“Code is the lifeblood of technology.”

“Code is art, and you are the artist.”

CODING HYGIENE

This powerpoint is based on:

https://medium.com/@anishmahapatra/code-hygiene-dont-laugh-it-off-2a5aebcdd84b

https://stat.ethz.ch/Teaching/maechler/R/useR_2014/Maechler-2014-pr.pdf

https://style.tidyverse.org/syntax.html ← *Very detailed with Good and Bad examples*

https://waterdata.usgs.gov/blog/intro-best-practices/

https://www.r-bloggers.com/r-best-practices-r-you-writing-the-r-way/

https://swcarpentry.github.io/r-novice-inflammation/06-best-practices-R/

And based on the use of Rstudio!

Search: « Best coding practices » or « Best coding practices in R »

The main idea is that your code should be reproductible!

*Someone should be able to re-run your code and **understand** **what** and **why** you did what you did.*

SPOILERS: This *someone* could be **YOU**!

… getting lost with your own code… #fail

MAIN RULE # 1

COMMENT

Section 1 #### (or … # Section 1 ####) *Will make the section foldable!*

Major « sectioning » comments or for section subtitles

For usual comments

 # For end-of-line comments

MAIN RULE # 2

- To indent blocks of lines, select lines and press “Tab”
- To remove indent, select lines and press “Shift + Tab”

Indent

Bad

```
249
250 ### On refait les mêmes étapes suivantes afin d'enlever les erreurs potentielles
251 #-Supprimer les plantations (champ "treatment" = Plantation)
252 t0_2 = subset(t0_1, t0_1$treatment != "Plantation")
253 t0_2$treatment = factor(t0_2$treatment)
254 table(t0_2$treatment)
255 #-Supprimer les enregistrements où "species" = "picgla", "picmar" et "sampub" car le brout n'a pas été compté sur ces espèces.
256 t0_3 = subset(t0_2, t0_2$species_cor != "Picea" & t0_2$species_cor != "Sampub")
257 table(t0_3$species_cor)
258 t0_3$species_cor = factor(t0_3$species_cor)
259 #-Supprimer toutes les lignes où: "no_browsing" != 0 OU "young_browsing" != 0 OU  "old_browsing" != 0 (car division de 0 par 0 = erreur)
260 # NOTE: Cette opération enlève également les données de 2015 pour les sites mis en place en 2010 (ex: CA1001), car seulement
261 # la hauteur et l'espèce avaient été notées en 2015...
262 t0_4 = t0_3[ which(t0_3$young_browsing != '0' | t0_3$no_browsing != '0' | t0_3$old_browsing != '0'),]
263 # Vérifier s'il y a des "NA" dans les champs "no_browsing" et "young_browsing", puisque nous ferons des calculs sur ces champs.
264 # Ne pas faire "brout = na.omit(brout)" Il y a entre-autre des NA dans le champ "diam", mais ce n'est pas grave pour nos calculs.
265 # À la place:
266 sum(is.na(t0_4$no_browsing)) # 0 NA
267 sum(is.na(t0_4$young_browsing)) # 0 NA
268
269 # Calculer le champs "tot_dispo" qui est la somme de "no_browsing" et de "young_browsing"
270 str(t0_4)
271 t0_4$young_browsing = as.numeric(t0_4$young_browsing)
272 t0_4$no_browsing = as.numeric(t0_4$no_browsing)
273 t0_4$old_browsing = as.numeric(t0_4$old_browsing)
274
275 t0_4$tot_dispo = t0_4$no_browsing + t0_4$young_browsing
276
277 # *******
278 # pour voir quel est le Working Directory où le fichier sera enregistré: "getwd()"
279 getwd()
280 ### exporter le fichier en .csv
281 write.csv(t0_4,"brout_t0_corrige_10avr2017.csv", row.names=F)
282
283 t0_4 = read.csv(file.choose()) # open brout_t0_corrige_10avr2017.csv
284
285 ### Calculer le nombre total de ramilles disponibles pour chaque site à T0
286 t0_5_totram = aggregate(t0_4$tot_dispo, by=list(t0_4$territory, t0_4$site, t0_4$EX, t0_4$time, t0_4$exclos, t0_4$sample_year, t0_4$build_year), FUN = sum)
287 head(t0_5_totram)
288 colnames(t0_5_totram) = c("territory","site","EX","time", "exclos", "sample_year", "build_year", "tot_dispo")
289 # Il est assez bizard d'avoir 3406 ramilles pour CH1105TE à T0... vérification :
290 test = subset(t0_4, t0_4$exclos == "CH1105TE")
291 sum(test$tot_dispo) # 3406
292
293 ### Calculer le nombre total de young browsing pour chaque site à T0
294 t0_5_totyou = aggregate(t0_4$young_browsing, by=list(t0_4$territory, t0_4$site, t0_4$EX, t0_4$time, t0_4$exclos, t0_4$sample_year, t0_4$build_year), FUN = sum)
295 head(t0_5_totyou)
```

Good

```
161             # On veut ensuite merger avec les donnees d habitat
162
163                         # verifier les annees pour les donnees d habitat
164                         table(hab_fct_mas$annee)
165                         hab_fct_mas2 = subset(hab_fct_mas, annee != "2016" & annee != "2017" & annee != "2018" & annee != "2019")
166                         hab_fct_mas2$annee = factor(hab_fct_mas2$annee)
167
168                         # pour l analyse, on veut que chaque fonction de l habitat soit une variable explicative, donc une colonne.
169                         # mettre les superficies de chaque couvert en wide
170                               library(reshape2)
171
172                               # superficies en hectares
173                               hab_fct_mas_ha = hab_fct_mas2[,c(7,6,3,1,2,4)]
174                               hab_fct_mas_ha2 = reshape(hab_fct_mas_ha,
175                                                         timevar = "fonction",
176                                                         idvar = c("reserve", "sup_zone_ha", "annee", "secteur"),
177                                                         direction = "wide")
178
179                                     hab_fct_mas_ha2[is.na(hab_fct_mas_ha2)] = 0
180                                     colnames(hab_fct_mas_ha2) = gsub("sup_couvert_", "", colnames(hab_fct_mas_ha2))
181
182
183                               # superficies en pourcentage
184                               hab_fct_mas_pourc = hab_fct_mas2[,c(7,6,3,1,2,5)]
185                               hab_fct_mas_pourc2 = reshape(hab_fct_mas_pourc,
186                                                         timevar = "fonction",
187                                                         idvar = c("reserve", "sup_zone_ha", "annee", "secteur"),
188                                                         direction = "wide")
189
190                                     hab_fct_mas_pourc2[is.na(hab_fct_mas_pourc2)] = 0
191                                     colnames(hab_fct_mas_pourc2) = gsub("sup_couvert_", "", colnames(hab_fct_mas_pourc2))
192
193                               # remettre les deux dataset (ha et pourc) ensemble
194
195                               hab_fct_mas3 = merge(hab_fct_mas_ha2, hab_fct_mas_pourc2, by = c("reserve", "sup_zone_ha", "annee", "secteur"))
196                                       # les 3 datasets ont 924 obs. ce qui signifie que le merge c'est bien fait (on na pas creer ou supprimer d observations)
197
198                   # 2e Merge
199                   dat_mas = merge(suc_clim_mas, hab_fct_mas3, by = c("reserve", "annee", "secteur"), all.x = TRUE)
200                         # on devrait avoir 2171 obs, mais on obtient 2149...
201                         # on remarque qu'il n'y a pas de donnees d habitat pour le secteur 8 (22 observations)
202                         # les donnees sont absentes du fichier dbf de base ("area" dans ECO_reward_LDV.R)
203                               # Donc, on flush le secteur 8 dans Mastigouche pour les analyses
204
205                         dat_mas = merge(suc_clim_mas, hab_fct_mas3, by = c("reserve", "annee", "secteur"))
206                         dat_mas = dat_mas[, !names(dat_mas) %in% c("nb_group")]
```

MAIN RULE # 3

Nomenclature

- Do not name variables with letters or weird names: a, b, c, final, final1, …
- Do not name variables with names frequently seen in functions

- Use names that make sense !

  Even if they get longer, at least you understand what you ar working with… and with the assist name thing, it is not so bad!

MAIN RULE # 4

Use spaces!!!

https://style.tidyverse.org/syntax.html

MAIN RULE # 4

```
# ------------ INCORRECT SPACING ----------------------------
# The value of b is three times a
b=((a*3)/2)^4#This looks weird already, is it that tough??
c=[1,2,3]#Why are examples so difficult to understand


------------ CORRECT SPACING ----------------------------


# The value of b is three times a


b = ( (a*3) / 2) ^ 4  # BODMAS?
c = [1, 2, 3]         # Maybe they do not need to be
```

https://medium.com/@anishmahapatra/code-hygiene-dont-laugh-it-off-2a5aebcdd84b

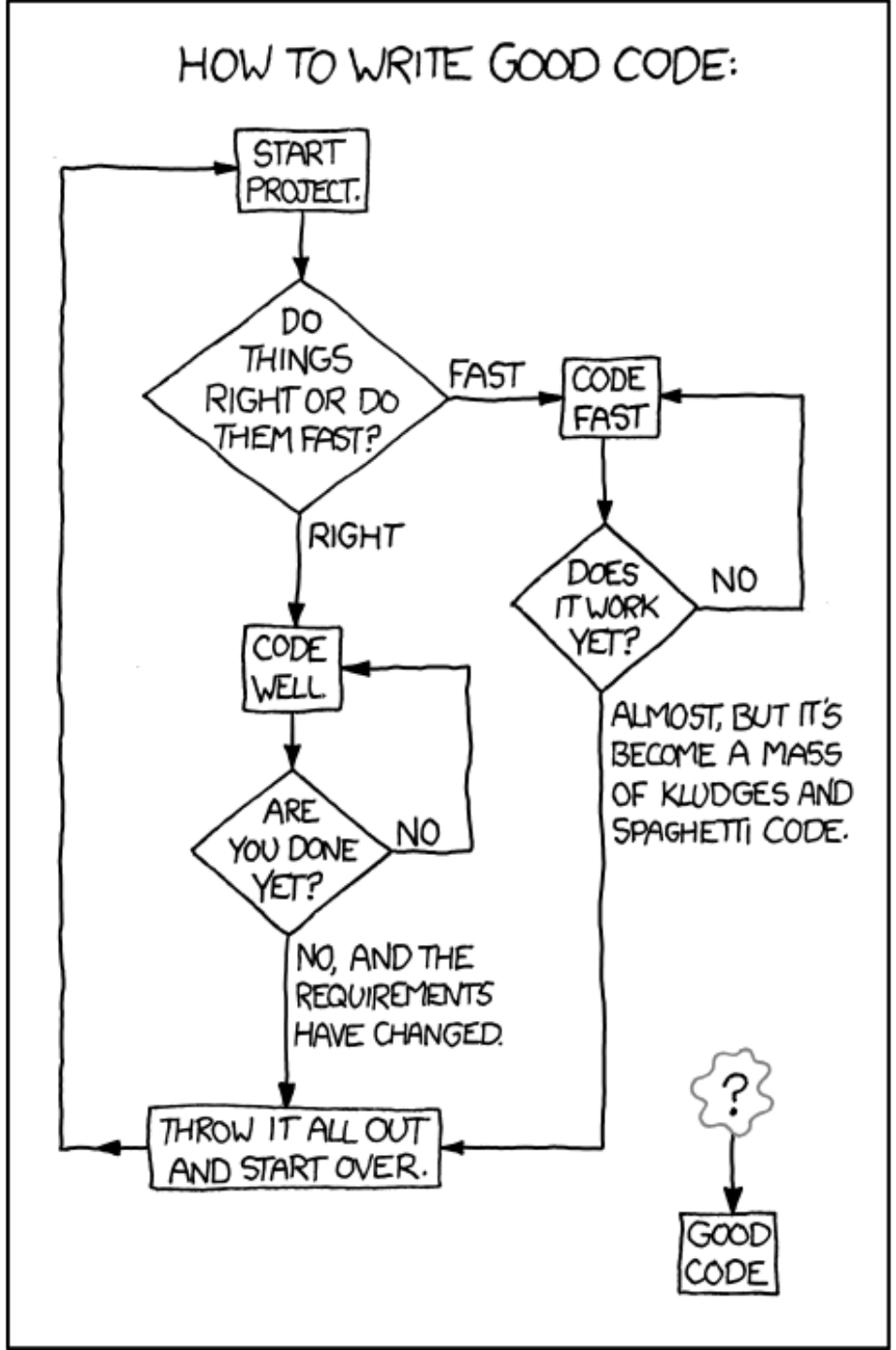
MAIN RULE # 5

Take the time

Be nice to your code

Choose a standard and stick to it!

MAIN RULE # 5



https://medium.com/@anishmahapatra/code-hygiene-dont-laugh-it-off-2a5aebcdd84b

Others advices/tricks found

- Start the code by describing who wrote the code and what it intent to do. Add date.
- Put all library(packages) at the beginning

  Frustrating to run a code and realize at the end that a package was missing…
- Set/create directory/source at the beginning:

Good

```
input_file <- "data/data.csv"
output_file <- "data/results.csv"

# read input
input_data <- read.csv(input_file)
# get number of samples in data
sample_number <- nrow(input_data)
# generate results
results <- some_other_function(input_file, sample_number)
# write results
write.table(results, output_file)
```

Bad

```
# check
input_data <- read.csv("data/data.csv")
# get number of samples in data
sample_number <- nrow(input_data)
# generate results
results <- some_other_function("data/data.csv", sample_number)
# write results
write.table("data/results.csv", output_file)
```

BE CAREFUL WITH *SETWD()*

Nobody has the same repository path….

- Set/create directory/source at the beginning:

```
55  # ********************************************
56  ### Donnée de brout 2015
57  data = read.csv(file.choose()) # open brout_2015_net.csv
58  info = read.csv(file.choose()) # open infopourmerge.csv
```

Disadvantage: cannot run the whole code without choosing each file one-by-one

```
3   directory1 <- "/Users/laurentdevriendt/OneDrive - Universite Laval/Université/"
4   directory2 <- "/Users/laurentdevriendt/OneDrive - Universite Laval/Université/Doctorat"
5
6
7   setwd(directory1)
8   info = read.csv("infopourmerge.csv", header = TRUE)
9
10  setwd(directory2)
11  brout_brut = read.csv("brout_R_nov2016.csv", header = TRUE)
12
```

Better? Someone else can change the path at the beginning of the code and then run the whole code… What do you think?

Others advices/tricks found

- Include arguments names when using functions
  Except for very frequent arguments…

- Use "<-", not "="
  Alt + space = "<-" (the assignment operator is configurable)

- If you use temporary code, delete it after or indent and state clearly that this is *temporary-code*
  Make sure that the temporary code can't interfere with the main code (use different names, like "test" or "temp")

- Do wrap long lines!
  You should never scroll left-right
  Try to limit your code to 80 characters per line
  Soft-wrap R tool…

Others advices/tricks found

- Don’t save session history/environment!
  “Start in a clean environment so that older objects don’t remain in your environment any longer than they need to. If that happens, it can lead to unexpected results.”

- Avoid Copy & Paste of codes lines
  Less possible errors
  By rewriting your code, you will get better at coding
  Be flexible though…

- Use png(), tiff(), jpg(), etc. Not plot-viewer – export
  Better control on parameters, better reproducibility!

- Working on somebody else’s code? Put your initials in comments to identify added/modified code

- See main rule #6…

MAIN RULE # 6

Write Functions

and loops!

Who's next? I am bad at writing functions…

Extra L. De Vriendt

- Keep a script of « useful codes »
- Include the date in final files/figures (“Figure_prc_browsing_classes_2020_01_05”)
- ALWAYS write date as YEAR-MONTH-DAY (for everything, everywhere!)
- Adding new code to old code? Write **“UPDATE + date”** and **indent**
- Add the url of the reference code in comments
- Indent by multiple of 3
- Always double-check data transformation/selection, especially in complex data manipulations

  Verify that the values haven’t changed or changed in the desired way, count the number of lines/columns you should have and compare with what you have. Data get lost so easily…

Extra L. De Vriendt

- Avoid capitalization!
  Can only be a source of problems. In code, in assignations, in file/figure names, etc. just DON'T
- Avoid spaces in names (use "_ ")
- Write results of analyses in end-of-line comments
  When rerolling the stats, you will see if an error slipped in
- Find & Replace
  Save time! But be careful and resist the temptation to "Replace all"
- Keep a "Trash" section at the end of your code

Everybody knows tricks! What is yours?

- When using {} for loops and function, put { alone in first line and } on the last line to allow folding.
- When working on an already existing code, start by saving into a new file to avoid overwriting the original code file.