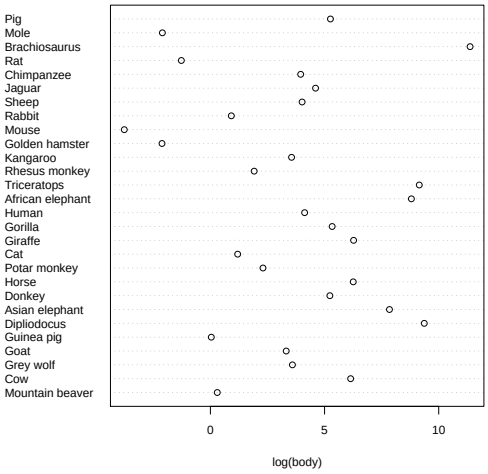
Exercises for the course

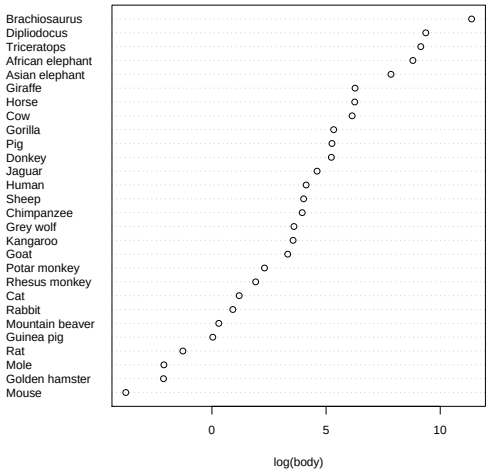
"An introduction to R"

Sheet 05

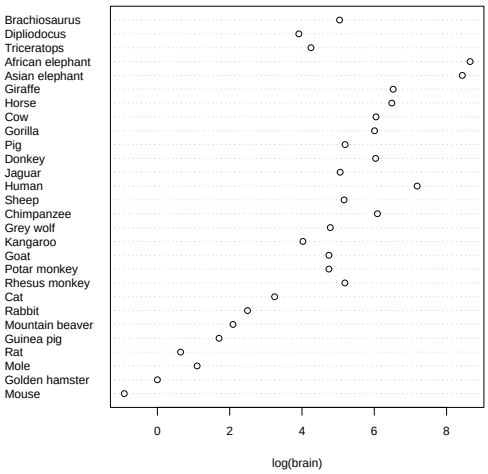
Exercise 25: Load the library `MASS` and attach the data frame `Animals` which contains the average brain and body weights for 28 species of land animals.

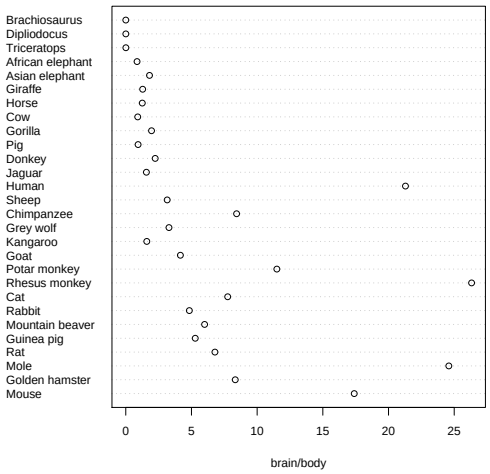
- Make a dotchart of `log(body)` with the row names (`row.names()`) of `Animals` as labels on the y-axis and with `"log(body)"` as label of the x-axis.
- Next improve this by ordering the list of animals according to `body`. Find the index vector `s` such that `body[s]` is the sorted vector. Repeat the dotchart such that the list of animals is now sorted according to body size.



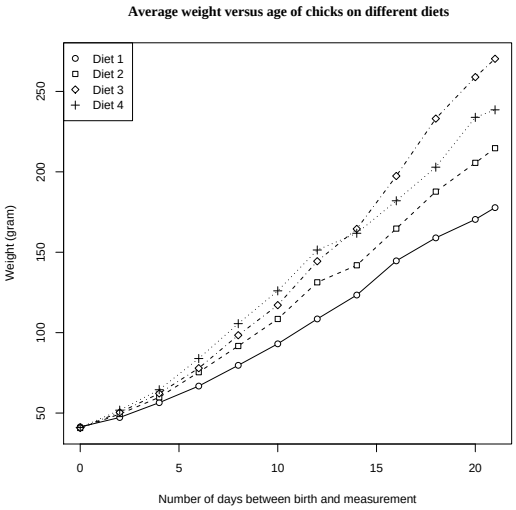


- Produce a similar dotchart of log(brain) sizes arranging the animals in sorted order by body size. What interesting features do you notice, if any?
- Finally produce a dotchart of `brain/body`.





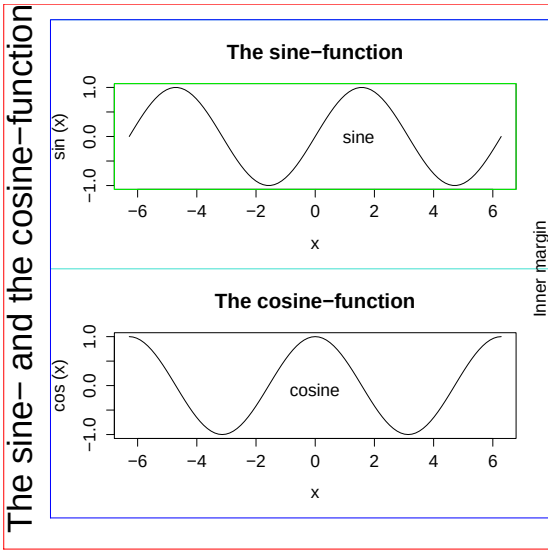
Exercise 26: Load the dataset `ChickWeights` with `data(ChickWeights)`. Attach this data set and get an overview over the data (including `?ChickWeight`). Calculate the mean of `weight` for each diet and for each day. Produce the following figure.



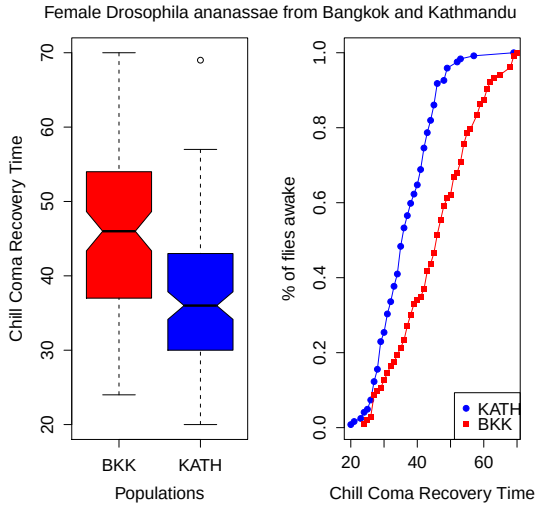
Hint: One difficulty is to obtain a properly scaled plotting region. For example if you start with a plot of the weight means of the chicks with diet 1, then the range of the y-axis will not be large enough. One way to find the range of the y-axis is to calculate the maximum and the minimum of `weight` and to use this to set the range of the y-axis. The point characters used for the plot are 21–23 and 3.

Exercise 27: *This exercise is to acquaint you with multi-figures and margins. Multi-figures are quite useful to illustrate different features of data.* Enter the command `op <- par(mfrow=c(2,1), oma=c(2,3,1,0)`. Subsequent figures will be drawn in a 2 by 1 multi-figure and the outer margins are as follows: 2 lines at the bottom, 3 lines on the left-hand side, 1 line on top and no outer margin on the right-hand side ((bottom,left,top,right)=(2,3,1,0)). The setting before this parameter change is stored into the variable `op`. Now enter a plot command with `plot()` which plots the sine-function between -2π and 2π. Then plot the cosine function. Then plot the tangent-function between -1.5 and 1.5. What happend to the plots? Recall that every call of `plot.new()` starts a new plot and every high-level plotting command such as `plot()` calls `plot.new()`. In a multi-figure `plot.new()` causes an advance to the next plotting region. Have a try: Enter `plot.new()` and then plot again the sine-function. What happend to the plot? Now restore the old parameter setting by entering `par(op)`. Plot the sine-function again to check that.
Now let's see where low-level potting commands go to in a multi-figure. Enter the command `par(mfrow=c(2,1),oma=c(2,3,1,0)`. Plot the sine-function and add a main title with `title()`. Then plot the cosine-function and add a main title with `title()`. Explain the behaviour of the two calls to `title()`. Next we wish to add text to the sine-plot. For this specify with `par(mfg=c(1,1))` that the next low-level plotting commands affect the figure on top. Add the text 'sine' at a suitable position. Then change to the cosine-plot and add 'cosine' at a suitable position.
Finally mark the margins and the plotting area. Change back to the sine-plot with `mfg=`. The plotting area is surrounded by a green box by entering `box(which="plot",col="green")`. The figure area is surrounded by a turquoise box by entering `box(which="figure",col="turquoise")`. Then enter `box(which="inner",col="blue")` and `box(which="outer",col="red")` to mark the boundaries of the inner and of the outer margins, respectively. Use `mtext()` to write text into the margins. The command `mtext("The sine- and the cosine-function",side=2,line=1,cex=2)` produces the text on the left-hand side. Find the command which produces the text on the right-

hand side (`line=-1`).
Having finished the multiplot, restore the parameter setting with `par(op)`.



Exercise 28: *Download the data file* `ccrt.txt` *from the web page. This file contains the chill coma recovery times (ccrt) for two populations of Drosophila ananassae from Bangkok (BKK) and from Kathmandu (KATH). We thank Sonja Grath for providing this data set.* Produce the following figure:

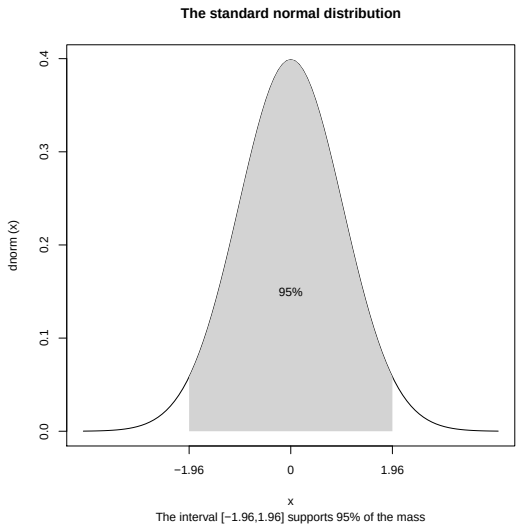


Hints: Do Exercise 27 first, then this exercise is much easier. The vectors which are plotted on the right-hand side are defined as follows. Let `fun1 <- ecdf(ccrt[population=="BKK"])` be the empirical cumulative distribution function of the Bangkok population. Define `k1 <- knots(fun1)` to be the set of jump points of the step function `fun1`. Plot `fun1(k1)` against `k1`. Proceed similarly for the Kathmandu population. The colors are 'red' and 'blue'. The point characters are '19' and '15'. Use `mtext(...,line=-3)` for the main title; outer margins are not needed.

Exercise 29: Get an overview of the data set 'CO2'. Load it with `data(CO2)` if necessary. Produce a coplot which shows the CO2 uptake as a function of the CO2 concentration for each of the 12 plants. Choose plot type "l" or "b". Then produce a coplot which splits up the dependence of the CO2 uptake on the CO2 concentration according to the origin of the plant and according to the treatment, that is, according to every combination of the values of 'Type' and 'Treatment'.

Note that the result of a coplot depends on whether the variable to be conditioned on is a factor variable or a numeric variable. Here is an example where you need to be careful. Download `miete03.txt` from the web page. Produce a coplot which shows the net rent per square meter as a function of the year of construction and split up this dependence according to the number of rooms. Check with `is.factor()` or `class()` whether `rooms` is a factor variable.

Exercise 30: Produce the following figure:



Hints: First plot a the density of the standard normal distribution without axes. Add the y-axis. Add the x-axis and specify where to put the ticks and specify the labels (-1.96,0,1.96) of the ticks. The box around the plot is added with `box()`. The grey area is added with `polygon()` as follows. Define a vector `u` which increases from -1.96 to 1.96. Choose the step size yourself. Define a vector `x` to be the concatenation of `u` and `rev(u)`. So `x` increases from -1.96 to 1.96 and then decreases back to -1.96. Define a second vector `y` to be the concatenation of `rep(0,length(u))` and of `dnorm(rev(u))`. So `y` follows the x-axis as long as `x` increases and then goes back along the graph of `dnorm`. Use `x` and `y` as arguments for `polygon()`. Change the border of the polygon with `border=`.