Introduction to R: Session 01

Holger Sennhenn-Reulen[a], Nordwestdeutsche Forstliche Versuchsanstalt (NW-FVA)

November 02, 2021 (Version 0.4)

Contents

[a] Private webpage: uncertaintree.github.io

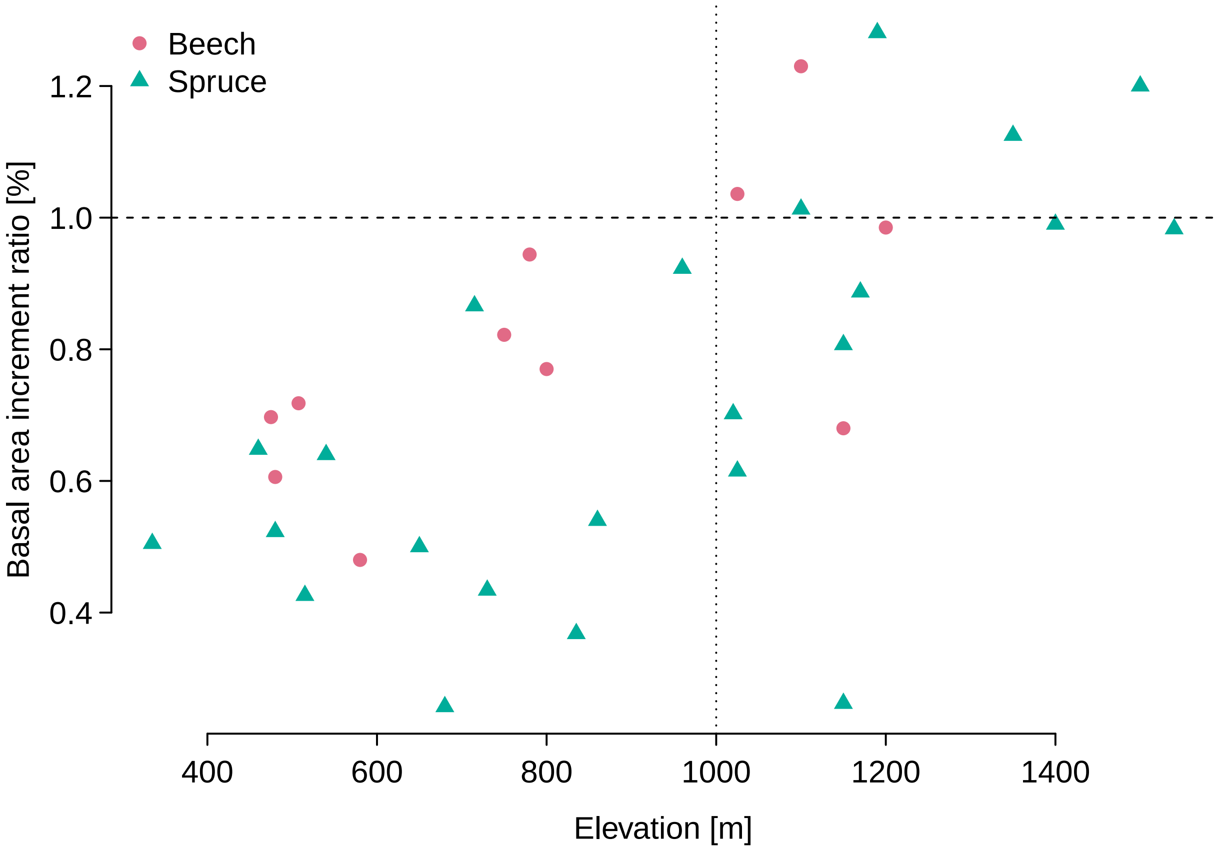
1 (Int)R(o)

1.1 What is R?

'A Language for Data Analysis and Graphics'

A first example: **Scatter plot, data-management and descriptive analysis**

```
## A scatter-plot:
paint <- colorspace::qualitative_hcl(n = 2)
par(mar = c(3, 3, .1, .1), mgp = c(2, .5, 0), tcl = -.3)
plot(drought$elev, drought$bair, las = 1, bty = "n",
     xlab = "Elevation [m]", ylab = "Basal area increment ratio [%]",
     pch = c(16, 17)[1 + (drought$species == "Spruce")],
     col = paint[1 + (drought$species == "Spruce")])
abline(h = 1, lty = 2)
abline(v = 1000, lty = 3)
legend("topleft", pch = c(16, 17), col = paint, legend = c("Beech", "Spruce"),
       bg = "white", bty = "n")
```




```
## ... some data-management:
range(drought$elev)

## [1]  335 1540

(tmp <- range(drought$elev %/% 250))

## [1] 1 6

br <- seq(tmp[1] * 250, (tmp[2] + 1) * 250, by = 250)
(drought$elev_cut <- cut(drought$elev, breaks = br))

##  [1] (250,500]          (250,500]          (250,500]          (500,750]
##  [5] (500,750]          (500,750]          (500,750]          (500,750]
##  [9] (500,750]          (750,1e+03]        (750,1e+03]        (750,1e+03]
```

```
## [13] (1e+03,1.25e+03]   (1e+03,1.25e+03]   (1e+03,1.25e+03]   (1e+03,1.25e+03]
## [17] (1e+03,1.25e+03]   (1e+03,1.25e+03]   (1e+03,1.25e+03]   (1.25e+03,1.5e+03]
## [21] (1.25e+03,1.5e+03] (1.25e+03,1.5e+03] (1.5e+03,1.75e+03] (250,500]
## [25] (250,500]          (500,750]          (500,750]          (500,750]
## [29] (750,1e+03]        (750,1e+03]        (1e+03,1.25e+03]   (1e+03,1.25e+03]
## [33] (1e+03,1.25e+03]   (1e+03,1.25e+03]
## 6 Levels: (250,500] (500,750] (750,1e+03] ... (1.5e+03,1.75e+03]
```

```
## ... and some descriptive statistics:
library("plyr")
ddply(.data = drought, .variables = c("species", "elev_cut"), summarize, .drop = F,
      n = length(bair),
      mean_bair = mean(bair))
```

```
##    species          elev_cut n mean_bair
## 1    Beech         (250,500] 2 0.6515000
## 2    Beech         (500,750] 3 0.6733333
## 3    Beech       (750,1e+03] 2 0.8570000
## 4    Beech  (1e+03,1.25e+03] 4 0.9827500
## 5    Beech (1.25e+03,1.5e+03] 0       NaN
## 6    Beech (1.5e+03,1.75e+03] 0       NaN
## 7   Spruce         (250,500] 3 0.5586667
## 8   Spruce         (500,750] 6 0.5205000
## 9   Spruce       (750,1e+03] 3 0.6103333
## 10  Spruce  (1e+03,1.25e+03] 7 0.7952857
## 11  Spruce (1.25e+03,1.5e+03] 3 1.1050000
## 12  Spruce (1.5e+03,1.75e+03] 1 0.9830000
```

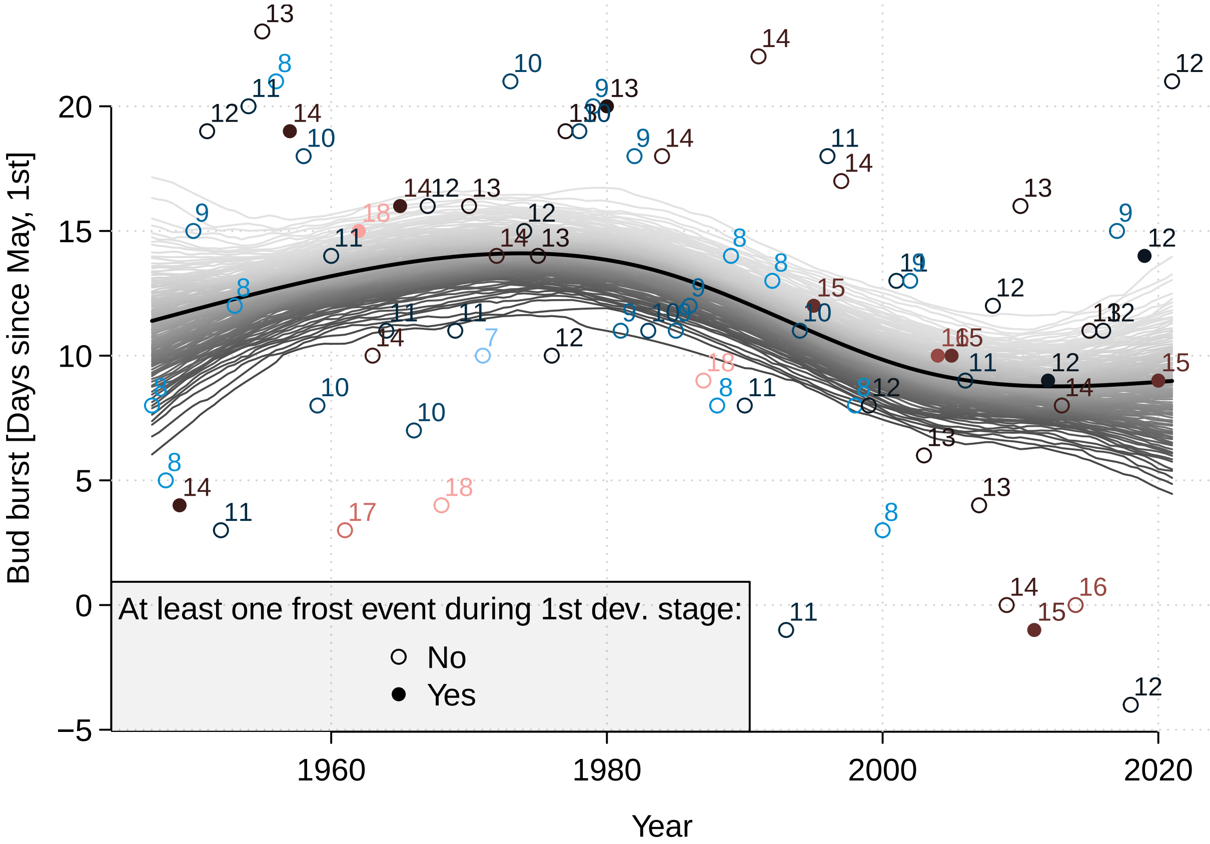
1.2 Why useR?

Because of packages such as mgcv! ... and because we can organize our whole working-with-data-process reproducibly (markdown!) in one place!

(Note: the following figure is much too overloaded!)

```
par(mar = c(3, 3, .1, .1), mgp = c(2, .5, 0), tcl = -.3)
plot(frost$year, frost$bud_burst_days_since_may1st, type = "n",
     xlab = "Year", ylab = "Bud burst [Days since May, 1st]",
     las = 1, bty = "n")
grid()
legend("bottomleft", pch = c(1, 16), legend = c("No", "Yes"),
       title = "At least one frost event during 1st dev. stage:",
       bg = rgb(.5, .5, .5, alpha = .1))
## library("mgcv")
m <- mgcv::gam(bud_burst_days_since_may1st ~ te(year), data = frost)
nd <- data.frame(year = seq(min(frost$year), max(frost$year), by = .5))
br <- mgcv::gam.mh(m, thin = 5, ns = 2000, rw.scale = 2)$bs
coda::effectiveSize(coda::as.mcmc(br))
Mu <- predict(m, newdata = nd, type = "lpmatrix") %*% t(br)
Mu_q <- apply(X = Mu, MAR = 1 , FUN = quantile,
              probs = seq(.01, .99, by = .01))
paint <- colorspace::sequential_hcl(n = nrow(Mu_q), pal = "Light Grays")
for (i in 1:nrow(Mu_q)) {
  lines(nd$year, Mu_q[i, ], col = paint[i])
}
lines(nd$year, predict(m, newdata = nd), lwd = 2)
tmp <- as.numeric(frost$end_1st_dev_stage - frost$bud_burst)
tmp <- tmp - min(tmp) + 1
paint <- colorspace::diverging_hcl(n = max(tmp), pal = "Berlin")
points(frost$year, frost$bud_burst_days_since_may1st, col = paint[tmp],
       pch = c(1, 16)[1 + (frost$n_frost > .5)])
```

```
text(frost$year, frost$bud_burst - frost$may1st,
     labels = frost$end_1st_dev_stage - frost$bud_burst,
     adj = c(-.1, -.5), cex = .8, col = paint[tmp])
```



1.3 Literature

1.3.1 Books...

- Everitt, Hothorn (2006): *A Handbook of Statistical Analyses using R*. Chapman and Hall.
- Ligges (2008): *Programmieren mit R*. Springer Verlag.
- Venables, Smith (2002): *An introduction to R*. Network Theory Verlag.
- Verzani (2005): *Using R for Introductory Statistics*. Chapman and Hall.
- Wickham (2016): *Advanced R*.

1.3.2 ... and the wide web:

- education.rstudio.com
- cran.r-project.org manual R-intro
- cran.r-project.org manual R-lang

1.4 R and RStudio

R:

- https://r-project.org
- Freely available
- Supported by all major operating system: Windows, Linux/Unix, Mac OS, ...

RStudio and others editors for working with R:

- **RStudio** for Windows, Mac OS and Linux:
- RStudio website
- RStudio cheat sheet
- Previously, I presented a list of alternatives to RStudio, but in 2021, that doesn't seem suitable anymore!?

1.5 R is a language

- 'To understand computations in R, two slogans are helpful: Everything that exists is an object. Everything that happens is a function call.' (John M. Chambers)
- Interpreted language: *R* interprets and evaluates your function calls without a compilation step.
- Simple syntax with clear similarities to mathematical notation.

1.6 R is an environment for statistical computing

- rich toolbox for doing (graphical) statistics with an ever growing list of ever improving 'add-on' packages.
- R is freely available for anyone.
- R code is the product: it is transparent and allows you to reproduce any single step of your analysis.

1.7 A short history of R

- 1976: Development of programming language S at Bell Laboratories
- 1988: Software S-PLUS released (for purchase)
- 1992: Ross Ihaka and Robert Gentleman start project R
- 1995: R available for free under GPL
- 1998: Comprehensive R archive network (CRAN) is founded
- 2000: First 'complete' version R-1.0.0 released
- 2004: First conference on R (useR!) takes place
- 2021: Current version is *R-4.1.2* (November 2021)

2 Basic vector types

In R, there is no such thing as a single number, since in R, a single number is just a vector of length 1:

```
length(3)
```

```
## [1] 1
```

R works on six basic vector types, of which four a used in our daily work as empirical researchers: `logical` (`TRUE`and `FALSE`), `integer` (numbers we know as `...`, `-1`, 0, 1, 2, `...`; explicitly created in R with suffix `L`), `real` (extending integers to 'any' value such as `1/3`), and `string` (`'R'`, `'Beech'`, `...`).

```
class(TRUE)
```

```
## [1] "logical"
```

```
class(-1L)
```

```
## [1] "integer"
```

```
class(1/3)
```

```
## [1] "numeric"
```

```
class("R")
```

```
## [1] "character"
```

Again, the difference between integers and reals is not important for applied work, so just think of `integer` and `real` as class that represent numerical values in a computer. The following section introduces some manipulations we can do with them, `logical` and `string` will be treated later.

3 Basic mathematical commands

Mathematical operation	Command
Addition	+
Subtraction	-
Multiplication	*
Division	\
Exponentiation	^
Rest of integer division (Modulo)	%%
Integer division	%\%
Brackets	()

3.1 Exercise

Run the following lines.

Note: First and third line define objects `a` and `b` by using an indexing command on data-object `drought`. You don't need to fully understand what this doeas at this point of the course, just think that objects `a` and `b` each represent a single number with some applied meaning (Basal area increment ratio at 1st, and 2nd plot, respectively).

```
a <- drought$bair[1] ## Basal area increment ratio at 1st plot
a ## ... growth is about a half in 2003 in comparison to growth in 2002
```

```
## [1] 0.505
```

```
b <- drought$bair[2] ## Basal area increment ratio at 2nd plot
b ## ... growth is about two thirds in 2003 in comparison to growth in 2002
```

```
## [1] 0.648
```

```
a - b ## Whats the difference of bair?
```

```
## [1] -0.143
a / b ## Whats the ratio of bair?
## [1] 0.779321
b %/% a ## How often is 'b' contained in 'a'?
## [1] 1
b %% a ## If 'b' is contained once in 'a', what's remaining?
## [1] 0.143
(a + b)/2 ## Arithmetic mean of 'a' and 'b'
## [1] 0.5765
```

Did we just calculate an arithmetic mean 'by hand'?

```
mean(c(a, b))
## [1] 0.5765
```

4 Symbols and values

Objective	Call
Decimal sign	`.`
List and seperate objects, arguments, ...	`,`
Several *R*-calls in one line (not recommended)	`;`
'from-to' operator for integer sequences	`:`
Comments and help	`#`, `?`
Number π	`pi`
A concept called infinity	`Inf`
Base 10 exponential notation (eg. $10^{-3} = 0.001$)	`1e-3`
Integer value	`L`
Empty object	`NULL`
Missing value (*not available*)	`NA`
Non-defined value (*not a number*)	`NaN`
Comparison	`>`, `>=`, `==`, `!=`, `<=`, `<`
Boolean	`TRUE`, `T`, `FALSE`, `F`
Negation (*not*)	`!`
Conjunction (*and*)	`&`
Disjunction (*or*)	`|`

Remember, everything in R is an object or a function, so we even can't use `,` without a function call such as:

```
c(a, b) ## first two bair values in a vector
```

```
## [1] 0.505 0.648
```

4.1 Exercise

Run the following lines and ask yourself *What is the applied question behind each line?*.

```
pi ## Okay, there is no applied meaning here, it's just a number, an important one, though :)
a / b == pi
a > b
a < b
a == a & b != 2/3
a > b | b < 2/3
a / 0
0 / 0 ## Let: x = 0 / 0. So: 0 * x = 0, which is true for any number x.
```

5 Mathematical functions

Mathematical function	Call
Exponential function with basis e	`exp()`
Natural logarithm	`log()`
Square root	`sqrt()`
Absolute value	`abs()`
Trigonometric functions	`sin()`, `cos()`, `tan()`
Sum and product	`sum()`, `prod()`
Round (up and down)	`round()` (`floor()`, `ceiling()`)
Maximum and minimum value	`max()`, `min()`
Factorial $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$	`factorial()`
Binomial coefficient $\binom{n}{k}$	`choose()`

... for further functions, see `?Special` and `?groupGeneric`.

Also note: There are only countable many numbers that a computer is able to represent. Therefore, computers can also only represent some real numbers, and as a consequence, 'rounding' is more complicated than we might think! Here are some explanations why R results in `0 2 2 4 4 6 6` instead of `0 1 2 3 4 5 6` when it does `round(seq(0.5, 6.5))`. Many thanks to Jan Schick for pointing me towards this 'rounding' standard in R, as well as providing me this nice example. Session 05 will also briefly touch upon R and computers and the 'density' of real numbers.

5.1 Exercise

Run the following lines.

```
a <- drought$bair[1]; a; b <- drought$bair[2]; b ## Hoehen erster und zweiter Baum
exp(x = a)
sqrt(x = a)
a - b
abs(x = a - b)
```

(Computers construct real values)[https://en.wikipedia.org/wiki/Floating-point_arithmetic]:

```
sin(x = pi)
```

```
## [1] 1.224647e-16
```

```
round(sin(pi), digits = 5)
```

```
## [1] 0
```

```
cos(x = pi)
```

```
## [1] -1
```

```
sum(a, b)
max(a, b)
```

6 Functions basics

(We will have a more detailed focus on functions in the third(?) session.)

Basic properties of functions in R:

- Function call of function `f` using brackets: `f()`
- Documentation (*'help-page'*) using preceded question mark (`?f`), or calling `help(f)`
- Search for documentation of `f` by: `help.search('f')`
- Syntax:

```
f(arg1 = wert1, arg2 = wert2, ...) ## '...' is for optional further arguments.
```

- Arguments often have default objects, eg. for `pch` in `plot()`:

```
plot(x = a, y = b)
```

```
plot(x = a, y = b, pch = 1) ## same!
```

- Giving objects to arguments is not optional in any case:

```
sin(pi/2)
```

```
## [1] 1
```

```
sin(x = pi/2) ## Same
```

```
## [1] 1
```

```
plot(pi, pi, 1) ## Error!
```

```
## Error in plot.xy(xy, type, ...): ungültiger Plottyp
```

7 Documentation

Structure:

- **Description**: In brief, what is this function about?
- **Usage**: How to call the function? What are mandatory arguments?
- **Arguments**: Explaining each argument
- **Details**: Some text about implementation, scope, ...
- **Value**: Explaining the resulting object
- **Authors** and **References** and **See also** and ...
- **Examples**: Always at the bottom, always scroll down there, you will never be disappointed!

7.1 Exercises

Go to the documentation for `lm`.

- What is this function about?
- Copy-paste and run the example.

8 Objects

Basic Conventions:

- Objects store values, results or algorithms, ie. functions.
- Assignment of contents by <- or = (or very seldomly ->).
- Type of contents is described by object classes.

```
names(drought)
```

```
## [1] "bair"     "elev"     "species"  "elev_cut"
```

```
#rm(spati2)
#(a <- d$h[1]); (a = d$h[1])
#log(b <- d$h[2]); b
#(a <- b)
```

8.1 Objekt names

There are few hard technical restrictions on how to name objects in R, and a few soft rules that make life much simpler:

- Object names are requared to begin with a lower or upper case letter, no numbers or any other signs are allowed.
- As a second or any trailing character, numbers and some signs (restrict yourself to underscore _ and .) are allowed
- Don't use ?, $, %, ^, &, *, (,), -, #, ?,,, <, >, /, |, \, [,] ,{, and @
- **As short as possible, as long as needed!**
- This trade-off is simple, but (hopefully) still very useful: long object names mean more typing (RStudio weakens this point by auto-complete), but leave less room for questions on content (*What was a again?*).
- So try to give 'telling names', ie. contents (and units!) should be clearly visible from the object's name: `first_bair_measurement` is better than `a`, `elev_m` is better than `elev`.
- Ask yourself: Will I be able to immediately remember the contents from the name after two weeks without working on this project?

We get an error message:

```
3values <- c(1, 2, 4)
```

```
## Error: <text>:1:2: unerwartetes Symbol
## 1: 3values
##      ^
```

```
three_values <- c(1, 2, 4)
```

8.2 Save and load

```
save(frost, file = "frost_data.RData")
load(file = "frost_data.RData")
```

8.3 Litter service

- Names of objects in current session: `ls()`
- Litter service using `rm()`
- `dput()` comes as a helper

```
ls()
```

```
##  [1] "a"            "b"            "br"           "df"           "drought"
##  [6] "frost"        "i"            "m"            "Mu"           "Mu_q"
## [11] "nd"           "paint"        "three_values" "tmp"
```

```r
dput(ls())
## c("a", "b", "br", "df", "drought", "frost", "i", "m", "Mu", "Mu_q",
## "nd", "paint", "three_values", "tmp")
dput(ls()[!(ls() %in% c("a", "b", "drought", "frost"))])
## c("br", "df", "i", "m", "Mu", "Mu_q", "nd", "paint", "three_values",
## "tmp")
rm("bair", "br", "d", "d_breaks_cut", "dd", "elev", "i", "m", "Mu",
   "Mu_q", "nd", "paint", "quantile_sequence", "species", "three_values", "tmp")
## Warning in rm("bair", "br", "d", "d_breaks_cut", "dd", "elev", "i", "m", :
## Objekt 'bair' nicht gefunden
## Warning in rm("bair", "br", "d", "d_breaks_cut", "dd", "elev", "i", "m", :
## Objekt 'd' nicht gefunden
## Warning in rm("bair", "br", "d", "d_breaks_cut", "dd", "elev", "i", "m", :
## Objekt 'd_breaks_cut' nicht gefunden
## Warning in rm("bair", "br", "d", "d_breaks_cut", "dd", "elev", "i", "m", :
## Objekt 'dd' nicht gefunden
## Warning in rm("bair", "br", "d", "d_breaks_cut", "dd", "elev", "i", "m", :
## Objekt 'elev' nicht gefunden
## Warning in rm("bair", "br", "d", "d_breaks_cut", "dd", "elev", "i", "m", :
## Objekt 'quantile_sequence' nicht gefunden
## Warning in rm("bair", "br", "d", "d_breaks_cut", "dd", "elev", "i", "m", :
## Objekt 'species' nicht gefunden
ls()
## [1] "a"       "b"       "df"      "drought" "frost"
```

9 Dataobject classes

9.1 Vector

A vector is a one-dimensional combination of length 1 vectors of the same basic vector type.

9.1.1 Basics

Objective	Call
Combination of elements	`c(A, B)`
Indexing	`c(A, B)[1]`
Length of vector	`length(x)`
Integer sequence	`A:B`
Flexible sequence	`seq(A, B, length = N)`, `seq(A, B, by = K)`
Repeat	`rep(A, times = N)`, `rep(c(A, B), each = K)`
Sort elements	`sort(c(A, B), decreasing = FALSE)`
Order of elements	`order(c(A, B))`

9.1.2 Exercises

Copy-paste and run each line: describe what it does and what it results in.

```
a <- drought$elev[1]; b <- drought$elev[2]
(x <- c(a, b))
```

```
## [1] 335 460
```

```
c(x, a)
```

```
## [1] 335 460 335
```

```
c(x, x)
```

```
## [1] 335 460 335 460
```

```
x[2] <- drought$elev[3]; x
```

```
## [1] 335 480
```

```
length(x)
```

```
## [1] 2
```

```
a:b
```

```
##   [1] 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352
##  [19] 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370
##  [37] 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388
##  [55] 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406
##  [73] 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424
##  [91] 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442
## [109] 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460
```

```
seq(a, b, by = .5)
```

```
##   [1] 335.0 335.5 336.0 336.5 337.0 337.5 338.0 338.5 339.0 339.5 340.0 340.5
##  [13] 341.0 341.5 342.0 342.5 343.0 343.5 344.0 344.5 345.0 345.5 346.0 346.5
##  [25] 347.0 347.5 348.0 348.5 349.0 349.5 350.0 350.5 351.0 351.5 352.0 352.5
##  [37] 353.0 353.5 354.0 354.5 355.0 355.5 356.0 356.5 357.0 357.5 358.0 358.5
##  [49] 359.0 359.5 360.0 360.5 361.0 361.5 362.0 362.5 363.0 363.5 364.0 364.5
##  [61] 365.0 365.5 366.0 366.5 367.0 367.5 368.0 368.5 369.0 369.5 370.0 370.5
##  [73] 371.0 371.5 372.0 372.5 373.0 373.5 374.0 374.5 375.0 375.5 376.0 376.5
##  [85] 377.0 377.5 378.0 378.5 379.0 379.5 380.0 380.5 381.0 381.5 382.0 382.5
```

```
##  [97] 383.0 383.5 384.0 384.5 385.0 385.5 386.0 386.5 387.0 387.5 388.0 388.5
## [109] 389.0 389.5 390.0 390.5 391.0 391.5 392.0 392.5 393.0 393.5 394.0 394.5
## [121] 395.0 395.5 396.0 396.5 397.0 397.5 398.0 398.5 399.0 399.5 400.0 400.5
## [133] 401.0 401.5 402.0 402.5 403.0 403.5 404.0 404.5 405.0 405.5 406.0 406.5
## [145] 407.0 407.5 408.0 408.5 409.0 409.5 410.0 410.5 411.0 411.5 412.0 412.5
## [157] 413.0 413.5 414.0 414.5 415.0 415.5 416.0 416.5 417.0 417.5 418.0 418.5
## [169] 419.0 419.5 420.0 420.5 421.0 421.5 422.0 422.5 423.0 423.5 424.0 424.5
## [181] 425.0 425.5 426.0 426.5 427.0 427.5 428.0 428.5 429.0 429.5 430.0 430.5
## [193] 431.0 431.5 432.0 432.5 433.0 433.5 434.0 434.5 435.0 435.5 436.0 436.5
## [205] 437.0 437.5 438.0 438.5 439.0 439.5 440.0 440.5 441.0 441.5 442.0 442.5
## [217] 443.0 443.5 444.0 444.5 445.0 445.5 446.0 446.5 447.0 447.5 448.0 448.5
## [229] 449.0 449.5 450.0 450.5 451.0 451.5 452.0 452.5 453.0 453.5 454.0 454.5
## [241] 455.0 455.5 456.0 456.5 457.0 457.5 458.0 458.5 459.0 459.5 460.0
```

```
seq(a, b, by = 5)
```

```
##  [1] 335 340 345 350 355 360 365 370 375 380 385 390 395 400 405 410 415 420 425
## [20] 430 435 440 445 450 455 460
```

```
seq(a, b, length = 10) ## Sequenz
```

```
##  [1] 335.0000 348.8889 362.7778 376.6667 390.5556 404.4444 418.3333 432.2222
##  [9] 446.1111 460.0000
```

```
rep(x, times = 3)
```

```
## [1] 335 480 335 480 335 480
```

```
rep(x, each = 3)
```

```
## [1] 335 335 335 480 480 480
```

```
rep(x, length = 7)
```

```
## [1] 335 480 335 480 335 480 335
```

```
sort(x, decreasing = TRUE)
```

```
## [1] 480 335
```

```
order(c(x, 100))
```

```
## [1] 3 1 2
```

```
a <- as.character(drought$species[1]); b <- as.character(drought$species[2])
(x <- c(a, b))
```

```
## [1] "Spruce" "Spruce"
```

```
c(x, "Beech")
```

```
## [1] "Spruce" "Spruce" "Beech"
```

```
c(x, 2)
```

```
## [1] "Spruce" "Spruce" "2"
```

9.2 Calculations

- Basic calculations operate independently on all elements
- Function calls work (usually) with vectorized arguments
- Summary of the content: `summary()`
- Frequency table: `table()` (see Session 04 for much more detail)

9.2.1 Exercises

Copy-paste and run each line: describe what it does and what it results in.

```r
x <- drought$bair
x^2
sin(x)
length(x)
summary(x)
boxplot(x)
stripchart(x, add = T, pch = 16, method = "jitter", vertical = T)
```

```r
species <- drought$species
xtabs(~ species)
xtabs(~ species + (drought$elev > 1000))
```

9.3 Matrix

A vector is a two-dimensional combination of single-element objects of the same class.

9.3.1 Basics

P×P matrix with content content:

```r
A <- matrix(nrow = P, ncol = Q, data = content)
```

Fill matrix column- (default) or linewise using argument `byrow = TRUE`:

- Indexing of one element: `A[1, 1]`
- Indexing of first row (result is vector): `A[1, ]`
- Indexing of several rows (result is matrix): `A[1:3, ]`
- Indexing of first column (result is vector): `A[, 1]`
- Indexing of several columns (result is matrix): `A[, 1:3]`
- Indexing of several rows and columns (result is matrix): `A[1:3, 1:3]`

9.3.2 Exercises

```r
a <- drought$elev
b <- drought$bair
length(a) == length(b)
```

```
## [1] TRUE
```

```r
A <- matrix(nrow = length(a), ncol = 2, data = c(a, b))
all(A[, 1] == a)
```

```
## [1] TRUE
```

```r
A[1:3, ]
```

```
##      [,1]  [,2]
## [1,]  335 0.505
## [2,]  460 0.648
## [3,]  480 0.523
```

```r
A[1:3, 2]
```

```
## [1] 0.505 0.648 0.523
```

```r
A[c(1:3, 29), ]
```

```
##      [,1]  [,2]
## [1,]  335 0.505
## [2,]  460 0.648
## [3,]  480 0.523
## [4,]  780 0.944
```

```r
B <- matrix(nrow = 3, ncol = 3, data = 1:9); B
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
diag(B)
```

```
## [1] 1 5 9
```

```
B %*% B
```

```
##      [,1] [,2] [,3]
## [1,]   30   66  102
## [2,]   36   81  126
## [3,]   42   96  150
```

```
B %*% c(1, 1, 1)
```

```
##      [,1]
## [1,]   12
## [2,]   15
## [3,]   18
```

```
rowSums(B)
```

```
## [1] 12 15 18
```

```
c(1, 1, 1) %*% B
```

```
##      [,1] [,2] [,3]
## [1,]    6   15   24
```

```
colSums(B)
```

```
## [1]  6 15 24
```

9.4 List

A list is a general 'container' object.

9.4.1 Basics

- Lists may contain elements storing objects of varying classes, lengths, ... (`character`, `numeric`, `integer`, `factor`, ...)
- Construct a list using `list(...)`
- Indexing: `x[[1]]` or `x[['name']]` or `x$name`
- `str` gives the structure
- For lists of 'consistent' classes (or generic functions):

```
lapply(X = mylist, FUN = myfun)
```

9.4.2 Exercises

```
beech <- list(species = "Fagus",
              n = sum(drought$species == "Beech"),
              data = subset(drought, species == "Beech"))
beech
```

```
## $species
## [1] "Fagus"
##
## $n
## [1] 11
##
```

```
## $data
##     bair   elev species         elev_cut
## 24 0.697  475.0   Beech        (250,500]
## 25 0.606  480.0   Beech        (250,500]
## 26 0.718  507.5   Beech        (500,750]
## 27 0.480  580.0   Beech        (500,750]
## 28 0.822  750.0   Beech        (500,750]
## 29 0.944  780.0   Beech      (750,1e+03]
## 30 0.770  800.0   Beech      (750,1e+03]
## 31 1.036 1025.0   Beech (1e+03,1.25e+03]
## 32 1.230 1100.0   Beech (1e+03,1.25e+03]
## 33 0.680 1150.0   Beech (1e+03,1.25e+03]
## 34 0.985 1200.0   Beech (1e+03,1.25e+03]
```

```
beech[[1]]
```

```
## [1] "Fagus"
```

```
beech[['species']]
```

```
## [1] "Fagus"
```

```
beech$species
```

```
## [1] "Fagus"
```

```
str(beech)
```

```
## List of 3
##  $ species: chr "Fagus"
##  $ n      : int 11
##  $ data   :'data.frame': 11 obs. of  4 variables:
##   ..$ bair    : num [1:11] 0.697 0.606 0.718 0.48 0.822 ...
##   ..$ elev    : num [1:11] 475 480 508 580 750 ...
##   ..$ species : Factor w/ 2 levels "Beech","Spruce": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ elev_cut: Factor w/ 6 levels "(250,500]","(500,750]",..: 1 1 2 2 2 3 3 4 4 4 ...
```

```
lapply(beech, FUN = dim)
```

```
## $species
## NULL
##
## $n
## NULL
##
## $data
## [1] 11  4
```

```
lapply(beech, FUN = summary)
```

```
## $species
##    Length     Class      Mode
##         1 character character
##
## $n
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      11      11      11      11      11      11
##
## $data
##       bair             elev          species                 elev_cut
##  Min.   :0.4800   Min.   : 475.0   Beech :11   (250,500]         :2
##  1st Qu.:0.6885   1st Qu.: 543.8   Spruce: 0   (500,750]         :3
##  Median :0.7700   Median : 780.0               (750,1e+03]       :2
##  Mean   :0.8153   Mean   : 804.3               (1e+03,1.25e+03]  :4
##  3rd Qu.:0.9645   3rd Qu.:1062.5               (1.25e+03,1.5e+03]:0
```

```
## Max.   :1.2300   Max.   :1200.0              (1.5e+03,1.75e+03]:0
```

9.5 Synthesis and indexing

- We may provide names for elements of vectors, matrices and lists:
- `dimnames(A)`, `rownames(A)`, `colnames(A)` for matrices
- Select a named element by using the $ sign
- Select elements by `[[]]` (for lists), `[]` (for vectors), and `[, ]` (for matrices).

9.5.1 Exercises

```
c(one = 1, two = 2, four = 4)
x <- c(1, 2, 4)
x
names(x) <- c("one", "two", "four")
x
names(beech)
A <- matrix(nrow = 10, ncol = 2, data = c(drought$elev[1:10], drought$bair[1:10]))
colnames(A)
colnames(A) <- c("elev", "bair")
colnames(A)
rownames(A)
names(A)
dimnames(A)
```

9.6 Dataframe

Based on their object properties, dataframes can be classified between matrices and lists, whereby the columns (the so-called 'variables') of the dataframe correspond to the elements of a list:

- Dataframes are more rigid than lists because all columns must have the same length,
- Dataframes are more flexible than matrices, since all columns can contain different contents (numbers, strings, ...).
- Dataframes are generated with `data.frame ()`
- Indexing:
- Rows and columns can be indexed in the same way as the elements of a matrix,
- Columns can be indexed in the same way as the elements of a list (we have already used this by `drought$elev`).

9.6.1 Exercises

```
weather <- data.frame(day = c("Monday", "Tuesday", "Wednesday"),
                      daily_mean_temperature_C = c(12, 14, 11),
                      precipitation_sum_mm = c(5, 9, 25),
                      site = "Goettingen")
weather
```

```
##         day daily_mean_temperature_C precipitation_sum_mm       site
## 1    Monday                       12                    5 Goettingen
## 2   Tuesday                       14                    9 Goettingen
## 3 Wednesday                       11                   25 Goettingen
```

```
weather[1:2, ]
```

```
##       day daily_mean_temperature_C precipitation_sum_mm       site
## 1  Monday                       12                    5 Goettingen
## 2 Tuesday                       14                    9 Goettingen
```

```
weather$day[1:2]
```

```
## [1] Monday  Tuesday
## Levels: Monday Tuesday Wednesday
```

```
weather[["daily_mean_temperature_C"]][3]
```

```
## [1] 11
```

```
weather[1:2, c("daily_mean_temperature_C", "precipitation_sum_mm")]
```

```
##   daily_mean_temperature_C precipitation_sum_mm
## 1                       12                    5
## 2                       14                    9
```

```
weather[1:2, c("day", "precipitation_sum_mm")]
```

```
##       day precipitation_sum_mm
## 1  Monday                    5
## 2 Tuesday                    9
```

9.7 Functions on dataframes

- `summary`: Summary of each of the variables in the dataframe
- `str`: overview of the structure of the dataframe
- `dim`: Dimension of the dataframe (number of rows and columns)
- The first `N` lines of a dataframe `df` are extracted with `head(df, n = N)`, and
- the last `N` lines of a dataframe `df` are extracted with `tail (d, n = N)`.

9.7.1 Exercises

```
summary(weather)
```

```
##         day    daily_mean_temperature_C precipitation_sum_mm         site
##  Monday   :1   Min.   :11.00            Min.   : 5           Goettingen:3
##  Tuesday  :1   1st Qu.:11.50            1st Qu.: 7
##  Wednesday:1   Median :12.00            Median : 9
##                Mean   :12.33            Mean   :13
##                3rd Qu.:13.00            3rd Qu.:17
##                Max.   :14.00            Max.   :25
```

```
str(weather)
```

```
## 'data.frame':    3 obs. of  4 variables:
##  $ day                     : Factor w/ 3 levels "Monday","Tuesday",..: 1 2 3
##  $ daily_mean_temperature_C: num  12 14 11
##  $ precipitation_sum_mm    : num  5 9 25
##  $ site                    : Factor w/ 1 level "Goettingen": 1 1 1
```

```
head(weather, n = 2)
```

```
##       day daily_mean_temperature_C precipitation_sum_mm       site
## 1  Monday                       12                    5 Goettingen
## 2 Tuesday                       14                    9 Goettingen
```

```
dim(weather)
```

```
## [1] 3 4
```

10 Functions for character strings

If we want to change character strings (such as variable names of a dataframe), the following functions can be helpful:

nchar

Returns the number of characters of a string:

```
nchar(names(weather))
```

```
## [1]  3 24 20  4
```

tolower und toupper

Replaces uppercase with lowercase letters (and vice versa).

```
toupper(c("Beech", "BEech", "beech"))
```

```
## [1] "BEECH" "BEECH" "BEECH"
```

```
tolower(c("Beech", "BEech", "beech"))
```

```
## [1] "beech" "beech" "beech"
```

gsub

Replace a pattern:

```
names(weather)
```

```
## [1] "day"                      "daily_mean_temperature_C"
## [3] "precipitation_sum_mm"     "site"
```

```
names(weather) <- gsub(names(weather), pattern = "_mm", replacement = "__mm", fixed = T)
names(weather) <- gsub(names(weather), pattern = "_C", replacement = "__C", fixed = T)
names(weather)
```

```
## [1] "day"                       "daily_mean_temperature__C"
## [3] "precipitation_sum__mm"     "site"
```

substring

Substrings from `first` to `last`.

```
substring(names(weather), first = 1, last = 5)
```

```
## [1] "day"   "daily" "preci" "site"
```

strsplit

Splits strings according to a certain pattern (`strsplit` always returns a list).

```
names(weather) <- gsub(names(weather), pattern = "__", replacement = "_", fixed = T)
(tmp <- strsplit(names(weather), split = "_", fixed = T))
```

```
## [[1]]
## [1] "day"
##
## [[2]]
## [1] "daily"       "mean"        "temperature" "C"
```

```
##
## [[3]]
## [1] "precipitation" "sum"           "mm"
##
## [[4]]
## [1] "site"
```

paste

Merges strings (arguments sep orcollapse).

```
sapply(tmp, FUN = function(x){paste(x, collapse = "_")})
```

```
## [1] "day"                        "daily_mean_temperature_C"
## [3] "precipitation_sum_mm"       "site"
```

11 Factors

For the analysis of qualitative characteristics, it makes sense to represent variables with character strings as **factors**.

A factor comes with three ingredients:

- data vector `x`
- `levels`
- `labels`

We can generate a factor without specification of levels and labels: In this case, factor levels (`levels`) are generated (in alphabetical order), numbered internally (see this numeric expression using `as.numeric`), and represented externally by the original values of the character string.

```
(tmp <- factor(x = c("Beech", "Spruce", "Beech")))
```

```
## [1] Beech  Spruce Beech
## Levels: Beech Spruce
```

```
attributes(tmp)
```

```
## $levels
## [1] "Beech"  "Spruce"
##
## $class
## [1] "factor"
```

```
levels(tmp)
```

```
## [1] "Beech"  "Spruce"
```

```
as.numeric(tmp)
```

```
## [1] 1 2 1
```

```
as.character(tmp)
```

```
## [1] "Beech"  "Spruce" "Beech"
```

We can provide levels, to which the numerical representation will refer to:

```
(tmp <- factor(x = c("Beech", "Spruce", "Beech"),
               levels = c("Spruce", "Beech")))
```

```
## [1] Beech  Spruce Beech
## Levels: Spruce Beech
```

```
attributes(tmp)
```

```
## $levels
## [1] "Spruce" "Beech"
##
## $class
## [1] "factor"
```

```
levels(tmp)
```

```
## [1] "Spruce" "Beech"
```

```
as.numeric(tmp)
```

```
## [1] 2 1 2
```

```
as.character(tmp)
```

```
## [1] "Beech"  "Spruce" "Beech"
```

Using labels, the values represented externally are the result of the mapping from `levels` to `labels`

```
(tmp <- factor(x = c("Beech", "Spruce", "Beech"),
               levels = c("Beech", "Spruce"),
               labels = c("Fagus sylvatica", "Picea abies")))
```

```
## [1] Fagus sylvatica Picea abies     Fagus sylvatica
## Levels: Fagus sylvatica Picea abies
```

```
attributes(tmp)
```

```
## $levels
## [1] "Fagus sylvatica" "Picea abies"
##
## $class
## [1] "factor"
```

```
levels(tmp)
```

```
## [1] "Fagus sylvatica" "Picea abies"
```

```
as.numeric(tmp)
```

```
## [1] 1 2 1
```

```
as.character(tmp)
```

```
## [1] "Fagus sylvatica" "Picea abies"     "Fagus sylvatica"
```

11.1 Exercises

```
tmp <- factor(x = c("Beech", "Spruce", "Beech", "Oak"),
               levels = c("Beech", "Spruce"),
               labels = c("Fagus sylvatica", "Picea abies"))
attributes(tmp)
levels(tmp)
as.numeric(tmp)
as.character(tmp)
tmp <- factor(x = c("Beech", "Spruce", "Beech", "Oak"),
               levels = c("Beech", "Oak", "Spruce"),
               labels = c("Deciduous", "Deciduous", "Conifer"))
attributes(tmp)
levels(tmp)
as.numeric(tmp)
as.character(tmp)
```