 3* qnorm(0.75)
rect(1.8,n25,2.2,n75)
lines(c(1.8,2.2),c(mean(y),mean(y)),lty=3)
lines(c(2,2),c(n75,n91))
lines(c(1.9,2.1),c(n91,n91))
lines(c(2,2),c(n98,n91),lty=3)
lines(c(1.9,2.1),c(n98,n98),lty=3)
lines(c(2,2),c(n25,n9))
lines(c(1.9,2.1),c(n9,n9))
lines(c(2,2),c(n9,n2),lty=3)
lines(c(1.9,2.1),c(n2,n2),lty=3)
lines(c(1.2,1.8),c(top,n75),lty=3,col="gray")
lines(c(1.1,1.9),c(w1u,n91),lty=3,col="gray")
lines(c(1.1,1.9),c(w2u,n98),lty=3,col="gray")
lines(c(1.2,1.8),c(bottom,n25),lty=3,col="gray")
lines(c(1.1,1.9),c(w1d,n9),lty=3,col="gray")
lines(c(1.1,1.9),c(w2d,n2),lty=3,col="gray")
```

```r
# label the two boxes
axis(1,c(1,2),c("data","normal")) }


y <- rnbinom(100,1,0.2)

normal.plot(y)
```



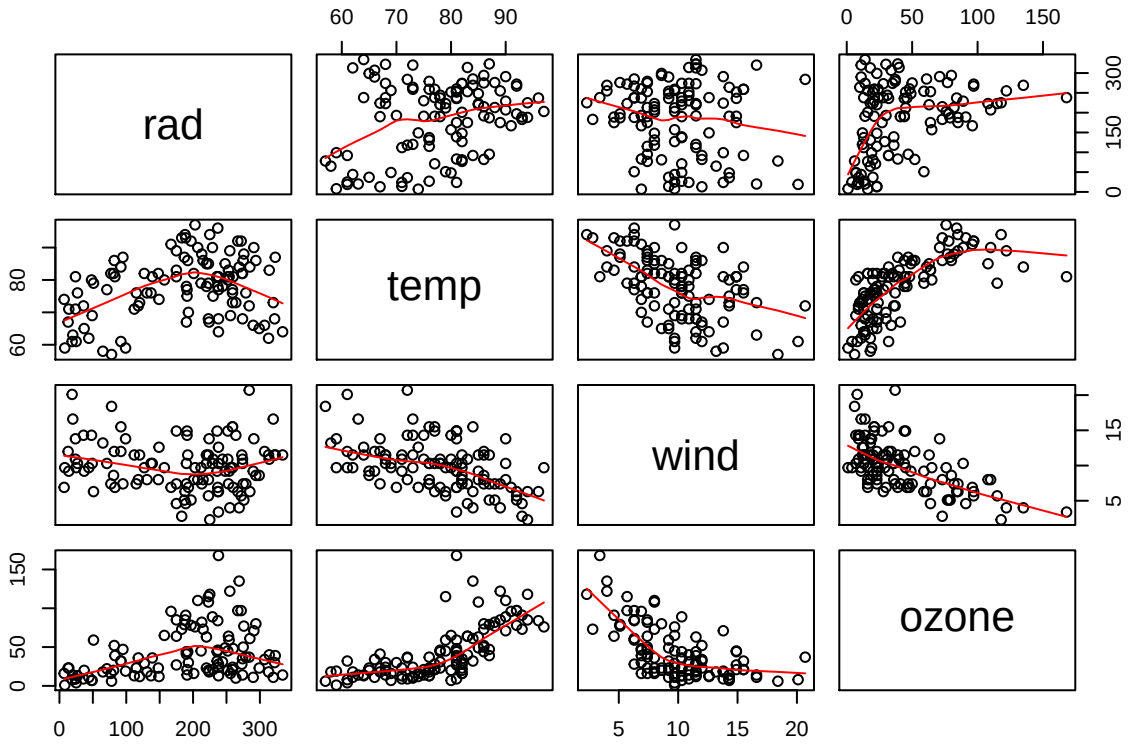
Plot with multiple variables

- `pairs` for a scatter matrix of numeric variables
- `coplot` for y ~ x for different z values
- `xyplot` for a set of panel plots

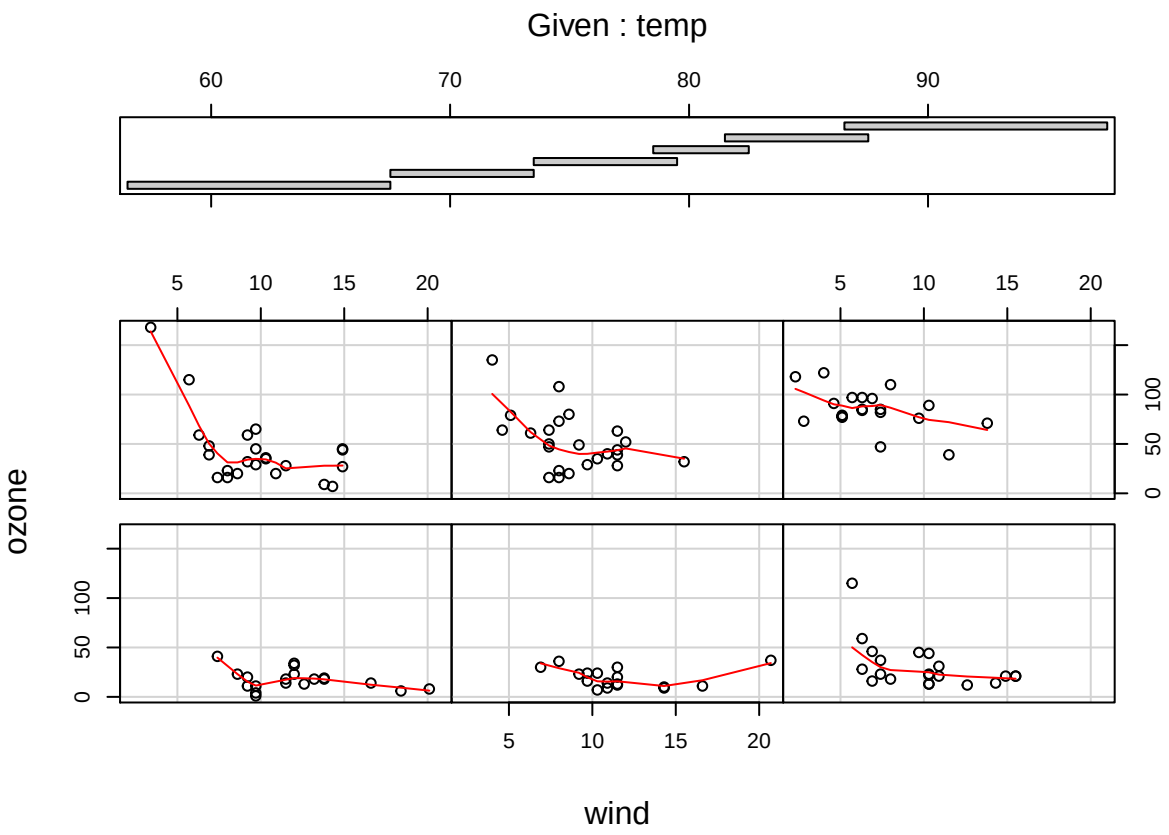
```r
ozonedata <- read.table("ozone.data.txt", header = TRUE)
attach(ozonedata)
names(ozonedata)
```

```
## [1] "rad"   "temp"  "wind"  "ozone"
```

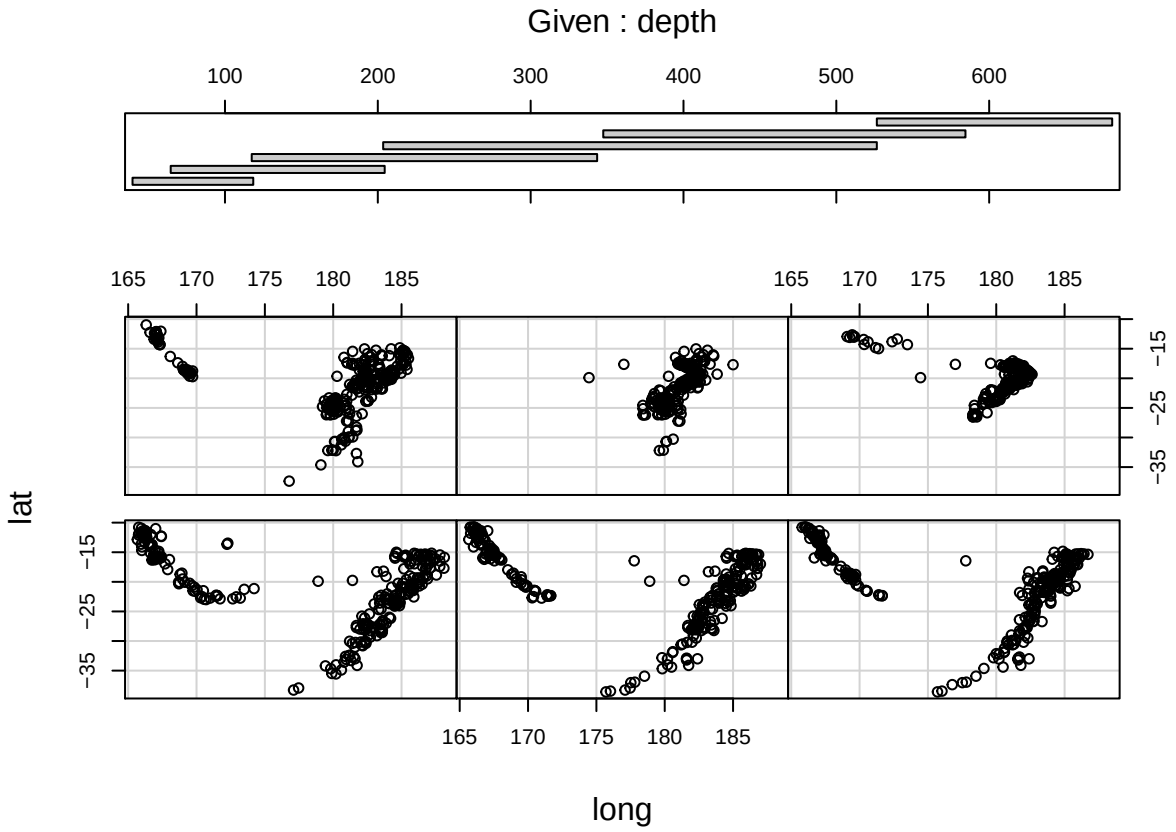
```r
pairs(ozonedata, panel = panel.smooth)
```



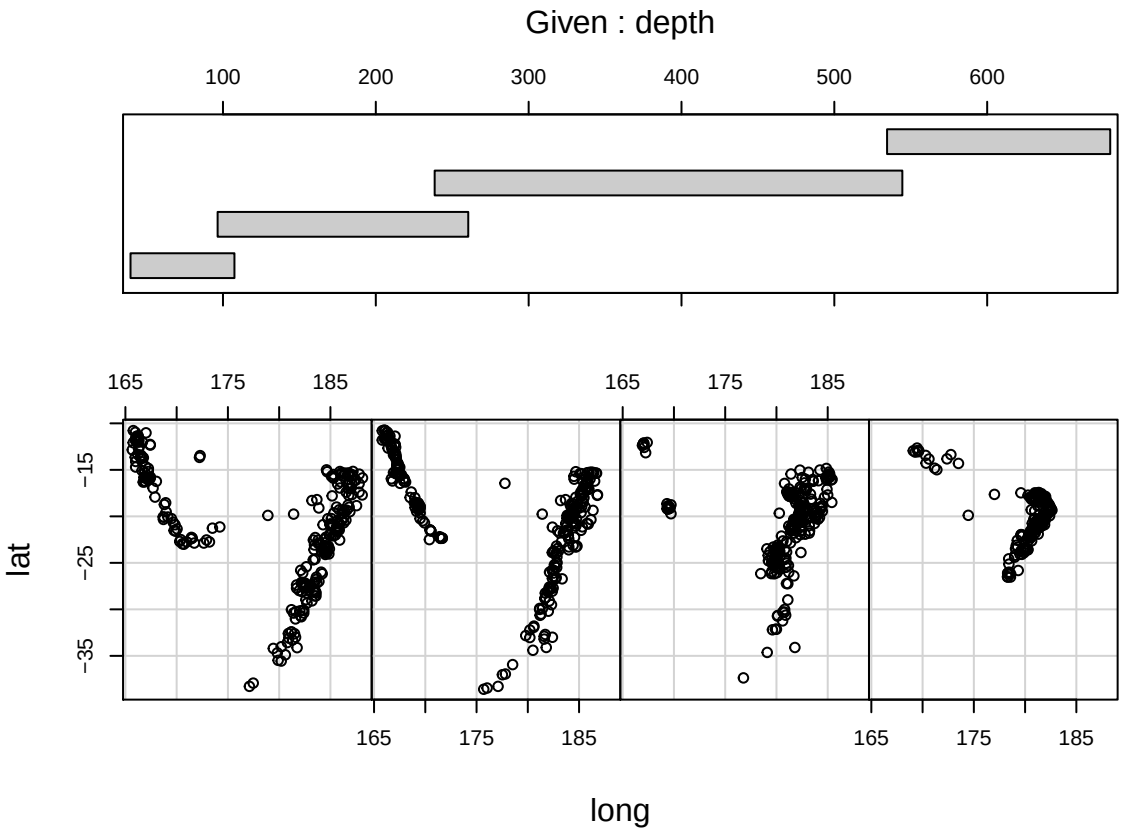

```
# coplot
coplot(ozone ~ wind|temp, panel = panel.smooth, overlap = -0.05)
```



```
## Tonga Trench Earthquakes
coplot(lat ~ long | depth, data = quakes)
```



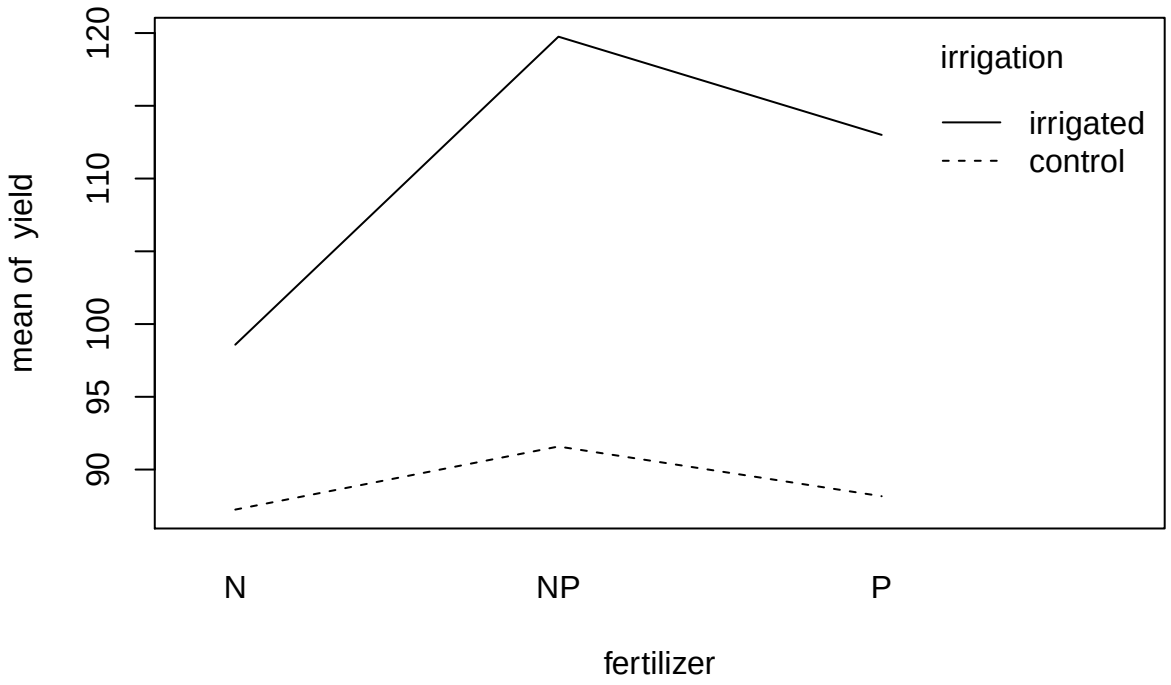

```
given.depth <- co.intervals(quakes$depth, number = 4, overlap = .1)
coplot(lat ~ long | depth, data = quakes, given.v = given.depth, rows = 1)
```




```
# ordered from lowerleft to upperright with corresponding temp shown in the upper panel
# overlap can be adjusted to control the data points in each panel
detach(ozonedata)

# Interaction plots by "interaction.plot"
yields <- read.table("splityield.txt", header = TRUE)
attach(yields)
names(yields)
```

```
## [1] "yield"      "block"      "irrigation" "density"    "fertilizer"
```

```
interaction.plot(fertilizer, irrigation, yield)
```



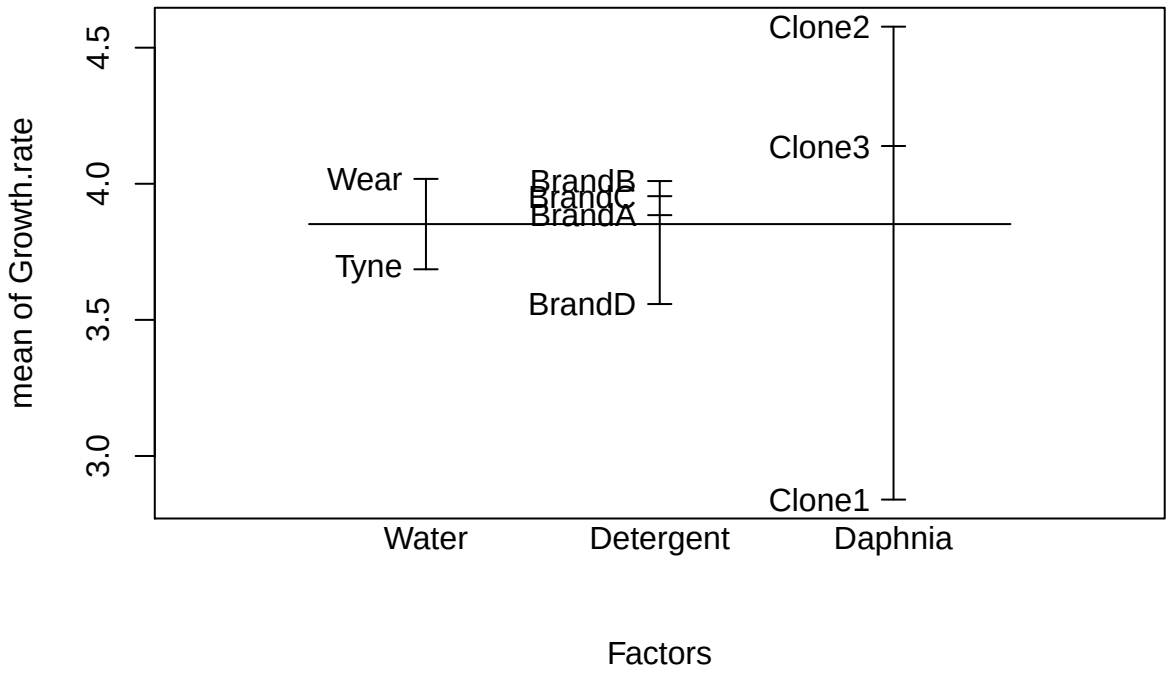

```
# interaction.plot(x.factor, trace.factor, response)
detach(yields)
```

Special plots

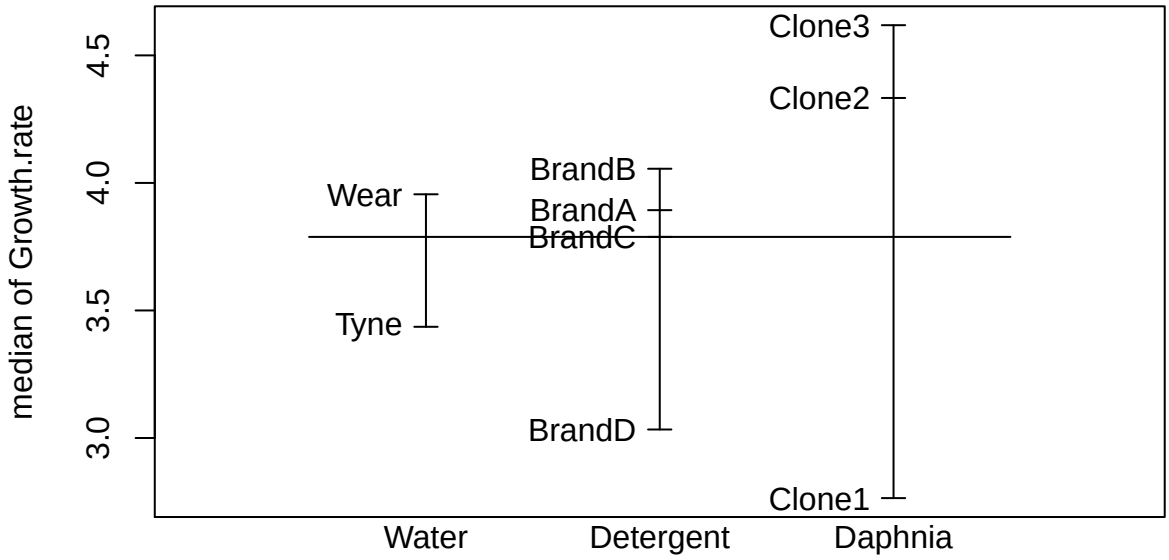
- `design plots` for visualizing effect sizes in designed experiments
- `bubble plots` for ilustrating a third variable across different locations in the x-y plane
- plots with many **identical** values

1. use `jitter` to add a small noise to break ties
2. use `sunflowerplot`

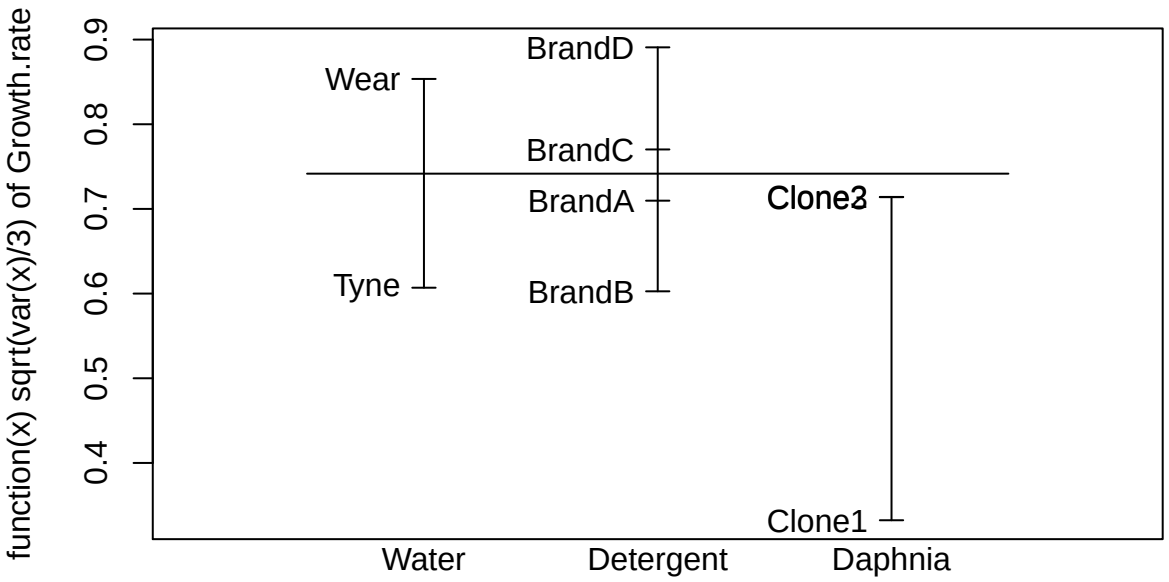
```
# design plots  with default means plot
data <- read.table("daphnia.txt", header = TRUE)
attach(data)
plot.design(Growth.rate ~ Water*Detergent*Daphnia)
```




```
# median plot
plot.design(Growth.rate ~ Water*Detergent*Daphnia, fun = median)
```



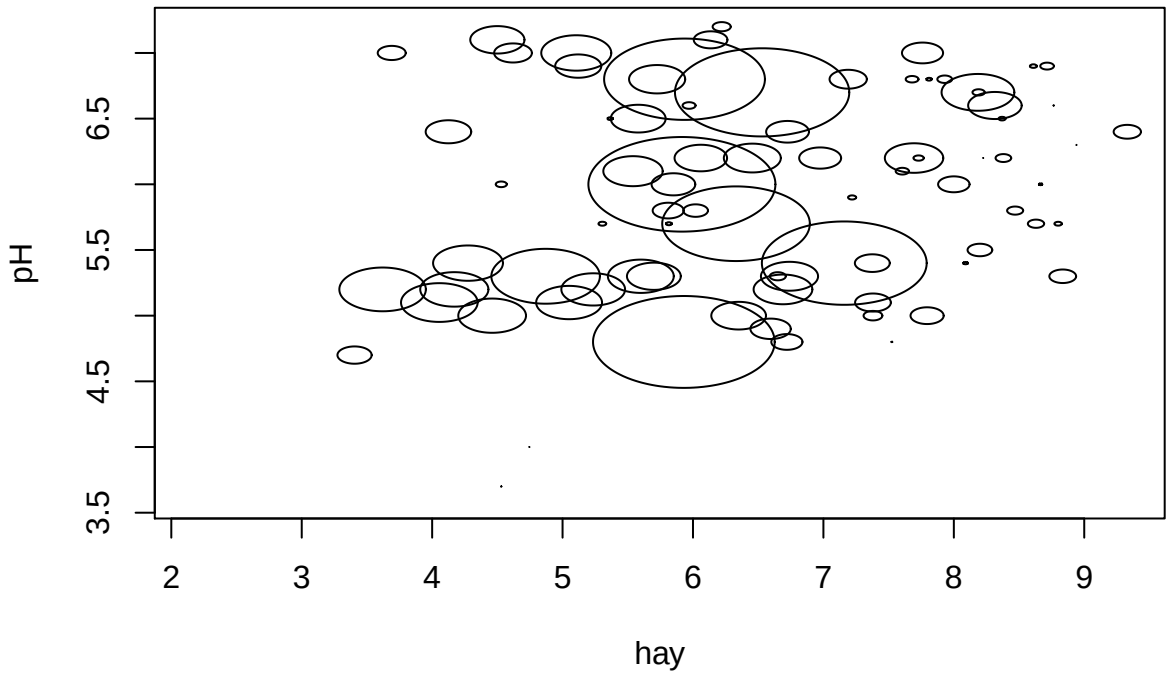

```
# supply anonymous function to plot
plot.design(Growth.rate ~ Water*Detergent*Daphnia, fun = function(x) sqrt(var(x)/3))
```




```
detach(data)

# bubble plot
# Function to plot the bubble plot centered at the points and with radius corresponding to the third va
# xv: x axes points
# yv: y axes points
# rv: a third variable related to the radius of the bubbles

bubble.plot <- function(xv,yv,rv,bs=0.1){
      r <- rv/max(rv)
      yscale <- max(yv)-min(yv)
      xscale <- max(xv)-min(xv)
      plot(xv,yv,type="n", xlab=deparse(substitute(xv)), ylab=deparse(substitute(yv)))
for (i in 1:length(xv)) bubble(xv[i],yv[i],r[i],bs,xscale,yscale) }

# function to plot the circles
bubble <- function (x, y, r, bubble.size, xscale, yscale) {
theta <- seq(0,2*pi, pi/200)
yv <- r*sin(theta)*bubble.size*yscale # a set of points
xv <- r*cos(theta)* bubble.size*xscale
lines(x + xv,y + yv)
}

# test it
ddd <- read.table("pgr.txt", header = TRUE)
attach(ddd)
names(ddd)
```

```
## [1] "FR"  "hay" "pH"
```

```
bubble.plot(hay, pH, FR)
```

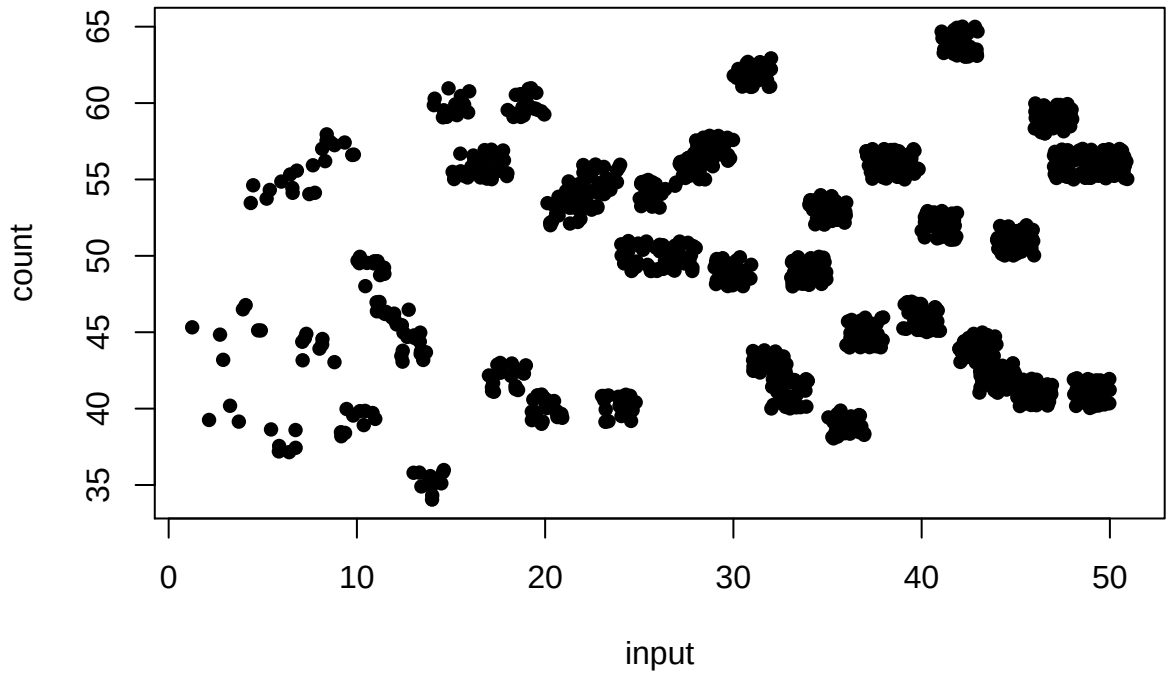



```
detach(ddd)
```

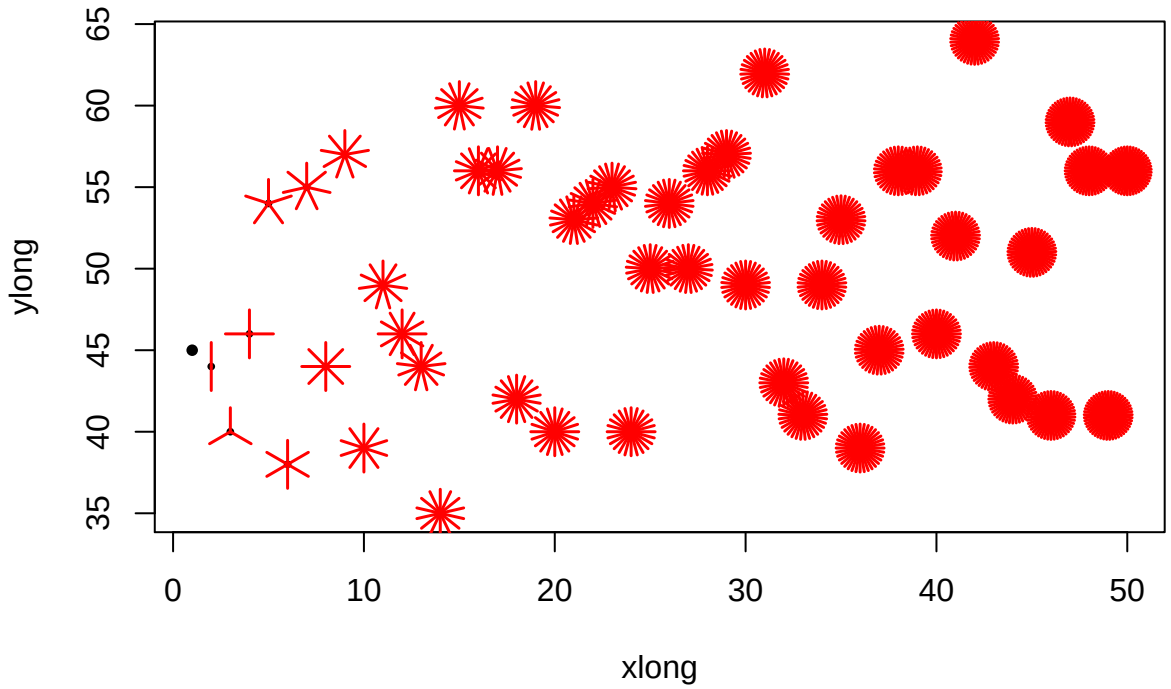
```
# plots with identical values

numbers <- read.table("longdata.txt", header = TRUE)
attach(numbers)
names(numbers)
```

```
## [1] "xlong" "ylong"
```

```
plot(jitter(xlong, amount = 1), jitter(ylong, amount = 1), xlab = "input", ylab = "count", pch = 16)
```




```
# jitter: Add a small amount of noise to a numeric vector.
# jitter(x, ...) returns a numeric of the same length as x, but with an amount of noise added in order


# use sunflowerplot
sunflowerplot(xlong, ylong)
```




```
# Multiple points are plotted as 'sunflowers' with multiple leaves ('petals') such that
# overplotting is visualized instead of accidental and invisiblec
detach(numbers)
```

Tables