Chapter 5: Graphics with R

Contents

From 2005 on, the library `ggplot2` was started. It is an attempt to take the good things about base and lattice graphics and improve on them (see later topic).

1 Scatterplot

Syntax:

```
plot(Y ~ X, ...)
```

Create a two-dimensional scatterplot of `X` versus `Y` in R as:
`plot(X, Y)` or `plot(Y~X)`
with possible arguments:

- `main`: plot title
- `xlab`: label for the x-axis
- `ylab`: label for the y-axis
- `col`: color of what is plotted
- `xlim`: limits for the x-axis
- `ylim`: limits for the y-axis
- `cex`: magnification factor
- `type`: `p` for points, `l` for lines, or `h` for histogram-like vertical lines, ...
- `pch`: style of the points
- `lty`: type of the line
- `lwd`: thickness of the line

High-level graphic functions will set up the plot figure.
Low-level graphic functions add to the existing figure.

Function	High- or low-level function	Description
`plot(x, y)` or `plot(y ~ x)`	High-level	Function to create a plot
`points()`	Low-level	Function for drawing points
`lines()`	Low-level	Function for joining points with line segments
`abline()`	Low-level	Function for adding lines to a figure
`curve()`	Low- or high-level. It depends on the argument `add = TRUE` or `add = FALSE`	Function for drawing a curve corresponding to a function.
`text()`	Low-level	Adds text to a figure or to special points
`title()`	Low-level	Adds a title to a figure
`legend()`	Low-level	Adds a legend to a figure

Consider the `airquality` data (package `datasets`)

airquality {datasets}

New York Air Quality Measurements

Description

Daily air quality measurements in New York, May to September 1973.

Usage

```
airquality
```

Format

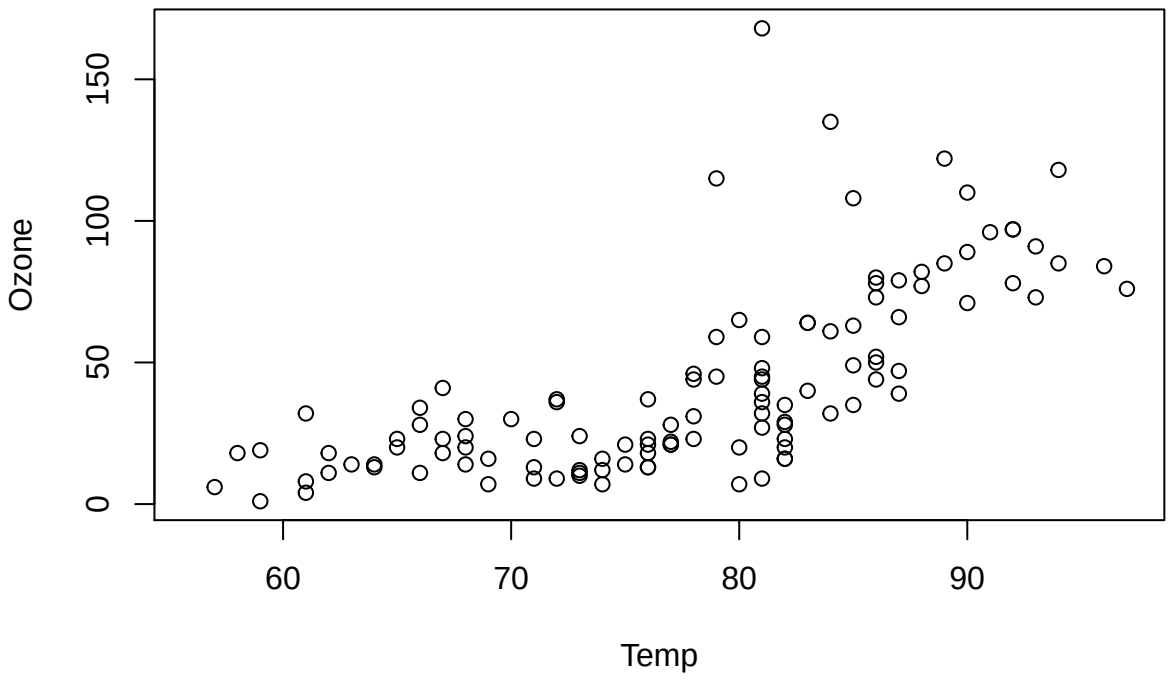
A data frame with 153 observations on 6 variables.

`[,1] Ozone`	numeric Ozone (ppb)
`[,2] Solar.R`	numeric Solar R (lang)
`[,3] Wind`	numeric Wind (mph)
`[,4] Temp`	numeric Temperature (degrees F)
`[,5] Month`	numeric Month (1--12)
`[,6] Day`	numeric Day of month (1--31)

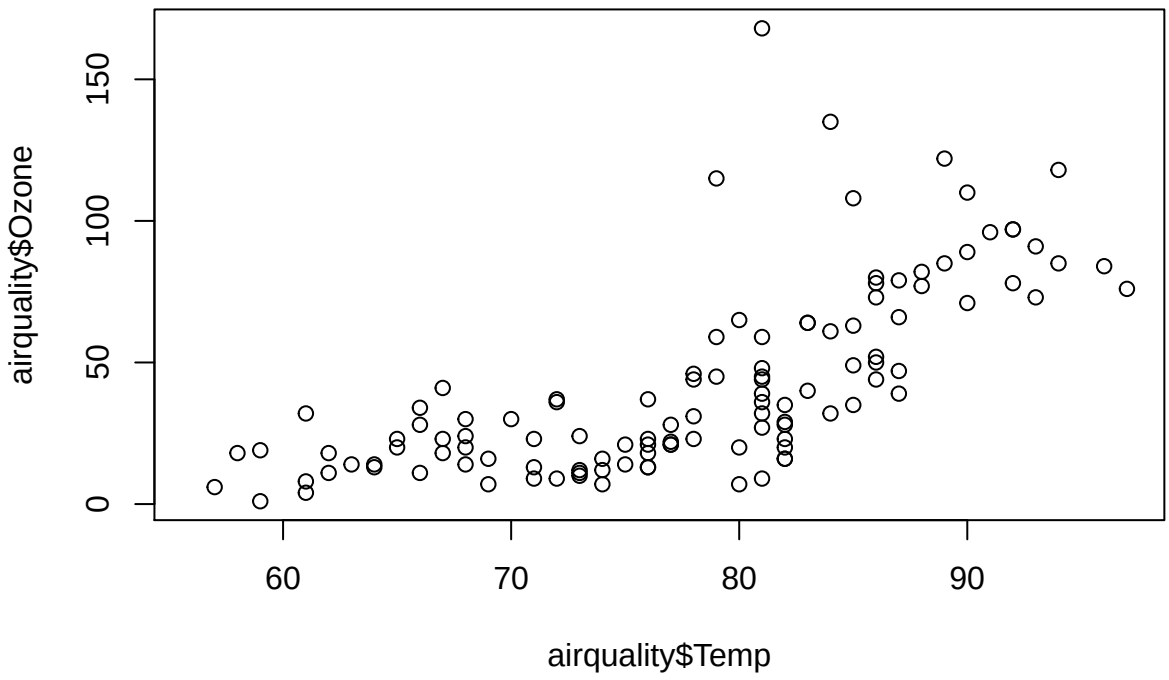
```
library(datasets)
names(airquality)
```

```
## [1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

```
plot(Ozone~Temp, data = airquality)
```




```
plot(airquality$Temp, airquality$Ozone)
```

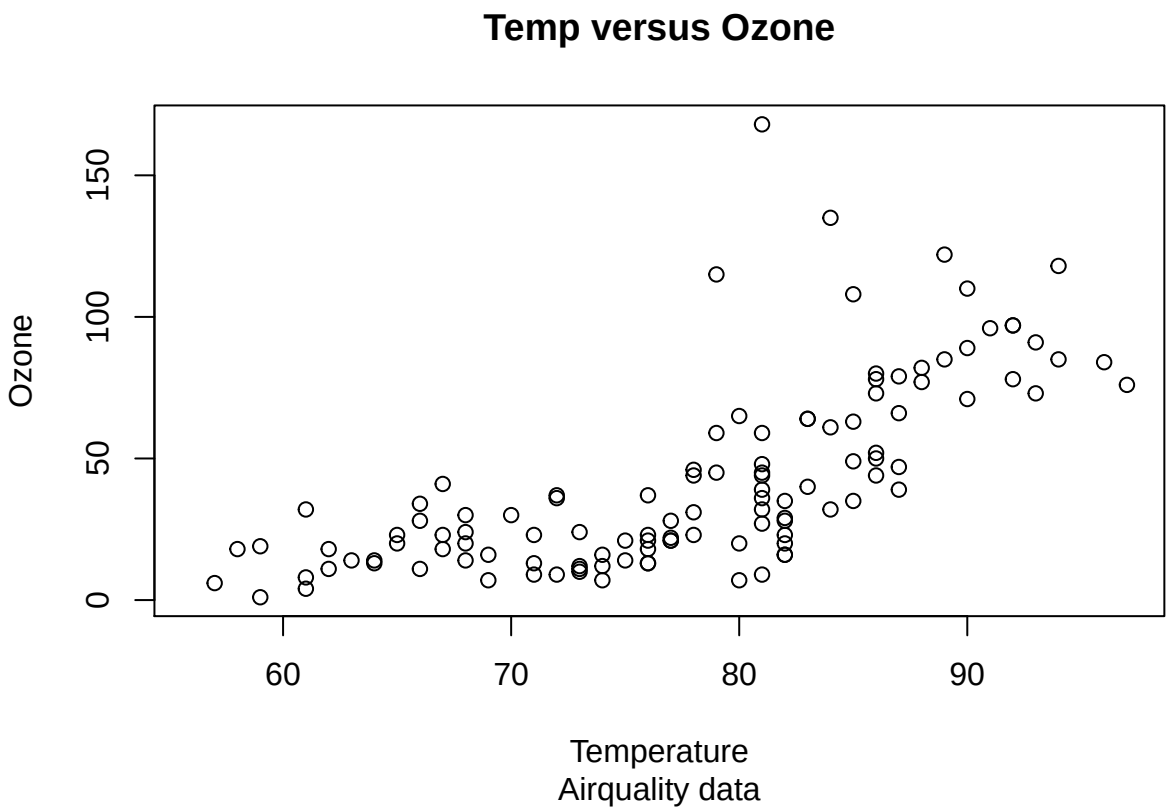


2 Adding to a figure

2.1 Adding titles and labels to a figure

Adding a overall title, a sub title and adjust the label of the x-axis:

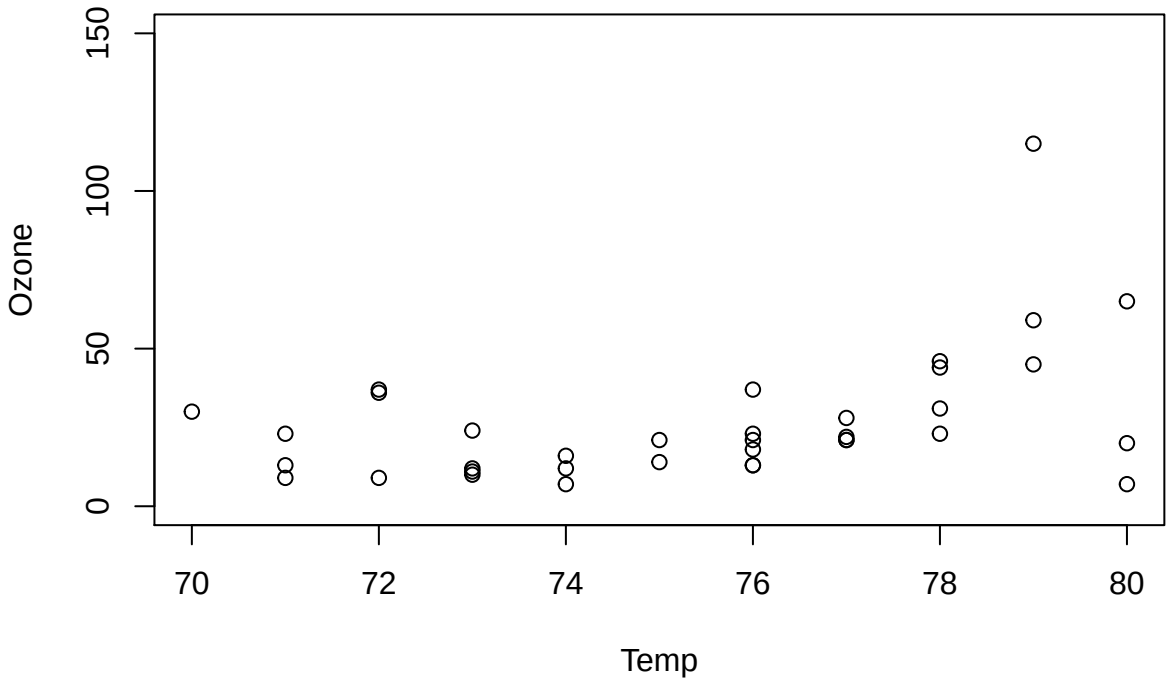
```
plot(Ozone~Temp, data = airquality, main = 'Temp versus Ozone',
     sub = 'Airquality data', xlab = 'Temperature')
```



2.2 Setting up the coordinate system and axes of a graph

```
xlim = c(xmin, xmax)
ylim = c(ymin, ymax)
```

```
plot(Ozone~Temp, data = airquality, xlim = c(70, 80), ylim = c(0, 150))
```



2.3 Plotting points, line types,...

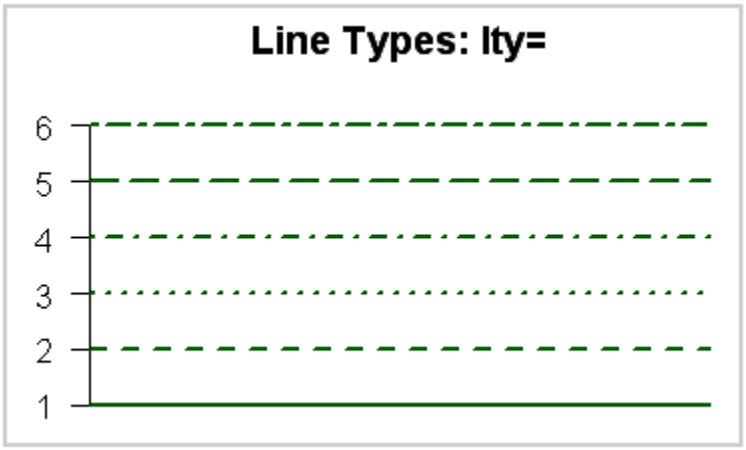
	Argument
What to visualize?	`type = ...`
Line types	`lty = ...`
Plotting characters	`pch = ...`
Plotting colors	`col = ...`
Rescale the size of the symbol	`cex = ...`
Rescale the thickness of the line	`lwd = ...`

2.3.1 Options for `type = ...`

What to visualize?	`type = ...`
Points	`type = 'p'`
Lines	`type = 'l'`
Points and lines	`type = 'b'`
"Overstruck" points and lines	`type = 'o'`
Vertical lines	`type = 'h'`
Stairstep plot	`type = 's'`
Empty plot	`type = 'n'`

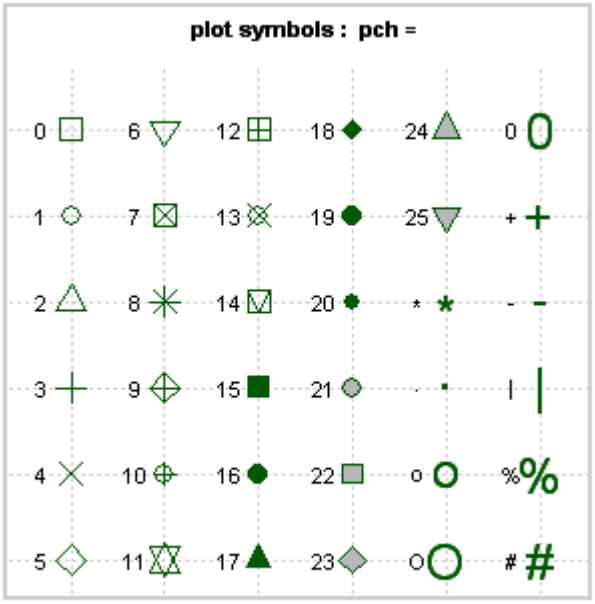
2.3.2 Options for `lty = ...` (line type)

You can select the line type by using the corresponding value for the option `lty = ...`.
The default version of R is `lty = 1` (solid line).
The following possibilities for the line type are available in R:



2.3.3 Options for `pch = ...` (plotting character)

To select the plot symbol, you can use option `pch = ...`.
The default version of R is `pch = 1` (circle). The following symbols are available in R:



2.3.4 Options for `col = ...` (color of the plot)

R stores a current palette that allows for reference of colors by a number.

Every color has its own number:

Color	`col = ...`
Black	`col = 1`
Red	`col = 2`
Green	`col = 3`
Blue	`col = 4`
Cyan	`col = 5`
Magenta	`col = 6`
Yellow	`col = 7`
Gray	`col = 8`

For more info about using colors in R, click here

2.3.5 Options for `cex = ...` (scale of the size of the symbol)

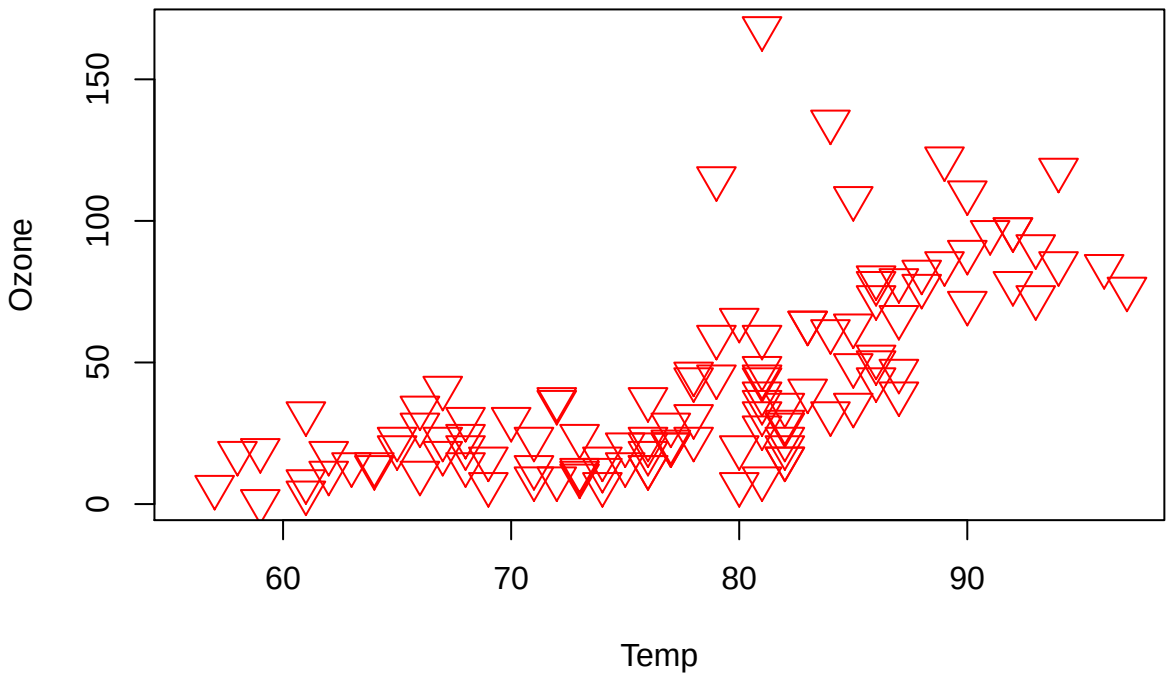
The default version of R is `cex = 1`. For `cex = x`, the size of the symbol is `x` times larger than the default. For instance, if `cex = 2`, the size of the symbol is 2 times larger than the default.

2.3.6 Options for `lwd = ...` (thickness of the line)

The line width can be adjusted with the argument `lwd = ...`. The default version of R is `lwd = 1`. Other values represent the line width relative to the default. For instance, the line is twice as wide than the default if `lwd = 2`.

2.3.7 Several arguments

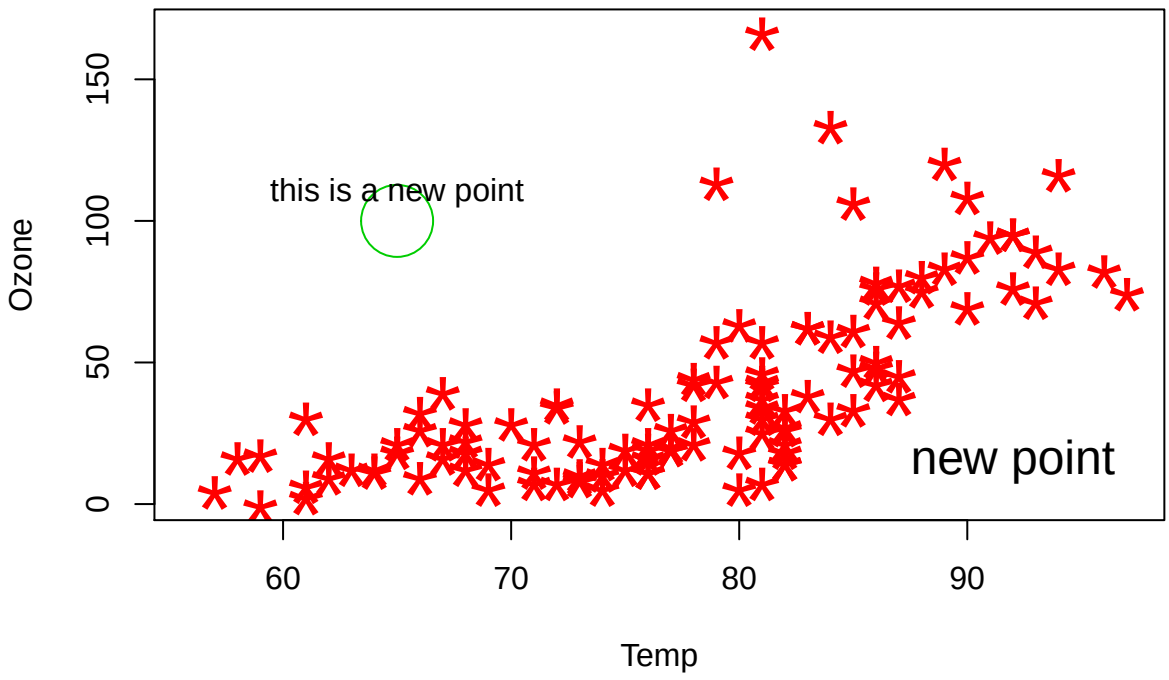
```
plot(Ozone ~ Temp, data = airquality, type = 'p', col = 2, pch = 6, lty = 3, cex = 2)
```



2.4 Adding information to a plot

Adding points or text to a figure

```
plot(Ozone ~ Temp, data = airquality, pch = '*', cex = 3, col = 2)
points(65, 100, col = 3, cex = 5)
text(locator(1), 'new point', cex = 1.5)
text(65, 110, 'this is a new point')
```



Remark:
By using the `locator()` you can add text interactively, after clicking the chosen location on the graph.

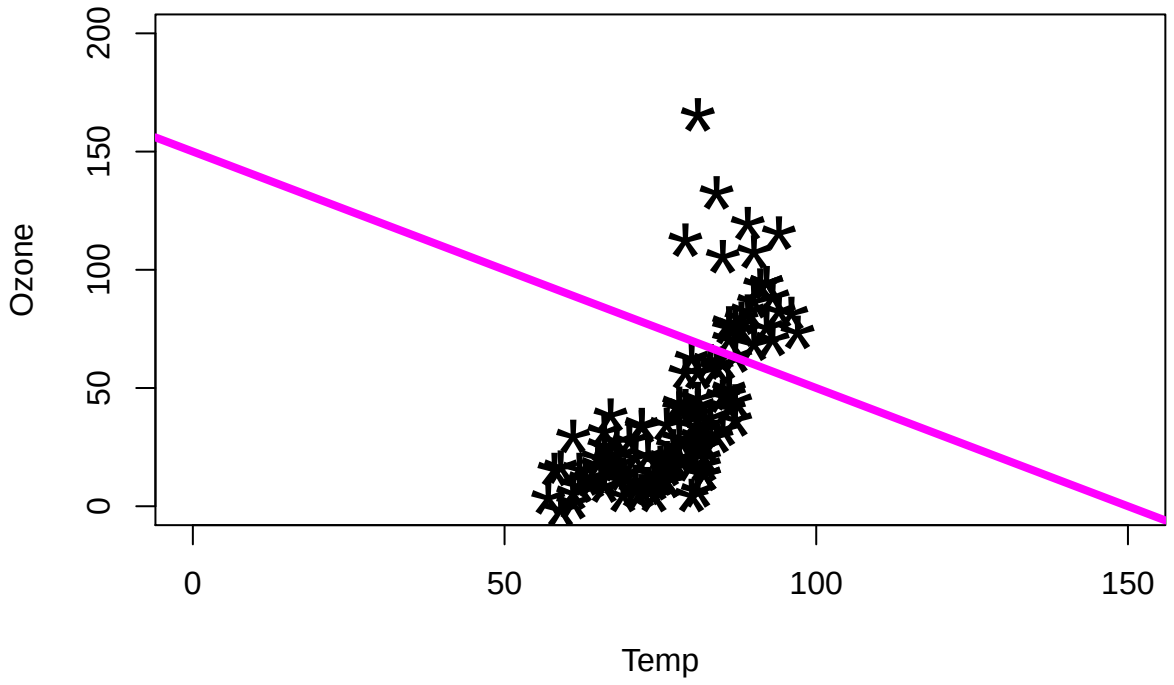
2.5 Adding lines to a graph

You can use the function `lines` or `abline` (or `curve` when you work with functions) to add a line to a graph.

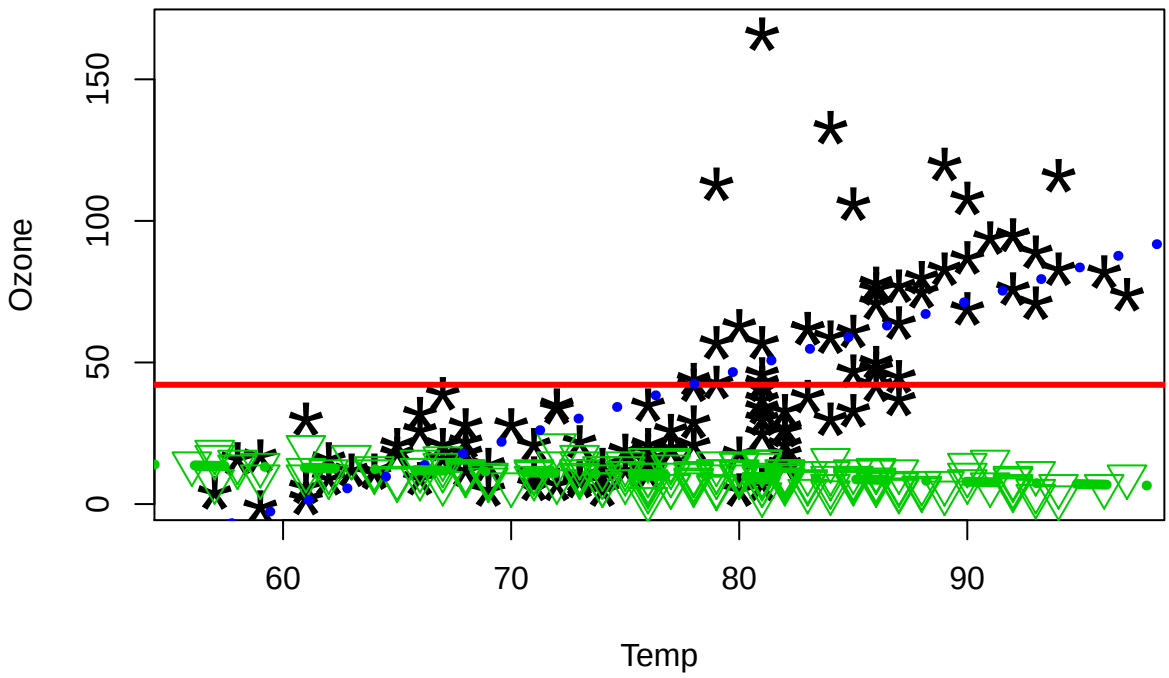
2.5.1 Function `abline`

- *Description:* This function adds one or more straight lines through the current plot.
- *Usage:*
  `abline(a = NULL, b = NULL, h = NULL, v = NULL, ...)`
- *Some arguments:*
  - `a, b` → the intercept and slope, single values.
  - `h` → the y-value(s) for horizontal line(s).

```
plot(Ozone~Temp, data = airquality, pch = "*", cex = 3, xlim = c(0,150), ylim = c(0,200))
abline(150, -1, lty = 1, col = 6, lwd = 4)
```



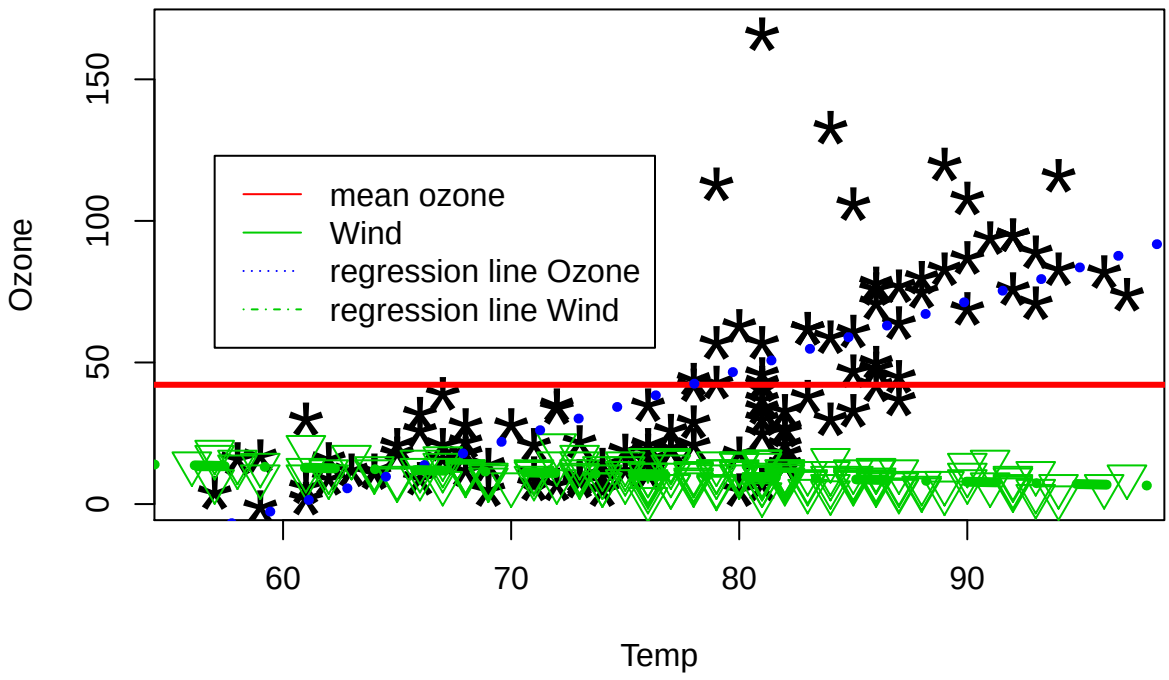

```
plot(Ozone~Temp, data = airquality, pch = "*", cex = 3)
mean_ozone <- mean(airquality$Ozone, na.rm = T)
abline(h = mean_ozone, lty = 1, col = 2, lwd = 3)

# Adding extra wind measurements
lines(Wind~Temp, data = airquality, pch = 6, col = 3, cex = 2, type = 'p')

# Adding a regression line for Ozone versus Temp
result.lm <- lm(Ozone~Temp, data = airquality)
abline(result.lm, lty = 3, col = 4, lwd = 5)

# Adding a regression line for Wind versus Temp
result2.lm <- lm(Wind~Temp, data = airquality)
abline(result2.lm, lty = 4, col = 3, lwd = 5)
```



Remark:
The function `lm(formula, data)` is used to fit linear models. The argument `formula` is a symbolic description of the model to be fitted. A typical model has the form `response ~ terms`. The function `lm` returns a list that contain the coefficients, the residuals, the fitted mean values, etc.

2.6 Adding a legend to a plot

```
legend(locator(1), c('mean ozone', 'Wind', 'regression line Ozone', 'regression line Wind'),
       lty = c(1, 1, 3, 4), col = c(2, 3, 4, 3))
```



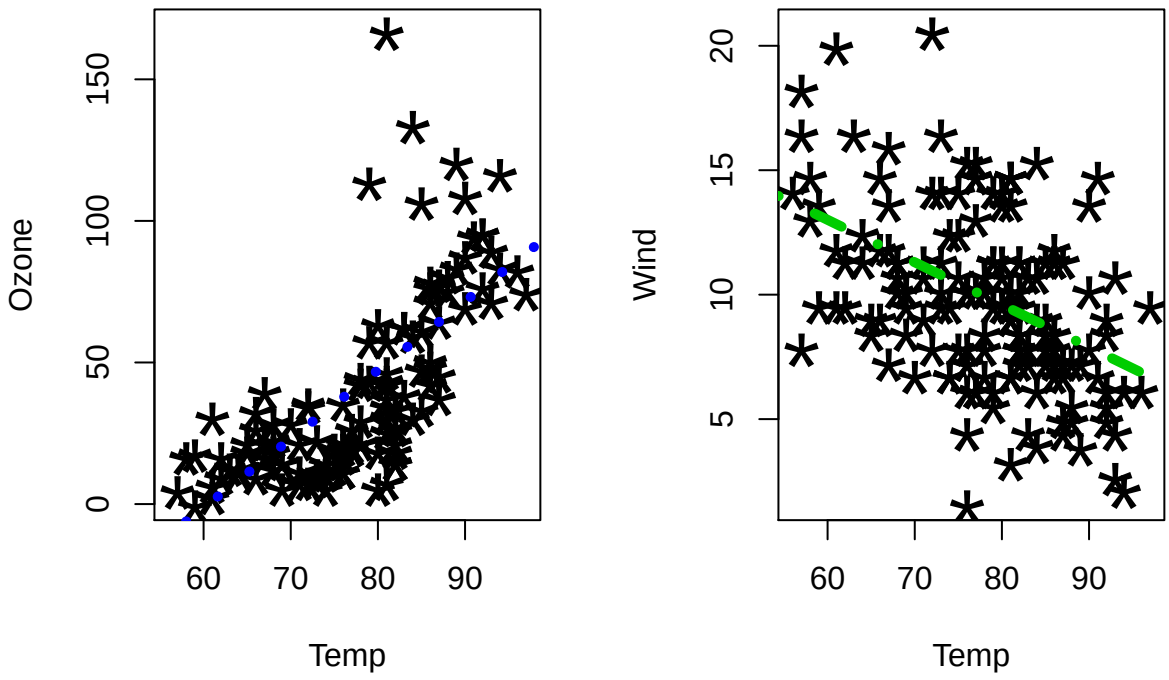
3 Multiple plots on one graphical window

Syntax:

```
par(mfrow = c(3,2))
```

This code tells R to split the graphical window into six equal pieces, with 3 rows and 2 columns.

```
# Multiple plots on same graphical window
par(mfrow = c(1,2))

# Scatterplot and regression line of Ozone versus Temp
plot(Ozone ~ Temp, data = airquality, pch = '*', cex = 3)
result.lm <- lm(Ozone ~ Temp, data = airquality)
abline(result.lm, lty = 3, col = 4, lwd = 5)

# Scatterplot and regression line of Wind versus Temp
plot(Wind ~ Temp, data = airquality, pch = '*', cex = 3)
result2.lm <- lm(Wind ~ Temp, data = airquality)
abline(result2.lm, lty = 4, col = 3, lwd = 5)
```



4 Histogram with usual R graphics

4.1 General

Syntax:

```
hist(x, ...)
```

Creating vectors (`x`, `y`, and `z`) of values for which a histogram will be created.

```
x <- rnorm(50)
y <- rexp(50, rate = 2)
range(y)
```

```
## [1] 0.01011667 1.72388573
```

```
z <- rchisq(50, 10)
range(z)
```

```
## [1]  2.977256 18.341695
```

Split the graphical window in three equal pieces:

```
par(mfrow = c(1, 3))
```

By default, R draws a histogram of absolute frequencies. By setting the argument `probability = TRUE`, we ask for a histogram of relative frequencies.

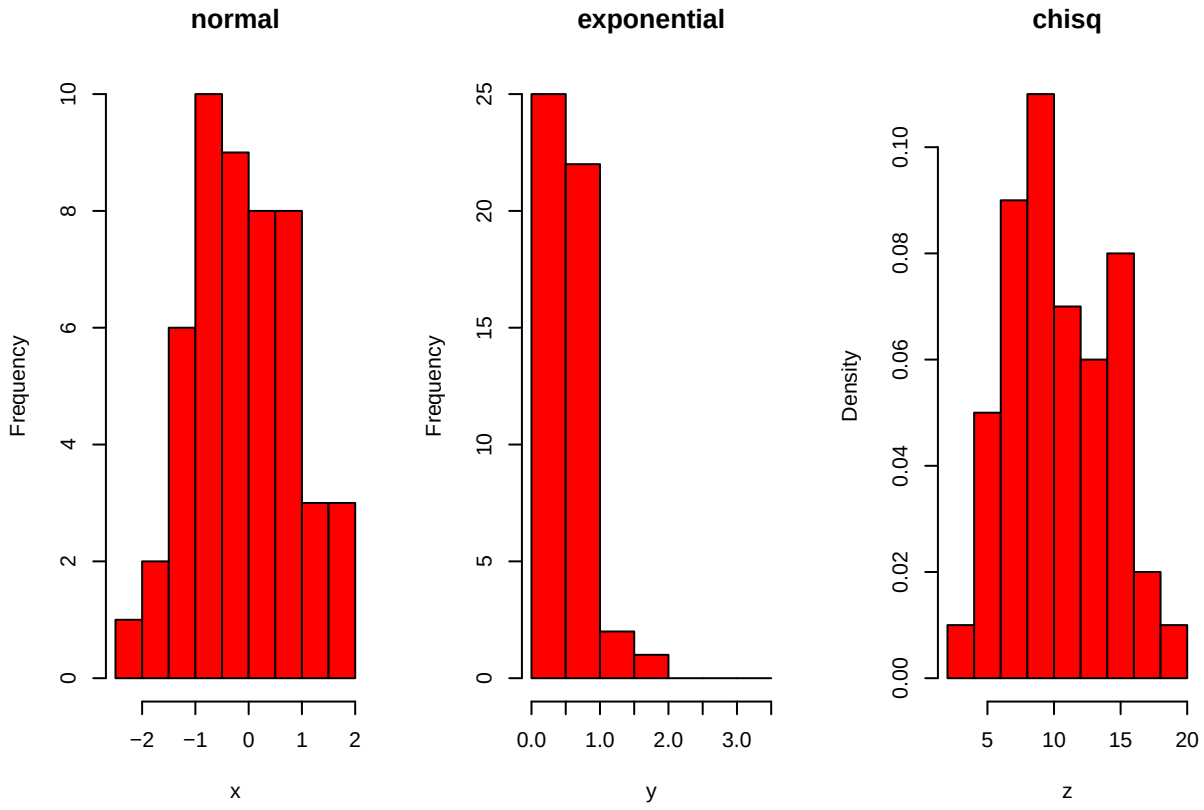
The first two histogram graphics are a representation of absolute frequencies.

```r
hist(x, nclass = 10, main = 'normal', col = 2)

hist(y, breaks = seq(0.0, 3.5, 0.5), main = 'exponential', col = 2)
```

In this third histogram, relative frequencies are plotted (since `probability = T` for this histogram).

```r
hist(z, nclass = 7, probability = T, main = 'chisq', col = 2)
```

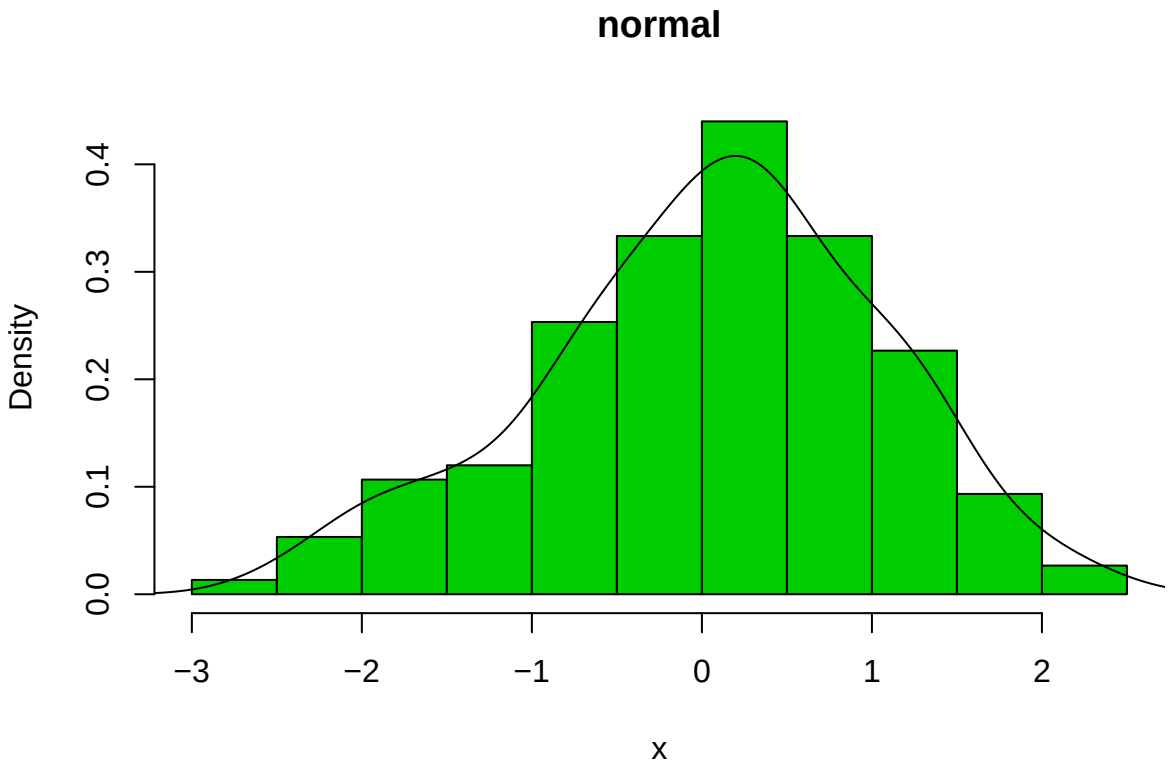
For the first and third histogram, `nclass = ...` determines the number of bars for the histogram (respectively `10` and `7`).
In the second histogram, the vector `seq(0.0, 3.5, 0.5)` determines the breakpoints.



4.2 Creating a histogram with a density curve on it

The `density()` function will find a density estimate from the data.
We then use the `lines()` to add the density estimate to the existing graph.

```r
x <- rnorm(150)
hist(x, prob = TRUE, main = 'normal', col = 3)
lines(density(x))
```



5 Boxplots

Syntax:

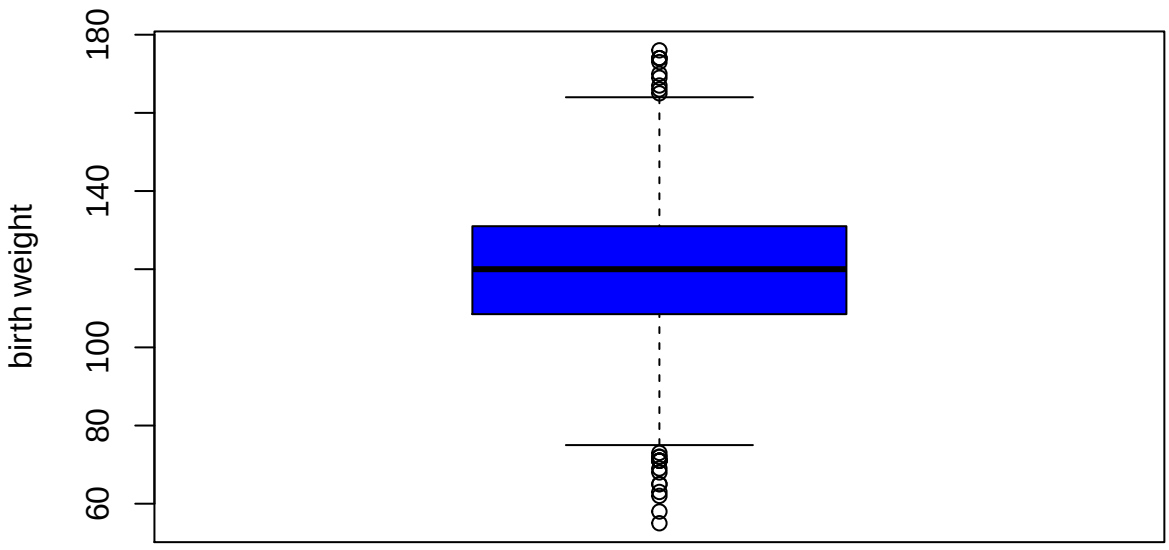
```
boxplot(x, ...)
```

In this section, we use the data frame `babies` from the package `UsingR`. This data frame is about new born babies. Some important variables are

- `Weight`: Birth weight of the baby (in ounces)
- `Smoke`: Smoke behavior of the mother.
  Does the mother smoke?
  - `0` = never
  - `1` = smokes now
  - `2` = until current pregnancy
  - `3` = once did, not now
  - `9` = unknown

We are making a boxplot for the birth weight (`wt`) variable from the data set `babies`.

```
boxplot(babies$wt, ylab = 'birth weight', col = 4)
```




```
f <- fivenum(babies$wt)
f
```

```
## [1]  55.0 108.5 120.0 131.0 176.0
```

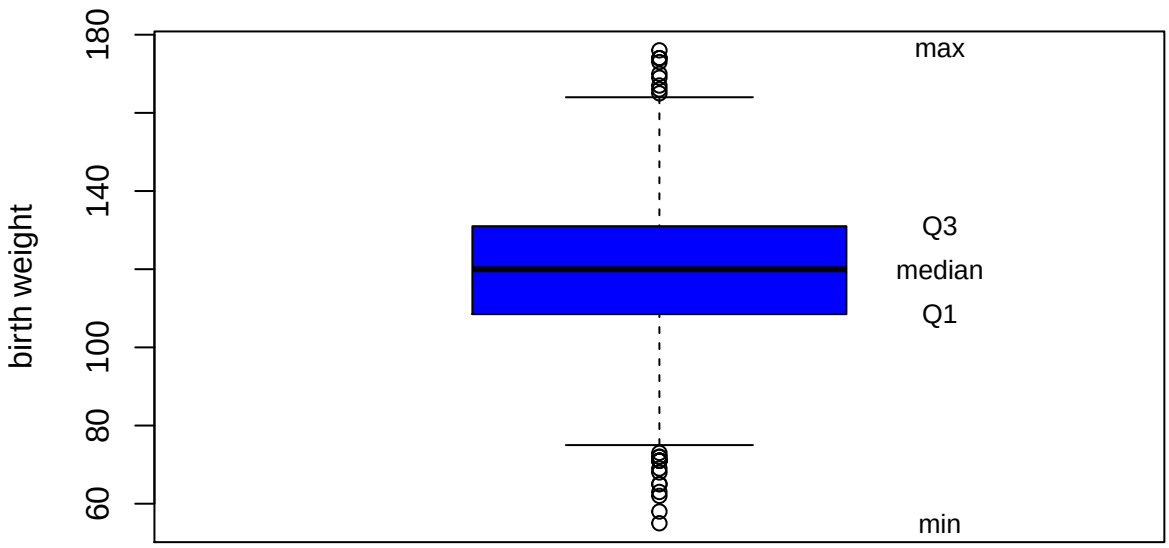
The function `fivenum(x, na.rm = TRUE)` returns Tukey's five number summary for the input data `x`, i.e.,

1. minimum
2. lower-hinge (= lower quantile, Q1)
3. median
4. upper-hinge (= upper quantile, Q3)
5. maximum.

The **`text()`** function places the values of labels on the graph as specified.
Syntax:

```
text(x, y, labels = ...)
```

```
boxplot(babies$wt, ylab = 'birth weight', col = 4)
text(rep(1.3, 5), f, labels = c('min', 'Q1', 'median', 'Q3', 'max'), cex = 0.8)
```



Remark:

1. Getting the outliers

Outliers are considered as data points which are not within 1.5 · IQR (interquartile range) from the lower quartile Q1 or the upper quantile Q3.

Looking for outliers in birth weight of the babies (`wt`):

```
f <- fivenum(babies$wt)
IQR <- f[4] - f[2]

# What is the identification number of the outlier babies?
outliers <- babies$id[babies$wt > f[4] + 1.5*IQR | babies$wt < f[2] - 1.5*IQR]
outliers
```

```
##  [1] 3906 5287 5845 6241 6343 6460 6534 6660 6760 6997 7080 7083 7109 7290 7334
## [16] 7524 7544 7722 7828 7884 7979 7984 8054 8187 8219 8369
```

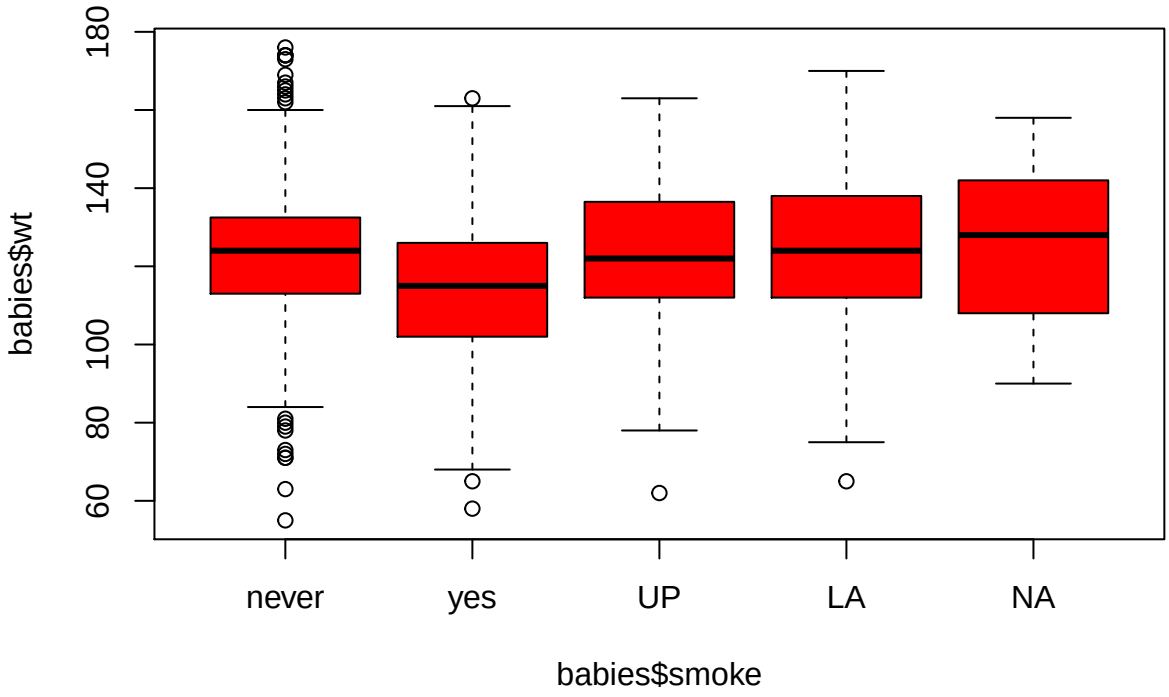
```
# What is the weight of the outlier babies?
outliers_wt <- babies$wt[babies$wt > f[4] + 1.5*IQR | babies$wt < f[2] - 1.5*IQR]
outliers_wt
```

```
##  [1] 173  71  71  68  69  71 174 170 176 166 167  71 174 165  62  72  58  55 169
## [20]  65  73  65 174  63  72  71
```

2. **Grouped** boxplots

We use the data set `babies` and compare the weight of the babies (`wt`) over the several smoking groups of the mothers (`smoke`).

```r
smoke.names <- c("never", "yes", "UP", "LA", "NA")

boxplot(babies$wt ~ babies$smoke, col = 2, data = babies, names = smoke.names)
```



6 Exercises

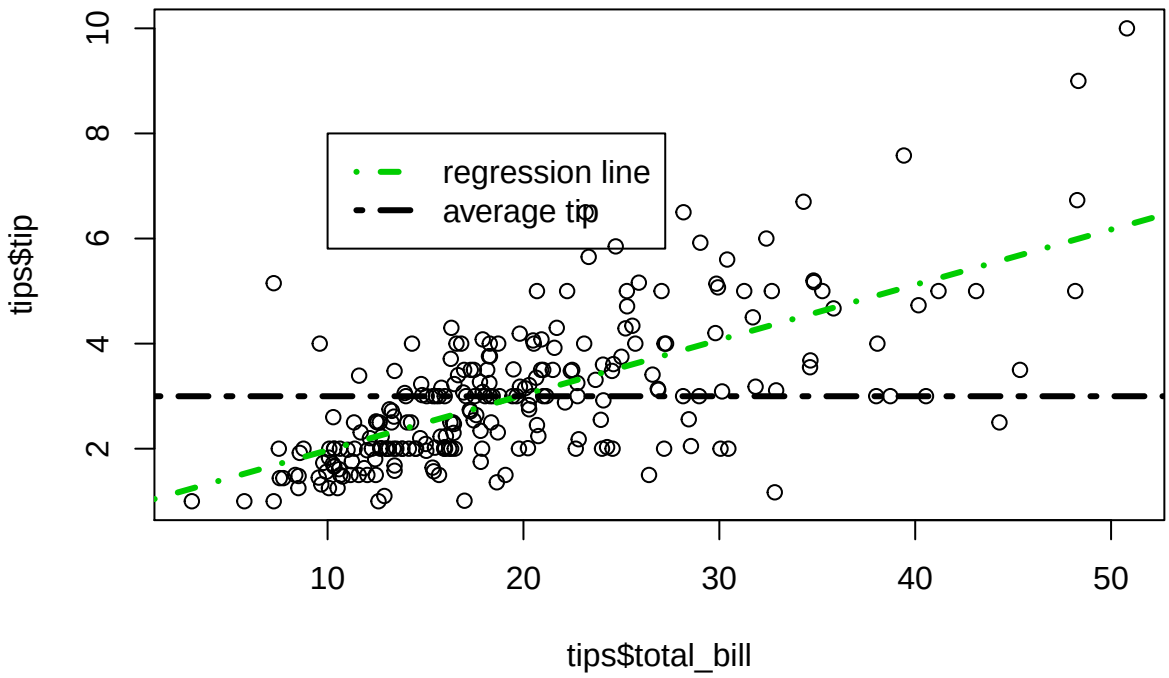
In the first two exercises, the data `tips` from the package `reshape` will be used.

One waiter recorded information about each tip he received over a period of a few months working in one restaurant. He collected several variables:

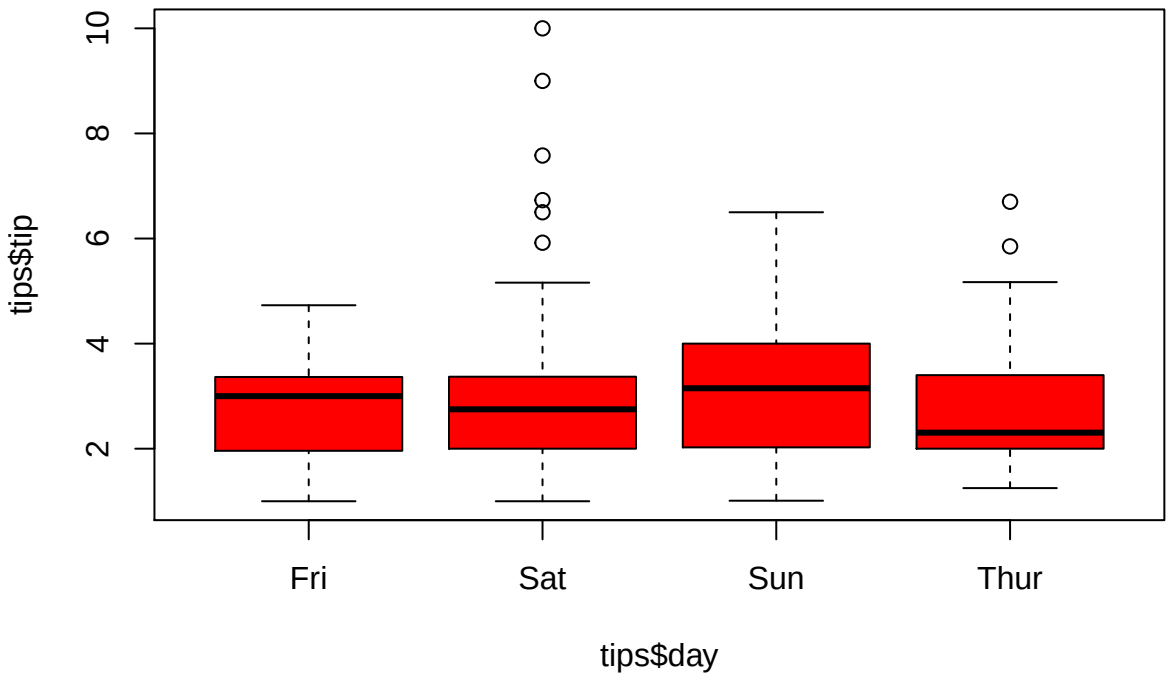
- `tip`: tip in dollars
- `total_bill`: bill in dollars
- `sex`: sex of the bill payer
- `smoker`: whether there were smokers in the party
- `day`: day of the week
- `time`: time of the day
- `size`: size of the party

In all he recorded 244 tips.

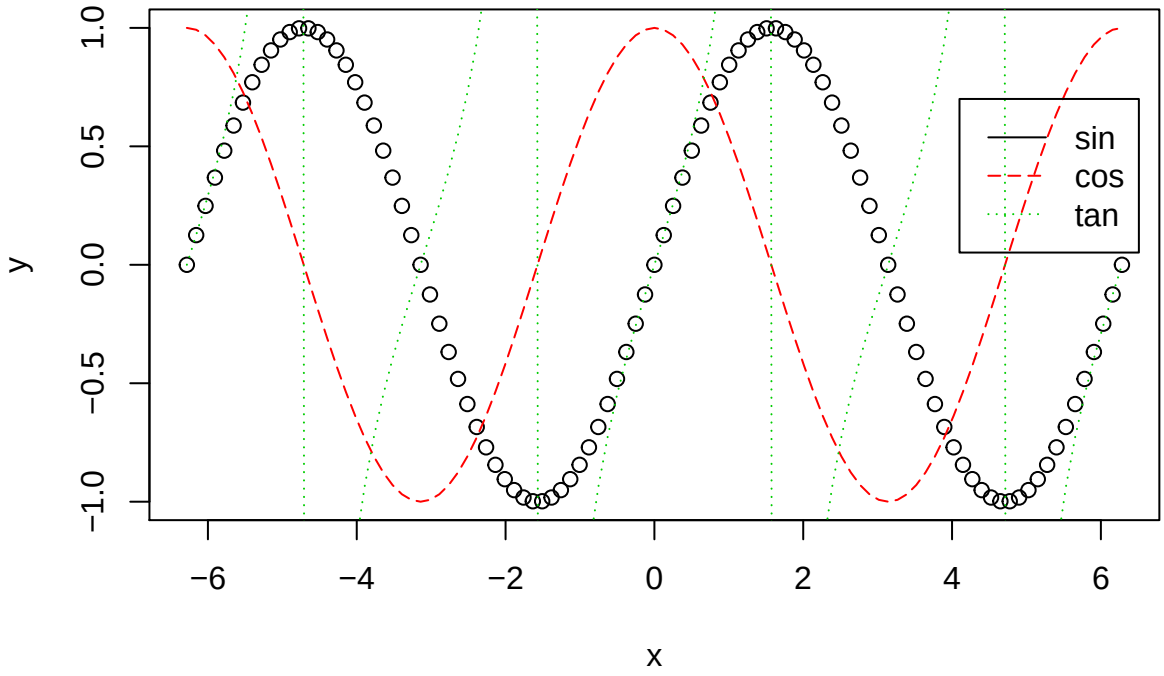
1. Make a simple scatterplot of the tip (Y) versus bill (X). Add a regression line and add a horizontal reference line with average tip. Add a legend.



2. Make a grouped boxplot of tip by day of the week



3. Plotting multiple lines
   This exercise is not using the ***`tips`*** *data frame*
   Create the graph of the functions `sin`, `cos`, and `tan` in one figure. Plot the functions on the interval $[-2\pi, 2\pi]$ and separate the graphs by line style (add a legend).


sin
cos
tan
y
x
1.0
0.5
0.0
−0.5
−1.0
−6
−4
−2
0
2
4
6