R Scripting

Exercises for unit 5 - Graphics

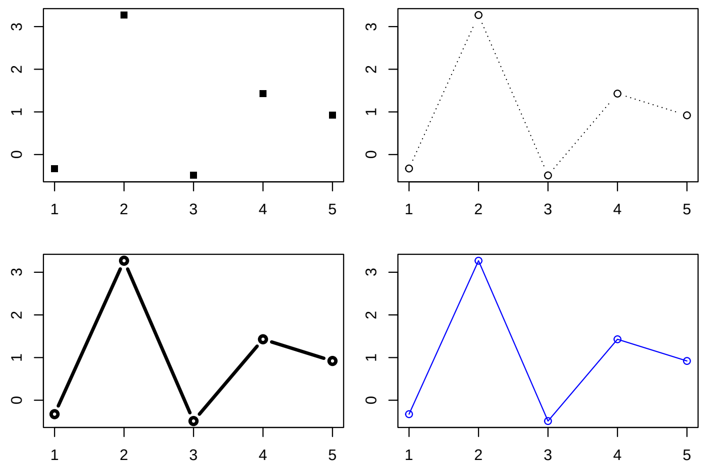
Marcus Wurzer

18. Oktober 2022

Please solve the following problems!

1. Explore the possibilities for different kinds of line and point plots (i.e., variations of the basic scatter-plot). Vary the plot symbol, line type, line width, and color.

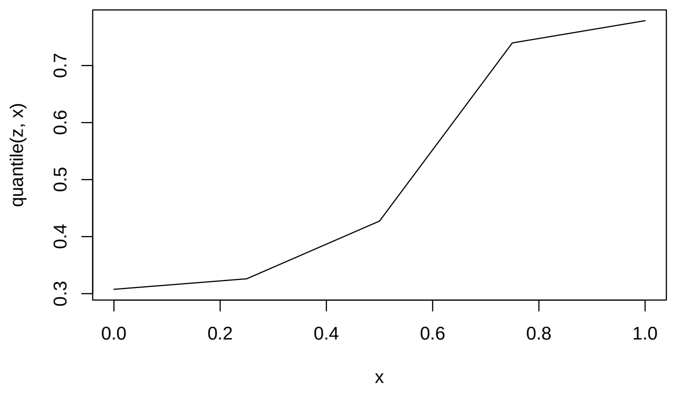
```r
x <- 1:5
y <- rnorm(5, 1)
par(mfrow = c(2, 2), mar = c(3, 2, 1, 1))
plot(x, y, pch = 15) # filled square
plot(x, y, type = "b", lty = "dotted")
plot(x, y, type = "b", lwd = 3)
plot(x, y, type = "o", col = "blue")
```



```r
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)
```

2. Generate a sample vector of 5 random numbers `z` from the uniform distribution and make a line plot of `quantile(z, x)` as a function of x (use `curve`, for instance). What do you notice?

```r
z <- runif(5)
curve(quantile(z, x), from = 0, to = 1)
```

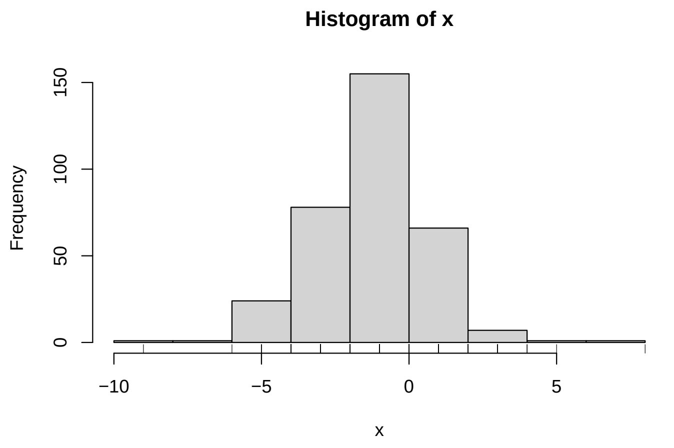


The thing to notice is the linear interpolation between data points.

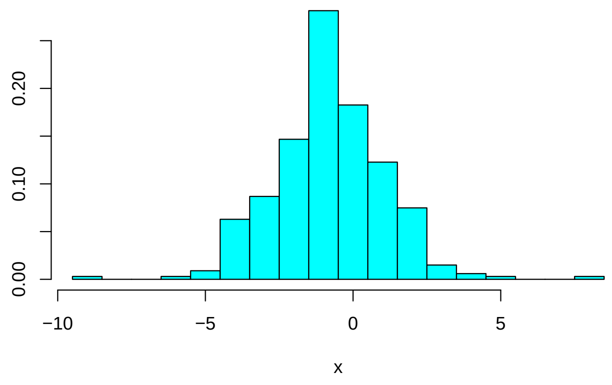
3. Plot a histogram for data set *diffs.txt* (a vector of 334 observations, available within the Moodle course). Since these data are highly discretized, the histogram will be biased. Why? You may want to try `truehist` from the `MASS` package as a replacement.

   The breaks occur at integer values, as do the data. Data on the boundary are counted in the column to the left of it, effectively shifting the histogram half a unit left. The `truehist()` function allows you to specify a better set of breaks.

```
x <- scan("diffs.txt")
hist(x)
rug(x)
```



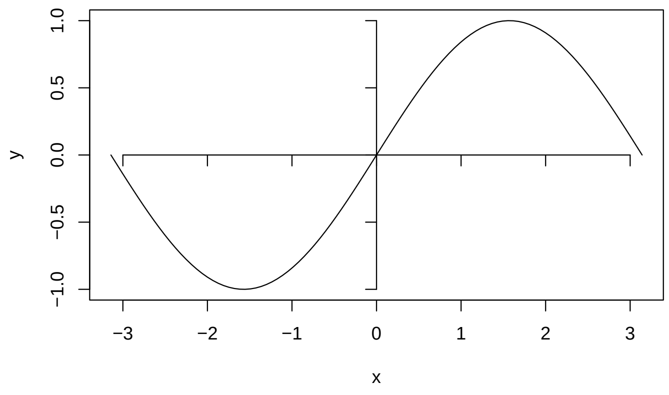
```
library(MASS)
truehist(x, h = 1, x0 = 0.5)
```



4. Given the sine function ($f(x) = sin(x)$) within the range of $[-\pi, \pi]$.

    a. Visualize the graph of the function.
    b. Add a coordinate system crossing the origin using the `axis()` command. The axes should not have numerical annotations (see `?axis`).

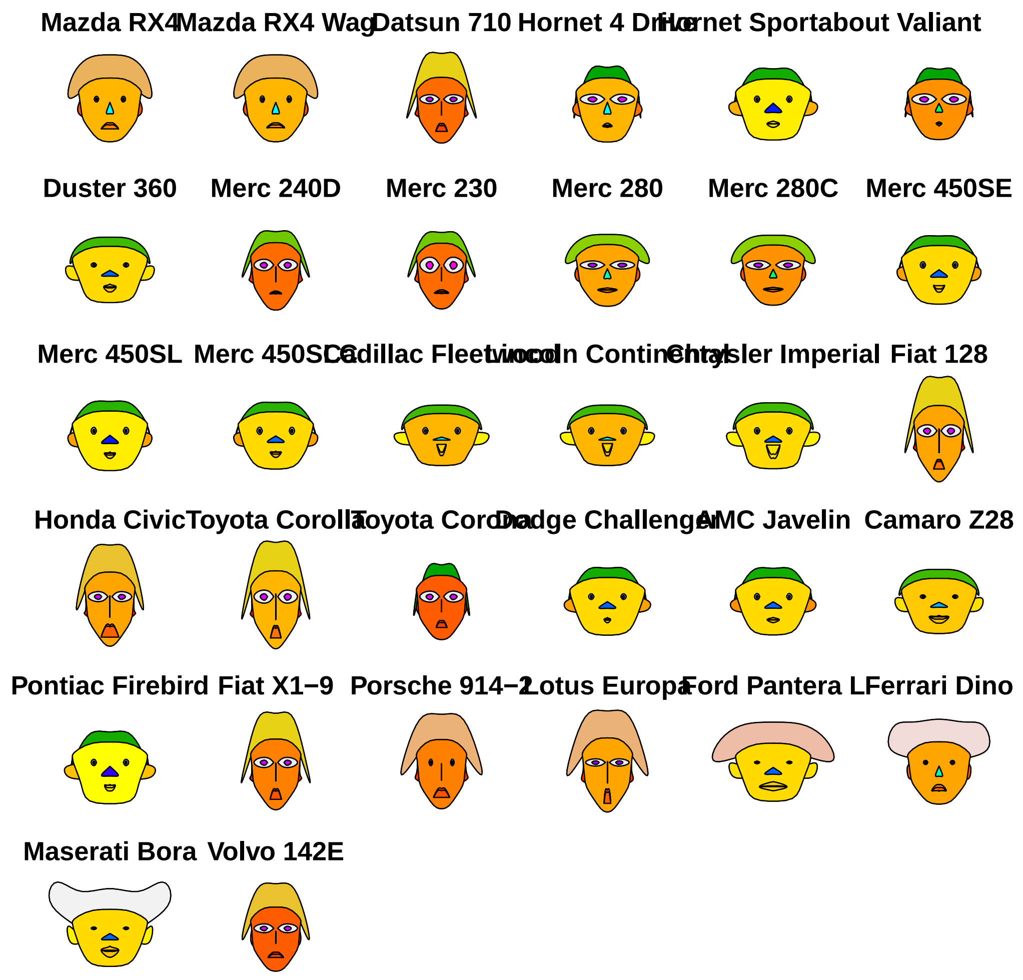
```
# a.
x <- seq(-pi, pi, length = 100)
y <- sin(x)
plot(x, y, type = "l")

# b.
axis(1, pos = 0, labels = FALSE)
axis(2, pos = 0, labels = FALSE)
```



5. Chernoff faces are another example of the *small multiples* technique. Install and load package `aplpack`, then load the `mtcars()` data set and visualize it using the `faces()` function.

```
## Lade nötiges Paket: aplpack
```

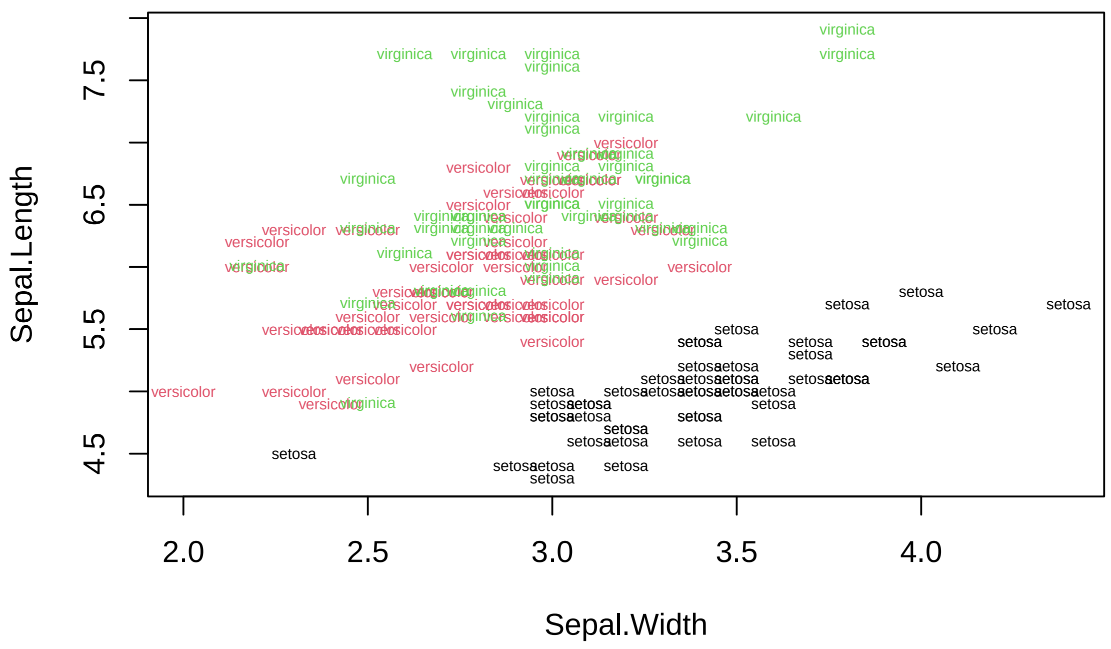
```
library(aplpack)
faces(mtcars)
```



```
## effect of variables:
##  modified item       Var
##  "height of face   " "mpg"
##  "width of face    " "cyl"
##  "structure of face" "disp"
##  "height of mouth  " "hp"
##  "width of mouth   " "drat"
##  "smiling          " "wt"
##  "height of eyes   " "qsec"
##  "width of eyes    " "vs"
##  "height of hair   " "am"
##  "width of hair   "  "gear"
## "style of hair    "  "carb"
## "height of nose   "  "mpg"
## "width of nose    "  "cyl"
## "width of ear     "  "disp"
## "height of ear    "  "hp"
```

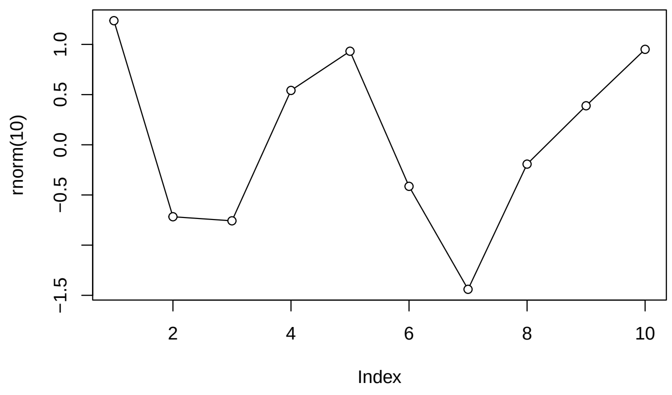
6. Load the `iris` data set and plot `Sepal.Length` vs. `Sepal.Width`, but use the species labels (`Species`) as plot symbols. In addition, each species should have its own color and the size of the text should only be half the default value.

```
plot(Sepal.Length ~ Sepal.Width, data = iris, type = "n")
with(iris, text(Sepal.Width, Sepal.Length, labels = Species,
                col = as.numeric(Species), cex = 0.5))
```



7. If you create a plot like `plot(rnorm(10), type = "o")` with overplotted lines and points, the lines will be visible inside the plotting symbols. How can this be avoided?

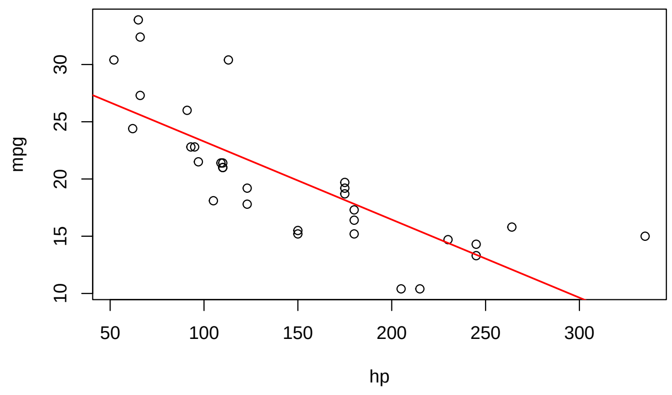
```
# Use a filled symbol, and set the fill colour
# equal to the plot background (here: white):
plot(rnorm(10), type = "o", pch = 21, bg = "white")
```



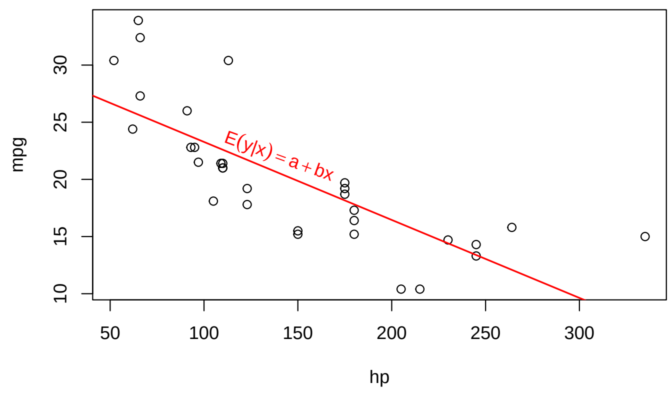
8. Load the `mtcars` data set and regress variable `mpg` on `hp` using `lm(mpg ~ hp, data = mtcars)`. Assign the result to an object.

   a. Plot dependent variable `mpg` versus `hp` and add a **red** regression line of width 1.5 (see the `lwd` argument in `?par`) using the saved regression model.
   b. Use the `text()` command for plot annotation (see `?text` for more information) to add the general regression formula to the plot (Hint: Use `expression(paste(E(y*"|"*x)) == a + b*x)` as text to be written). It should be added **above** the regression line in an area not showing many data points, using the same color as the regression line (red), and rotated by a reasonable degrees value (so that it is roughly parallel to the regression line) using graphical parameter `srt` (see `?par`).
   c. Add vertical lines showing all the residuals (= differences between data points and regression lines) in **blue** (Hint: You get all the estimated values by applying `fitted()` on your model object).
   d. Add **light grey** squares with side lengths equal to the residuals to show the least squares solution visually (Hint: You get all the residuals by applying `residuals()` on your model object). Note: If the squared residuals really appear as squares also depends on your plotting region, e.g., in RStudio, the "Plots" area should have a rectangular form; Using R Markdown, you have to set the chunk options accordingly (e.g., to `fig.height = 7, fig.width = 7`)!

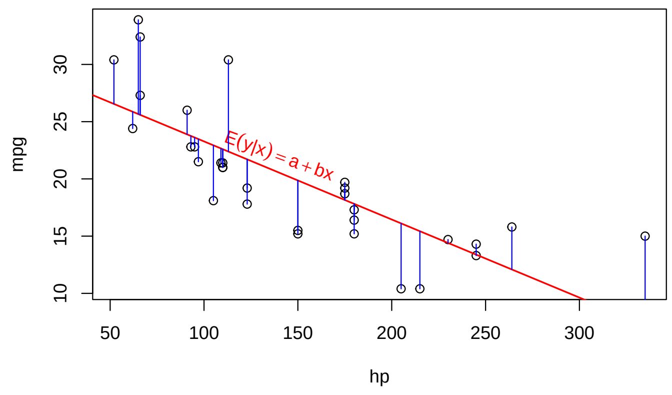
```
# a.
res <- lm(mpg ~ hp, data = mtcars)
plot(mpg ~ hp, data = mtcars)
abline(res, col = "red", lwd = 1.5)
```



```
# b.
plot(mpg ~ hp, data = mtcars)
abline(res, col = "red", lwd = 1.5)
text(140, 22, expression(paste(E(y*"|"*x)) == a + b*x), col = "red",
     srt = -20)
```



```
# c.
plot(mpg ~ hp, data = mtcars)
abline(res, col = "red", lwd = 1.5)
text(140, 22, expression(paste(E(y*"|"*x)) == a + b*x), col = "red",
     srt = -20)
resid.lines <- cbind(x1 = mtcars$hp, x2 = mtcars$hp, y1 = mtcars$mpg,
                     y2 = fitted(res))
apply(resid.lines, 1, function(x) lines(x[1:2], x[3:4], col = "blue"))
```



```
## NULL
```

```
# d.
plot(mpg ~ hp, data = mtcars)
abline(res, col = "red", lwd = 1.5)
```

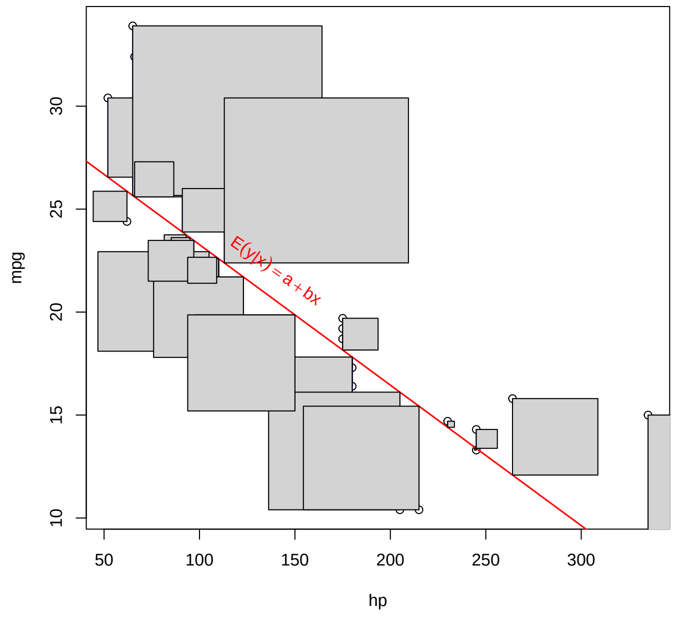
```
resid.lines <- cbind(x1 = mtcars$hp, x2 = mtcars$hp, y1 = mtcars$mpg,
                     y2 = fitted(res))
apply(resid.lines, 1, function(x) lines(x[1:2], x[3:4], col = "blue"))
```

```
## NULL
```

```
# Note: Since x- and y-axis are not on the same scale, we have to adjust
# the residuals that are used to determine the correct "xright"-value
# by the ranges of the variables:
ranges <- c(max(mtcars$mpg) - min(mtcars$mpg),
            max(mtcars$hp) - min(mtcars$hp))
resid.sq <- cbind(xleft = mtcars$hp, ybottom = mtcars$mpg,
                  xright = mtcars$hp + residuals(res) / ranges[1] * ranges[2],
                  ytop = fitted(res))
apply(resid.sq, 1, function(x) rect(x[1], x[2], x[3], x[4],
                                    col = "lightgrey"))
```

```
## NULL
```

```
# Note: We add the text only after the squares have been plotted;
# Since the height and width of the figure are now the same (because we set
# the according chunk options), we also have to change `srt`:
text(140, 22, expression(paste(E(y*"|"*x)) == a + b*x), col = "red",
     srt = -35)
```


E(y|x) = a + bx
mpg
hp
10
15
20
25
30
50
100
150
200
250
300