MATH 3424 Tutorial 2: Plot in R

Zhongyuan Lyu

Import data into R

1. We use `read.table` to import white space (or tab) delimited files (e.g., txt file).

- `header`: Use `header=TRUE` if the first line of the file contains the names of the columns.
- `sep`: If the delimiter of the columns is not white space, but another character, you need to use the additional argument sep.
- many other arguments..

2. `read.csv` is a sibling of `read.table` with the main difference that it assumes by default that the data is comma-separated and the first line contains the variable names.. In other words, you do not need to specify `sep=","` and `header=TRUE` when using `read.csv`.

3. Check `write.table` and `write.csv` yourself.

```
heightdata <- read.table("Heights of husband and wife.txt",
                         header=TRUE)
head(heightdata)
```

```
##   Husband Wife
## 1     186  175
## 2     180  168
## 3     160  154
## 4     186  166
## 5     163  162
## 6     172  152
```

Ploting in R

- To get an idea of what R can do about graphs, enter: `demo(graphics)`
- Cheatsheets in Rstudio might help.

The generic R function: plot

- The function plot is at the heart of R's built-in graphics.
  - `plot(x, y, ...)` where x and y are the vectors containing the coordinates.
  - `plot(y~x, data=data, ...)` where x and y are column in dataframe.
  - `plot(M, ...)` where M is a matrix having two columns (x- and y-coordinates). If M has more than two columns, R will ignore these.
  - `plot(lst, ...)` where lst is a list with (possibly amongst others) an element x and an element y.

Customising the plot

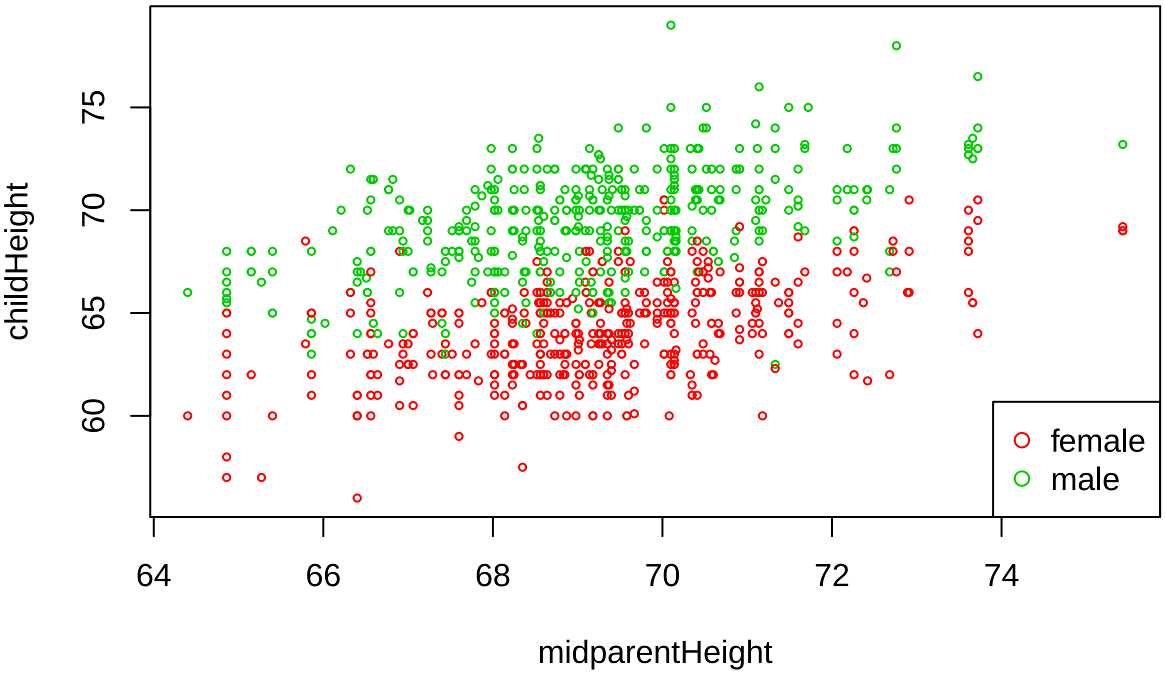
`plot` has a wide range of optional arguments. The most important ones are:

- `type`: controls how the data is plotted. Use `p` (default) for points, `l` for a line through the points, `b` for a line together with the points..
- `xlab`/`ylab`: the label of the x/y axis (in quotes).
- `main`/`sub`: the title/subtitle of the plot (in quotes).
- `xlim`/`ylim`: if set to `c(xmin, xmax)` the range of the x axis is from `xmin` to `xmax`.
- `log`: controls which axes should use a logarithmic scale.
- `lty`: the type of line (`0`=blank, `1`=solid (default), `2`=dashed, `3`=dotted, `4`=dot-dash, `5`=long dash, `6`=two dashes).
- `lwd`: the width of the line.
- `pch`: the plotting symbol (`0-14` for open symbols, `15-20` for solid symbols, `21-25` for filled symbols).
- `col`: the colour.
- `cex`: the size of the plotting symbol.

An example: scatter plot

- We use `col` to use colour to denote the different genders. However, gender variable in GaltonFamilies is a factor (which can be checked via `class(GaltonFamilies$gender)`), so we first need to convert it to integers (which R accepts as colours). We can do this using the function unclass – we add 1 to the result, otherwise one group would have their points drawn in black (R's first choice of colour).
- The function `legend(position=position, ..., legend=legend)` can be used to add a legend to a plot. `legend` is the vector containing the labels to be used in the legend. position can be `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"`. Alternatively you can specify the coordinates of the legend.

```r
library(HistData)
data(GaltonFamilies)
plot(childHeight~midparentHeight,
     data = GaltonFamilies,
     col = 1+unclass(gender),
     cex = 0.5)
legend("bottomright",
       pch=1,
       col=1+1:nlevels(GaltonFamilies$gender),
       legend=levels(GaltonFamilies$gender))
```



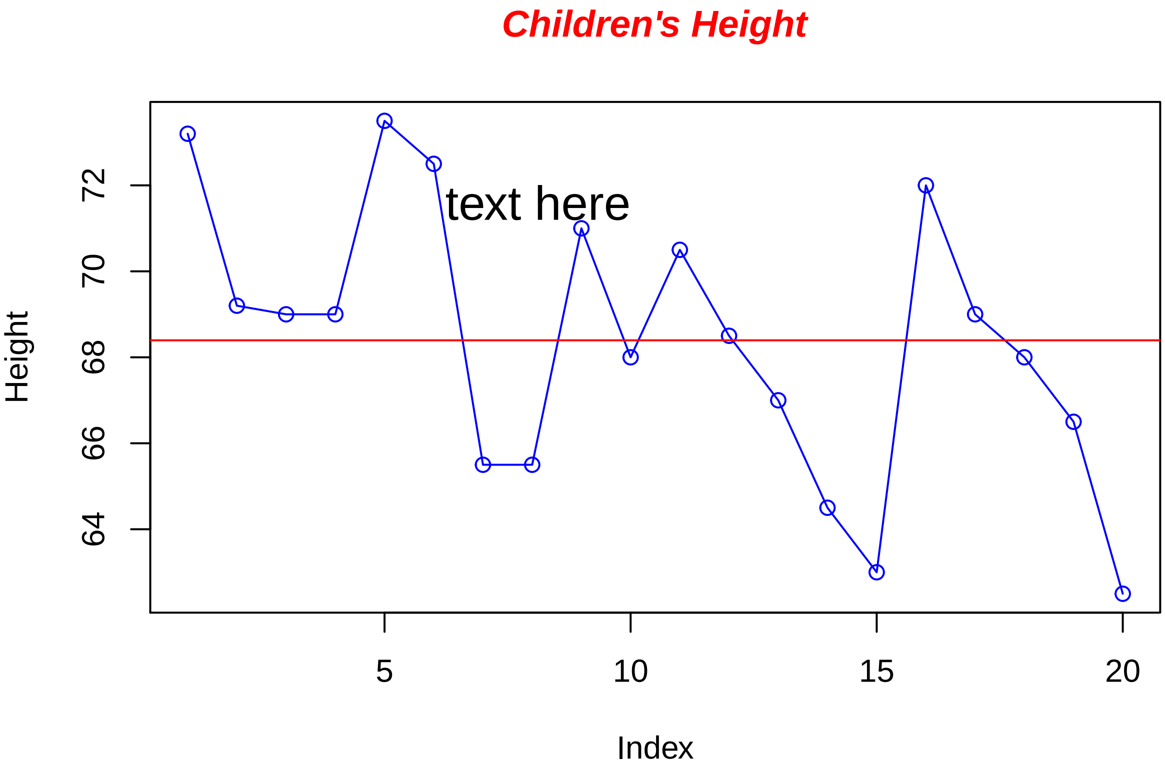
Play with plot

The function `abline` can be used to add a straight line to an existing plot

- `abline(h=ypos, ...)` draws a horizontal line at `ypos`.
- `abline(v=xpos, ...)` draws a vertical line at `xpos`.
- `abline(a=intercept, b=slope, ...)` draws a line with intercept as its `intercept` and `slope` as its slope.

The function `text(x, y, text, ...)` plots the text text (one character string or a vector of strings) at the coordinate(s) (`x`, `y`). The optional arguments include `col`, `cex`, and `adj=c(horiz, vert)`, which sets the horizontal adjustment to horiz (`0`: left justified, `0.5` centered, `1`: right justified) and the vertical adjustment to vert (`0`: bottom, `0.5` center, `1`: top). The defauilt is all centered `c(0.5, 0.5)`.

```
plot(x = 1:20,
     y = GaltonFamilies$childHeight[1:20],
     type="o",
     col="blue",
     xlab = "Index",
     ylab = "Height")
title(main="Children's Height", col.main="red", font.main=4)
abline(h=mean(GaltonFamilies$childHeight[1:20]),
       col="red")
text(x = 10, y = 72,                     # text draws the strings given in the vector labels
     labels = "text here",               # at the coordinates given by x and y
     adj=c(1,1),
     cex = 1.5)
```

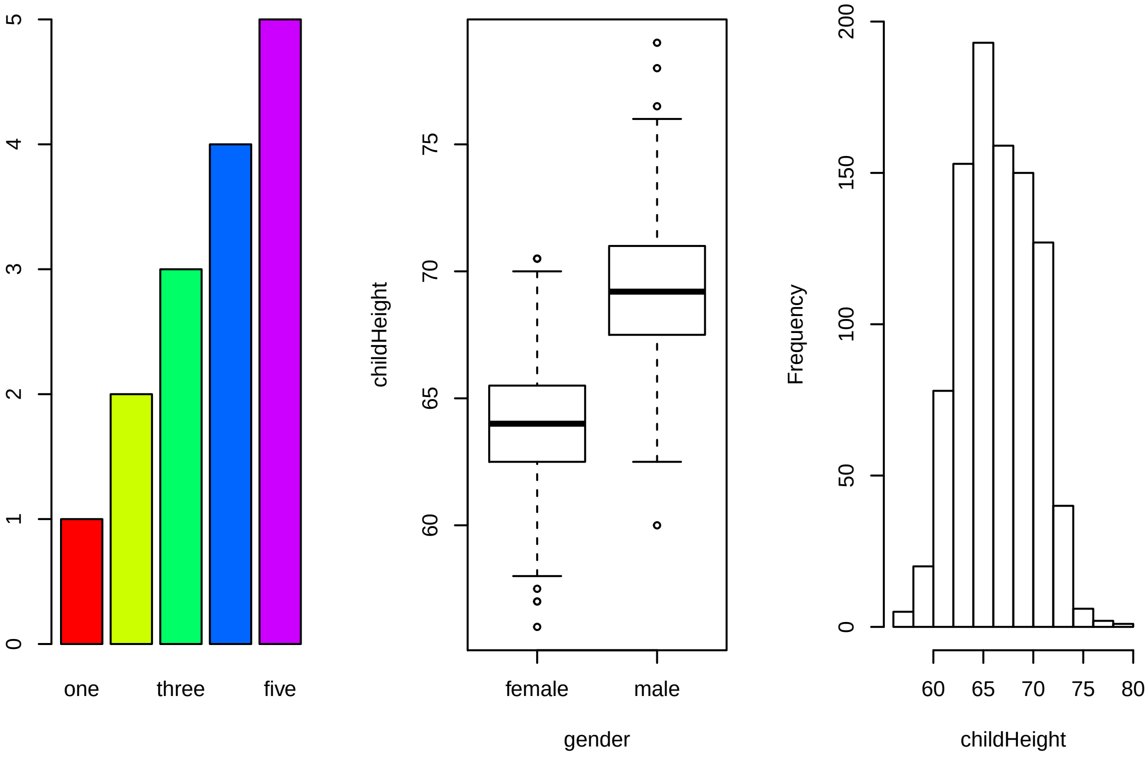


Mutiple graphs in one plot

More statistical plots are available in R. - `barplot(height, ...)` can be used to create bar plots. - `boxplot(y, ...)` creates a box plot of the data in the vector `y`. If `x` is a categorical variable then `boxplot(y~x, ...)` draws a boxplots separately for each level of `x`. - `hist(x, ...)` can be used to create histograms.

- par(mfrow=c(nrows, ncols)) divides the figure into nrows rows and ncols columns, which will be used in row-wise order.
- par(mfcol=c(nrows, ncols)) divides the figure into nrows rows and ncols columns, which will be used in column-wise order.

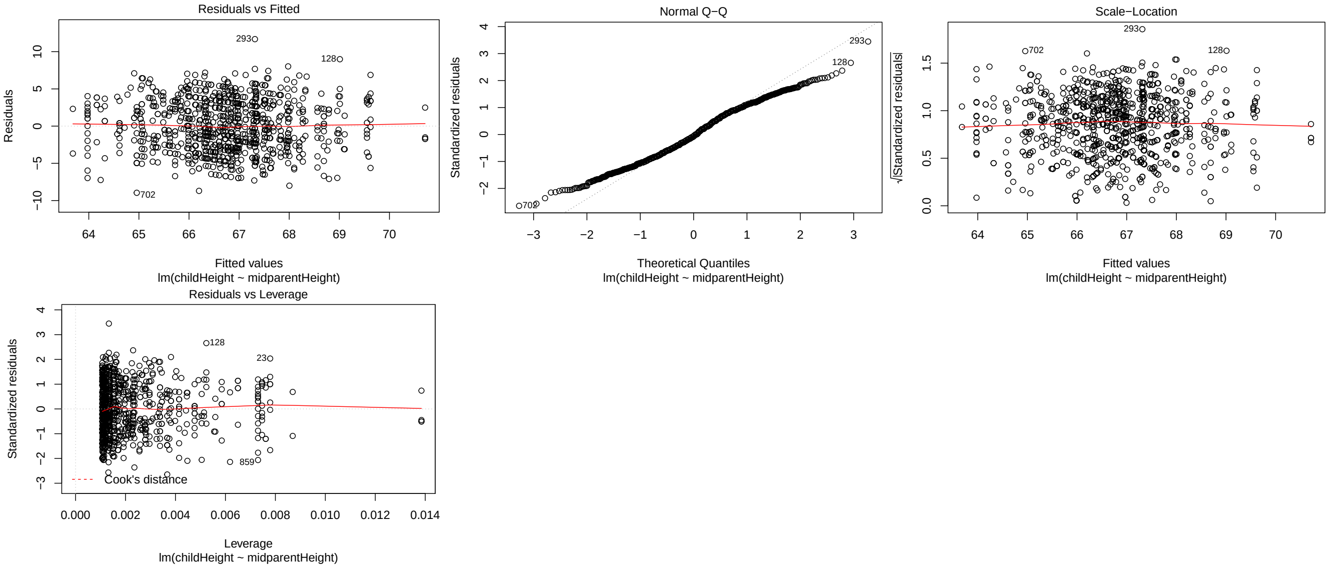
```
par(mfrow=c(nrows=1, ncols=3))
height <- 1:5;
names(height) = c("one", "two", "three", "four", "five");
barplot(height, col=rainbow(5))

boxplot(childHeight ~ gender,
        data = GaltonFamilies)
hist(x = GaltonFamilies$childHeight,
     main = "",
     xlab = "childHeight")
```



Plot an object

Many R objects (such as model fits) have a `plot` method, which draws a visualisation of or diagnostic check relating to this object. For example:

```
model <- lm(childHeight~midparentHeight, data=GaltonFamilies)
plot(model)
```



Elegant plot package: ggplot2

- Package ‘ggplot2’ is recommended if you want more sophisticated graphs, which is by far the most popular R package for graphics. It provides a declarative and simple, yet powerful interface.
- An official reference is available here: https://ggplot2.tidyverse.org.
- **If you want to have a quick start**, you can refer to “The R Graphics Cookbook” by Winston Chang, which is free online: https://r-graphics.org
- Check out more examples at: https://www.r-graph-gallery.com/ggplot2-package.html

Exercise:

We consider the default R dataset **cars**. We will implement simple linear regression **without** using `lm` function. Instead, we will use

1. Create two vectors `x` and `y`, such that `x` contains the speed of the cars (predictor) and `y` contains the stopping distance (response).
2. Compute $\hat{\beta}_0$ and $\hat{\beta}_1$ using the formula:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

3. Compute the estimate of variance:

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n-2}$$

4. Compute the goodnees-of-fit index R^2:

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

5. Create a design matrix:

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$$

6. Compute $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, \hat{\beta}_1)^\top$ using the formula:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top y$$

   which should give you the same quantities as in part 2.