

Writing your own R scripts

A follow-up to “A (very) short introduction to R”

Claudia Brauer & Paul Torfs

Hydrology and Quantitative Water Management Group,
Wageningen University, The Netherlands

8 August 2017

Introduction

With the exercises in this document you learn how to set up an R script from scratch. The exercises explain step by step how to build your script. The scripts in these exercises always follow the same structure with the following Sections (see Fig. 1):

Initializing Clear memory, set working directory and load packages.

Data Read data from file and data preprocessing (such as extracting and renaming one column of the total data set).

Processing Do computations, run a model, etc.

Output Make a dataframe and/or figure and save these to file.

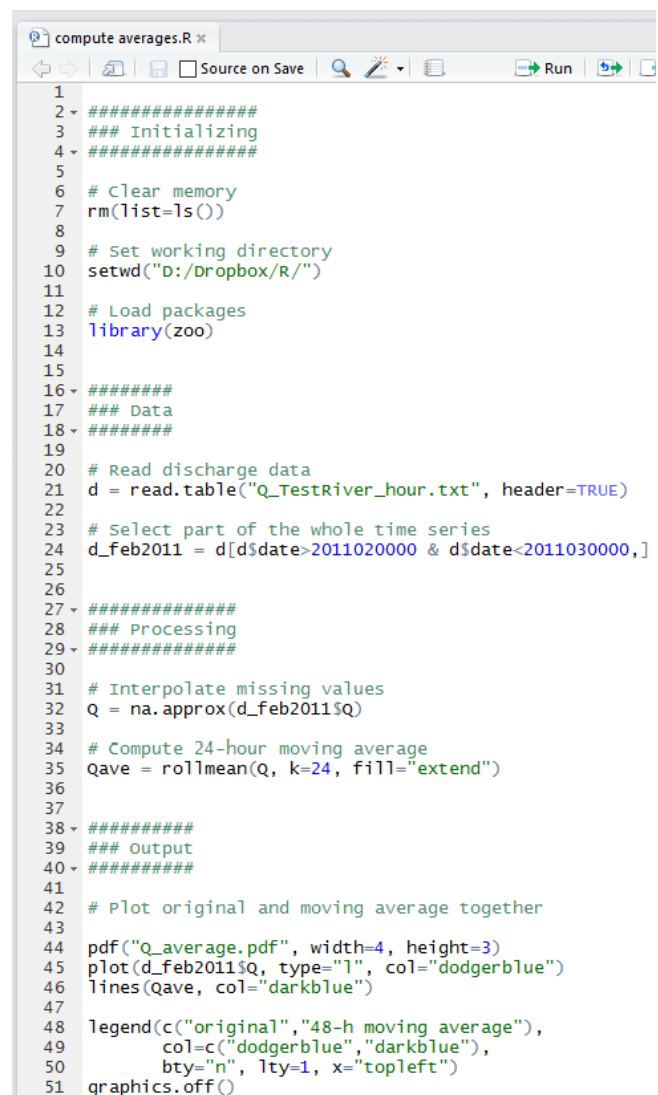
Setting the working directory and saving figures to file can also be done by clicking in RStudio. Hard-coding it in your script requires a few more lines of code, but in real projects, it saves time because you often have to rerun scripts, so it's good to make hard-coding these things a habit.

Before you start, some programming tips:

- Start small.
- Organise your work.
- Add comments (behind #) to your commands (don't overestimate the audience or future-you).
- Run your code (or part of it) every time when you add something, to make sure that it is still working. It is relatively easy to find the error when the code worked before and you just added one line since then.
- Post tips for others on FeedbackFruits Forum.
- Use the references at the end of “A (very) short introduction to R”.
- Copy-paste error messages in Google. You often get hits for StackOverflow, which is a good website with questions and answers on programming

(different languages). You can also Google your problem (type “R” + keyword / problem) and get solutions (functions, packages, code).

- Save the scripts you made as inspiration for your future (thesis) work.



```
1
2 #####
3 ## Initializing
4 #####
5
6 # Clear memory
7 rm(list=ls())
8
9 # Set working directory
10 setwd("D:/Dropbox/R/")
11
12 # Load packages
13 library(zoo)
14
15
16 #####
17 ## Data
18 #####
19
20 # Read discharge data
21 d = read.table("Q_TestRiver_hour.txt", header=TRUE)
22
23 # Select part of the whole time series
24 d_feb2011 = d[d$date>2011020000 & d$date<2011030000,]
25
26
27 #####
28 ## Processing
29 #####
30
31 # Interpolate missing values
32 Q = na.approx(d_feb2011$Q)
33
34 # Compute 24-hour moving average
35 Qave = rollmean(Q, k=24, fill="extend")
36
37
38 #####
39 ## Output
40 #####
41
42 # Plot original and moving average together
43
44 pdf("Q_average.pdf", width=4, height=3)
45 plot(d_feb2011$Q, type="l", col="dodgerblue")
46 lines(Qave, col="darkblue")
47
48 legend(c("original", "48-h moving average"),
49       col=c("dodgerblue", "darkblue"),
50       bty="n", lty=1, x="topleft")
51 graphics.off()
```

Figure 1: Example of an R script in RStudio with the different Sections, some commands and comments.

1 Basic plotting

Compare rainfall measurements

The aim of this exercise is to reproduce Figure 2. You will build it in steps. Don't forget to run (part of) the code every time you add something, to check if everything is still error-free.

1. Download the file `P_gauge_radar.dat` and save it on your computer. Make sure that the folders for scripts, input (data files) and output (figures and data files) are logically structured.
2. Start with an empty script. Make the main headers `Initializing`, `Data`, `Processing` and `Output`). Don't forget the `#`-sign to indicate that they are comments and not commands (see Sec. 5 in "A (very) short introduction to R").
3. Under `Initializing`, add commands to clear R's memory and set the working directory (see Fig. 1).
4. Add code to read the data file (Section Data).
5. Plot the rainfall measured by the radar against rainfall measured by the rain gauge. Look up in "A (very) short introduction to R" (Sec. 6.3) how to select columns in a data frame.
6. Add arguments to the `plot` function:
 - (a) Look in "A (very) short introduction to R" (Sec. 12.1, under Plotting parameters) for the most important plotting parameters.
 - (b) Make your points the color `dodgerblue`. Look up which argument you need.
 - (c) Type "R pch" in Google. Under images, you will find collections of point types. Transform your points into filled circles.
 - (d) Make the axis labels the same as in Fig. 2.
 - (e) Comparing two similar variables is easier when the x and y ranges are the same (and the figure is square, but you'll do that later). Make the ranges of both axes from 0 to 9.
 - (f) You can see that R automatically adds some space on the x and y axis. With the arguments `xaxs="i"` and `yaxs="i"`, you remove that space.
7. Some plotting parameters from the `par`-list have to be specified before the `plot` function.
 - (a) Type `par(pty="s")` above the `plot`-command to make the figure square.
 - (b) Type `?par` to see all plotting parameters in

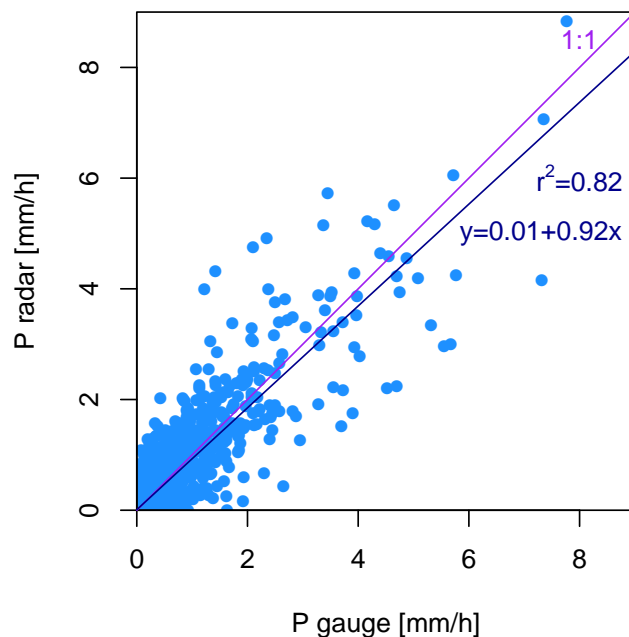


Figure 2: Comparing rainfall measured with a radar to rainfall measured with a rain gauge. Figure source: C.C. Brauer, A. Overeem, H. Leijnse, R. Uijlenhoet (2016): The effect of differences between rainfall measurement techniques on groundwater and discharge simulations in a lowland catchment, *Hydrol. Process.*, 30, 3885–3900, onlinelibrary.wiley.com/doi/10.1002/hyp.10898/epdf.

the R help and look up `mar`. Add this argument to the `par`-function above the plot. Make the bottom and left margins (space around the plot, where axes and labels are drawn) 4 and the top and right margins 0.5.

8. Add the 1:1-line with the function curve:
 - (a) Type `?curve` to see its arguments. `curve` computes the y value with the equation (which uses the x value) specified in the argument `expr`. To compute a 1:1-line, use `expr=1*x`. Add the curve function below the `plot` function.
 - (b) When you run this line, you see that it makes a new plot window instead of adding the line to the previous plot. Find the argument you need to add the line to the plot (and run `plot` and `curve` again).
 - (c) Make the line purple.
 - (d) You see that the line doesn't go all the way from one corner to the other. Specify `xlim` (within `curve`) to make that happen.
9. Compute and plot a regression line:
 - (a) Google "R linear regression". There are several pages that explain how to use the function `lm`. You can also type `?lm` in R.

- (b) Under Processing, type a line of code that fits a linear model and stores the output in a variable (for example called `regr`).
- (c) Type `coef(regr)` on the command line. What you get is a vector with two numbers: the intercept and the slope.
- (d) Add a line to the script to extract the intercept from `regr` using `coef` and regular vector indexing (see Sec. 12.1 in “A (very) short introduction to R”, under Extracting data) and call it `a`.
- (e) Extract the slope and call it `b`.
- (f) Type `summary(regr)` on the command line to see all the output from `lm`. With `summary(regr)$r.squared` you can extract the r^2 . Add a line to the script to do this and call the result `r2`.
- (g) Use the `curve` function again to plot a darkblue regression line using `a` and `b` (Sec. Output). Check if the line is the same as in Fig. 2.

10. Add text.

- (a) In the section Output, below the curves, use the function `text` to add “1:1” in purple on coordinate ($x=8, y=8.5$).
- (b) Add the equation of the regression line in dark blue on coordinate (7.3,5). Instead of typing the text manually, you can use `paste0("y=", a, "+", b, "x")` to get the correct values of `a` and `b` automatically. This is more work, but safer, because if you have to rerun the script with different data, the text is updated automatically. `paste0` glues character strings and variables together without space in between.
- (c) Add the r^2 on coordinate (8,6). Use `bquote(paste("r"~{2}, "=", .(r2)))` as text. This code is a bit complex (and you don't have to understand it), but it allows you to have both a superscript and the value of `r2` automatically.
- (d) It is not necessary (and a bit messy) to show all digits. Round `a`, `b` and `r2` to 2 decimals. Google “R round numbers” to find out which function to use. Add this function to the commands under Processing.

11. Add commands to write the figure to pdf with width 4 and height 4 (inches). You need to add

a command to open the pdf (starting with `pdf`, see the example in Fig. 1) before the plotting commands, and a command to close the pdf afterwards (`graphics.off()`).

2 Pretty plotting

Flow duration curve

The aim of this exercise is to reproduce Figure 3. You will build it in steps: starting with one figure (right panel), for one river (Rhine), with one x axis (bottom).

1. Download `Q_day_Rhine_Meuse_Rietholzbach_Hupsel_1989_2007.txt`.
2. Open an empty script and add the headers Initializing, Data, Processing and Output.
3. Set the working directory (Sec. Initializing).
4. Add lines (Sec. Data) to read the data.
5. Make a new vector with the Rhine discharges ordered from high to low (Section Processing). Divide all discharges in this vector by the mean discharge to get the normalised discharge. Give this vector a logical name.
6. The rest of the script is part of Section Output. Plot the ordered normalised discharges as a red line. The shape should be similar as in the right panel of Figure 3.
7. Make a vector with the same length as the ordered Rhine discharges, ranging from 0 to 1. For example, if there would be 11 discharge data points, this new vector would be `c(0, 0.1, 0.2, ..., 1)`.
8. Modify the plot such that the vector from 0 to 1 is on the x -axis and ordered discharge on the y -axis.
9. Let the x - and y -axis ranges match the example.
10. Add the labels on the x - and y -axes. Google “Stackoverflow subscripts-in-plots-in-r” to find out how to get the subscript in Q_{ave} .
11. Use the function `text` to write Rhine in the plot.
12. Repeat steps 5–11 for the other 3 catchments.
13. Google R `color chart` and look up which colors match the example.
14. Use plotting parameters `mar`, `mgp` and `tcl` to get the same margins, label locations and tick lengths. Search `par` in the R help for explanation. You can type `par(mar=...)` above the first plotting command.

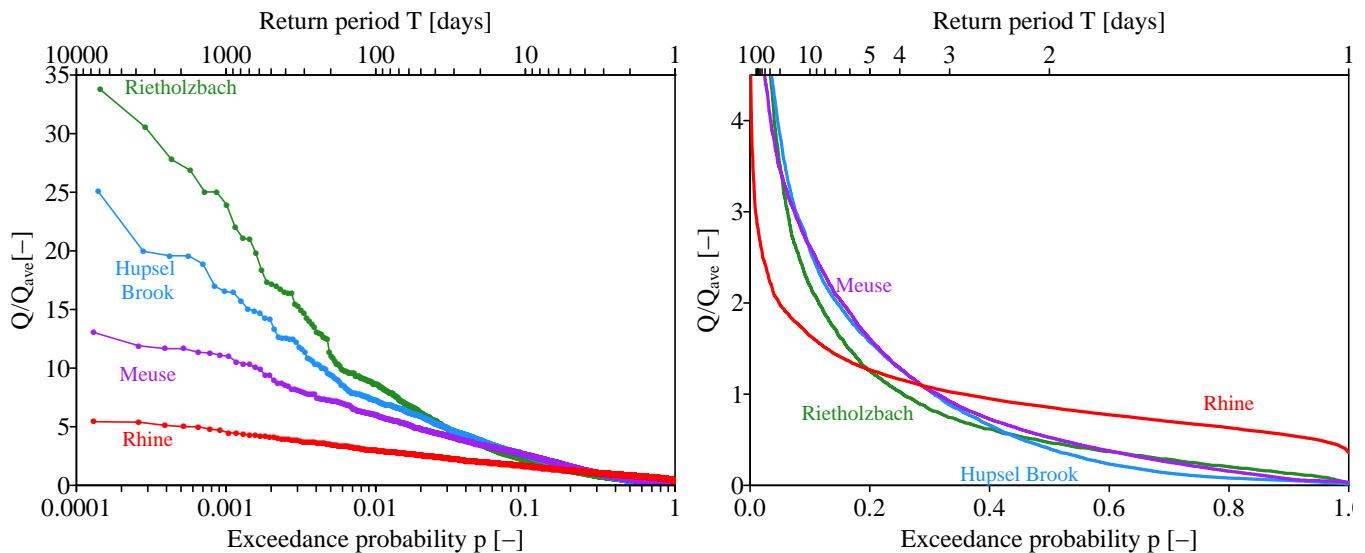


Figure 3: Return period and exceedance probabilities of discharge for four catchments. Figure taken from lecture notes of the course "Water 2", Wageningen University.

15. Make the x -axis on top:

- Type `?axis` to search in the R help which arguments you need.
- Make a vector with numbers 1, 2, 3, 4, 5, 10 and 100. Divide 1 by this vector (to convert exceedance probability to return period; $T = 1/p$).
- Use the function `axis` to make the axis on top with tick marks and numbers at return periods 1, 2, 3, 4, 5, 10 and 100.
- Make a vector from 20 to 90 with steps of 10. Make another vector from 6 to 9 with steps of 1. Combine the vectors. Divide 1 by this vector.
- Use the function `axis` to make an axis on top with 12 tick marks on the locations of the last vector and no numbers. Use the function `rep` to repeat an empty character string ("") 12 times.

16. The right figure is now finished. Copy the code for the figure (except the plotting parameters) and paste it above its duplicate (still in Section Output), after the plotting parameters. Add extra comments to clarify the difference.

17. Add argument `mfrow` to the plotting parameters to get two figures next to each other.

18. Add an argument to the `plot` function to make the bottom x -axis of the left figure logarithmic.

19. Let the x and y ranges match the example.

20. Add an argument to the `plot` function suppress drawing the bottom x -axis.

21. Make the bottom and top x -axes in a similar way as the top x -axis in the right figure.

22. Move the text in the left graph to the correct position.

23. Add commands to write the figure to pdf with width 8.8 and height 3.6. Add argument `family="Times"` to change the font. Add the command to close the pdf after the figure.

3 Reading data files

Process groundwater diver data

In this assignment you will read data from a less organised data file, process and plot them as in Fig. 4.

- Download `Diver_piezometer.MON`. This file contains pressure data from the water column and air above a sensor in a groundwater well (also called piezometer). Open it in Notepad to see what it looks like.
- Initialize the script with standard headers and working directory.
- Add the command `Sys.setenv(tz='UTC')` to set the system's clock to UTC. This is not always necessary, but it's often the safe option (to avoid trouble with time zones or daylight savings time).
- Add lines (Sec. Data) to read the data. Use an extra argument to prevent R from reading the text before the data begins.
- Use the function `head` to show the first rows of the data frame. Check if those are the same lines as in the file.

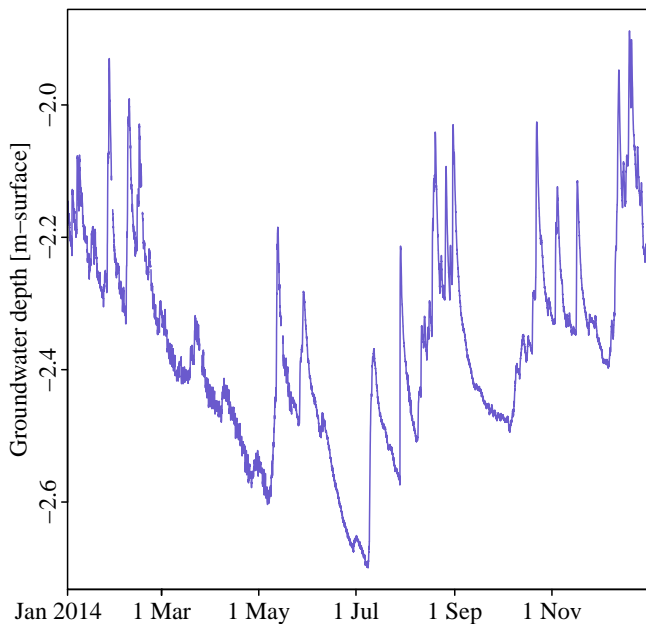


Figure 4: Return period and exceedance probabilities of discharge for four catchments. Figure taken from computer practical in the course “Field practical hydrology, water quality and meteorology”, Wageningen University.

6. Make three vectors, each consisting of one of the columns in the matrix, and give them four logical names. You can ignore the last column.
7. Check the class of each vector. Look up in “A (very) short introduction to R” what is meant with classes and how to check it.
8. Use the function `paste` to combine the date- and time vectors to one date-time vector. Its class should now be character.
9. Convert this date-time vector (which is still a character) to a POSIX class. Look up in “A (very) short introduction to R” what is meant with POSIX and how to convert character strings to it. Add `tz="UTC"` to set the timezone.
10. Make 2 new vectors which only contain the year 2014 from the larger date-time vector and the data series:
 - (a) Use a similar code to make two variables with POSIX class: one for the start date and one for the end date.
 - (b) Select the part of the vector where the date-time is larger than the start date and smaller than the end date. Something similar is done in Fig. 1
11. Repeat steps 5–11 for `Diver_baro.MON`. This file contains atmospheric pressure data (the air above a sensor which is not submerged a piezometer).

12. Under Processing, subtract the atmospheric pressure from the pressure from the piezometer to get the pressure of the water column in the piezometer. Make a quick plot to check your values.
13. Convert the pressure time series (p) to water levels above the sensor (h) according to $h = p \times 100 / 9.81$.
14. The sensor is 3.46 m below soil surface. Convert the water columns above the sensor to m below surface. Make a quick plot to check your values.
15. Remove strange values.
16. Under Output, plot the water level as a line with a pretty colour.
17. Modify the plot to get the time on the x -axis.
18. Make a vector with the dates belonging to the labels on the x -axis and a vector with the text you want to plot at these tick marks.
19. Add an argument to the plot function to suppress the default x -axis (search in the list with plotting parameters: `?par` or Google it). Then plot the x -axis separately with your customized labels.
20. Add a label on the y -axis.
21. Force the x -axis to range from the first measurement to the last using argument (use the whole horizontal space instead of the default white spaces). Search in `par` again.
22. Above the figure, add a line with general plotting parameters for margins, tick marks, etc. Also include an argument to make the figure square.
23. Add commands to write the figure to a pdf of 4.4x4.4 inch, with Times font. Don't forget the command to close the pdf after the figure.

4 Matrix operations

Compute statistics for multiple modelled discharge time series

In this assignment you will reproduce the top panel of Figure 6. This panel shows the range of simulated discharges when a model (WALRUS) is run with different parameter sets.

1. Download `Qmod_parameters.dat` and `output_WALRUS.dat`. The first file contains a matrix with time steps in rows and different model runs in columns, from the period 1 May–30 June 2012. The second

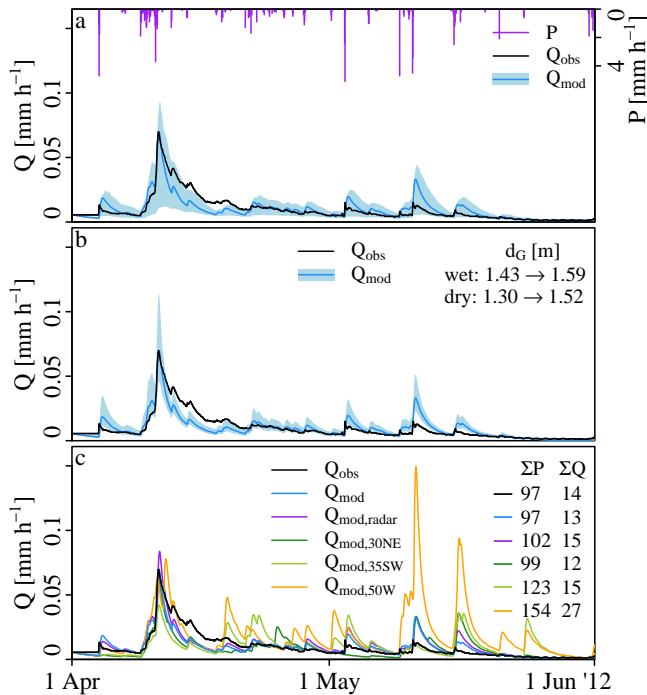


Figure 5: Simulated discharge with different parameter sets (top), initial conditions (middle) and rainfall input (bottom). Figure source: C.C. Brauer, P.J.J.F. Torfs, A.J. Teuling, R. Uijlenhoet (2014b): The Wageningen Lowland Runoff Simulator (WALRUS): application to the Hupsel Brook catchment and Cabauw polder, Hydrol. Earth Syst. Sci., 18, 4007-4028, www.hydrol-earth-syst-sci.net/18/4007/2014/hess-18-4007-2014.pdf

contains input and output of one of the model runs as well as observed precipitation (P) and discharge (Q_{obs}) of the whole of 2012.

2. Initialize the script with standard headers and working directory.
3. Read the data files. Make a small data frame with only the period 1 May–30 June 2012 from the output_calH_20112012.dat file.
4. Compute the mean discharge for each time step (averaged over all runs). Don't use a for-loop, but look for a standard function in the reference list of "A (very) short introduction to R". Give the resulting vector a logical name.
5. Compute the 10th and 90th percentile for each time step, resulting in two vectors. You need two standard functions to do this: one for the percentiles and one called `apply`.
6. Plot the observed discharge as a black line and add the mean modelled discharge as a blue line. You don't need x -values, because the data points are equidistant.
7. Plot the light blue range:

- (a) Google how to get a transparent version of the shade of blue you chose.

- (b) Combine the 10th and 90th percentile in one vector, where you reverse the order of the 90th percentiles (so from the last point in the time series to the first).
- (c) Make two vectors: from one to the number of datapoints and the same vector backwards. Then combine the vectors into one vector, which you will use as x -values for the percentile range.
- (d) Use the function `polygon` to plot the light blue range. Use the two vectors you created as x - and y -values.

8. Add the rainfall data to the plot. Use `par(new=TRUE)` to plot these over the previous plot. Search which arguments you need to add to `plot` to get vertical lines and suppress plotting x - and y -axes.
9. Add x -axis and y -axis on the right separately. Also add axis labels.
10. Add the legend (only the lines).
11. Add a light blue rectangle behind the blue line for the modelled discharge in the legend. Google "rectangle R" or similar to find the command.

5 GIS

Draw catchment map

In this assignment, you will process and plot shapefiles and raster (ascii grid) data. Shapefiles are actually a collection of files together specifying the coordinates of a polygon (for the border of the Netherlands) or lines (for water courses). ascii files contain information about the resolution and coordinates, and a matrix with a value per grid cell. There are 3 ascii files: for the surface elevation, the elevation of the clay layer (bottom of the top aquifer) and a mask for the catchment area, which distinguishes between inside (1) and outside the catchment (NA). Finally, a regular data file contains coordinates for some specific locations.

1. Download the zip file `shapefiles_Hupsel.zip` and unzip it in a folder. Open the ascii files to see what they look like.
2. Initialize the script with standard headers and working directory. Add a line to load package `maptools`. You probably have to install this package first (you don't need to add that to the script, because you only have to do that once).

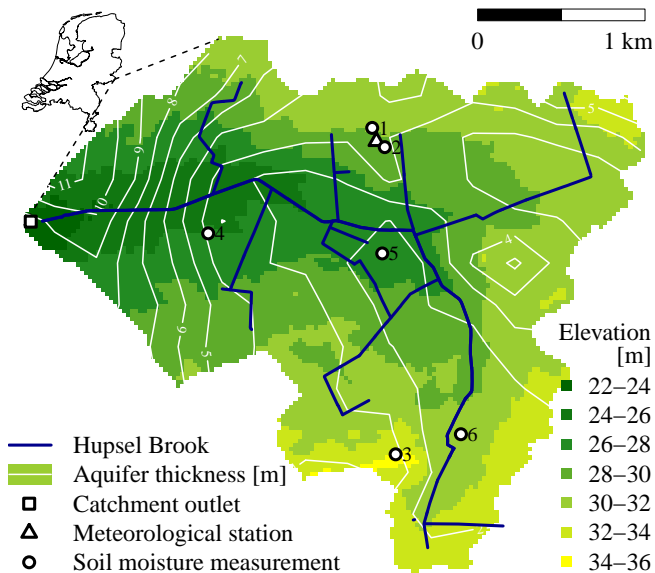


Figure 6: Hupsel Brook catchment. Figure source: C.C. Brauer, A.J. Teuling, P.J.J.F. Torfs, and R. Uijlenhoet (2013): Investigating storage-discharge relations in a lowland catchment using hydrograph fitting, recession analysis, and soil moisture data, *Water Resour. Res.*, 49, 4257–4264, onlinelibrary.wiley.com/doi/10.1002/wrcr.20320/pdf.

3. Use function `readAsciiGrid` to read the ascii data files (Sec. Data). Add argument `as.image=TRUE` so it returns a list with components `x` (`x`-coordinates), `y` (`y`-coordinates) and `z` (matrix with elevation/mask per pixel).
4. Use function `image` to plot a first elevation map, using default colors and ranges (Sec. Output).
5. Cut off the elevation map at the catchment boundary (Sec. Processing). Use square brackets to replace the elements in the elevation-matrix with NA if that same element in the mask-matrix is NA. The elevation-matrix and mask-matrix are in fact the `z`-component in the lists. Use “A (very) short introduction to R” to find out how to check if a value is NA.
6. Make the plot again to see if everything works.
7. Above the plot, set all margins to zero.
8. Make a vector with 7 colors green, either by hand (google R color chart) or by using `colorRampPalette` for a calculated range.
9. Make a vector of length 8 with the break points of the color legend. For example, all pixels with elevation between the first and second break points will get the first color. Note that the elevations in the matrix are in cm above sea level.
10. Add the colors and break points as arguments to

the `image` function.

11. Add the legend in the bottom right. Add the words Elevation and [m] manually with the function text.
12. Compute the thickness of the aquifer by subtracting the elevation of the clay layer from the elevation of the land surface (Sec. Processing). The easiest way is to make a copy of the elevation-list and adapt the `z` element. Note that the clay elevations are in m above sea level.
13. Use function `contour` to add clay elevation as white lines. Use argument `add=TRUE` to plot it over the previous map and plot lines every 1 m.
14. Read the shapefiles with `readShapePoly` or `readShapeLines` (Sec. Data). You only need one command to read all files starting with the same name together (ending with `.shp`).
15. Add the primary and secondary water courses as lines to the plot.
16. Read the file `coordinates.dat` (Sec. Data).
17. Add the locations of the outlet, meteorological station and soil moisture observation locations as points to the plot. Use different symbols (`pch`) for different measurements. To get a white-filled triangle, plot a solid white triangle and plot an open black triangle over it. Use arguments `cex` and `lwd` to change the size of the symbols.
18. Add numbers to the soil moisture observation locations as text.
19. Add the legend in the bottom left. Make vectors for `pch`, `lty`, `lwd` and `col` with NAs when that value does not apply.
20. Plot a green rectangle behind the white line for the aquifer thickness.
21. Add a scale bar with rectangles and text.
22. Add a small plot of the Netherlands:
 - (a) Below the map you already created, you will plot another map, but only using the top left part of the plotting area. Add `par(fig=c(...), ...)`. Look in the help for `par` how to use the argument `fig` to reduce the plotting area.
 - (b) Plot the map of the Netherlands with `plot`. Use `bg="transparent"` to avoid white overlaying the map.
 - (c) Before the `fig` command, add dashed lines from the location in the Netherlands to the corners of the large map.