



29 NOVEMBER 2018

Herman Teirlinck,
01.71 Frans Breziers

What has Damiano done?!?

28 days ago...



damianooldoni commented 28 days ago • edited ▾



I developed this workaround by using a `while` in an ad-hoc function:

```
get_random_pt <- function(x) {  
  random_pt <- st_sample(x , size = 1, type = "random")  
  while (length(random_pt) == 0) {  
    random_pt <- st_sample(x , size = 1, type = "random")  
  }  
  return(random_pt)  
}
```

Then, if you want do it for a data.frame of polygons (`polygons`) as I needed,wrap this function in a `map()` and then assign the CRS of the original polygons df (`crs_polygons <- st_crs(polygons)`) back via `st_sfc()` :

```
library(purrr)  
random_pts <- map(st_geometry(polygons), get_random_pt)  
random_pts <- st_sfc(unlist(random_pts, recursive = FALSE),  
                    crs = crs_polygons  
)
```

.. [suggestion](#) by
Damiano

We are the community!

15 days ago...

... [proposal](#) to have this fixed in `sf` package!

The screenshot shows a GitHub pull request interface for the repository `r-spatial/sf`. At the top, it indicates the pull request is **Open** and that `Robinlovelace` wants to merge 7 commits into `r-spatial:master` from `Robinlovelace:sample-exact-argument`. The repository statistics show 70 watches, 439 stars, and 108 forks. The pull request details include 7 conversations, 7 commits, 0 checks, and 3 files changed, with a net change of +91 lines and -30 lines. The commit history shows a sequence of updates and a reference to another pull request (#813). The most recent comment from `Robinlovelace` states: "Heads-up @edzer this is a first pass. Not tested yet. Feedback welcome." The right sidebar contains sections for Reviewers (No reviews), Assignees (No one assigned), Labels (None yet), Projects (None yet), Milestone (No milestone), and Notifications (Subscribe button, with a note that the user is not receiving notifications from this thread). At the bottom, it shows 2 participants.



DATA
MANIPULATION

PART 3

Data Transformation with dplyr : : CHEAT SHEET



dplyr functions work with pipes and expect tidy data. In tidy data:



Each **variable** is in its own **column**



Each **observation, or case**, is in its own **row**



pipes

$x \%>\% f(y)$ becomes $f(x, y)$

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function



summarise(data, ...) Compute table of summaries. `summarise(mtcars, avg = mean(mpg))`



count(x, ..., wt = NULL, sort = FALSE) Count number of rows in each group defined by the variables in ... Also **tally**(). `count(iris, Species)`

VARIATIONS

summarise_all() - Apply funs to every column.

summarise_at() - Apply funs to specific columns.

summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by**() to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.



`mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))`

group_by(data, ..., add = FALSE) Returns copy of table grouped by ... `g_iris <- group_by(iris, Species)`

ungroup(x, ...) Returns ungrouped copy of table. `ungroup(g_iris)`

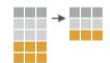
Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.



filter(data, ...) Extract rows that meet logical criteria. `filter(iris, Sepal.Length > 7)`



distinct(data, ..., keep_all = FALSE) Remove rows with duplicate values. `distinct(iris, Species)`



sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, env = parent.frame()) Randomly select fraction of rows. `sample_frac(iris, 0.5, replace = TRUE)`



sample_n(tbl, size, replace = FALSE, weight = NULL, env = parent.frame()) Randomly select size rows. `sample_n(iris, 10, replace = TRUE)`



slice(data, ...) Select rows by position. `slice(iris, 10:15)`

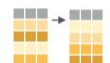
top_n(x, n, wt) Select and order top n entries (by group if grouped data). `top_n(iris, 5, Sepal.Width)`

Logical and boolean operators to use with filter()

`<` `<=` `is.na()` `%in%` `|` `xor()`
`>` `>=` `!is.na()` `!` `&`

See ?base::logic and ?Comparison for help.

ARRANGE CASES



arrange(data, ...) Order rows by values of a column or columns (low to high), use with **desc**() to order from high to low. `arrange(mtcars, mpg)` `arrange(mtcars, desc(mpg))`

ADD CASES



add_row(data, ..., before = NULL, after = NULL) Add one or more rows to a table. `add_row(faithful, eruptions = 1, waiting = 1)`

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.



pull(data, var = -1) Extract column values as a vector. Choose by name or index. `pull(iris, Sepal.Length)`



select(data, ...) Extract columns as a table. Also **select_if**(). `select(iris, Sepal.Length, Species)`

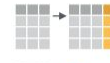
Use these helpers with **select**(), e.g. `select(iris, starts_with("Sepal"))`

contains(match) **num_range**(prefix, range) ; e.g. `mpg:cyl`
ends_with(match) **one_of**(...) ; e.g. `Species`
matches(match) **starts_with**(match)

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function



mutate(data, ...) Compute new column(s). `mutate(mtcars, gpm = 1/mpg)`



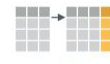
transmute(data, ...) Compute new column(s), drop others. `transmute(mtcars, gpm = 1/mpg)`



mutate_all(tbl, funs, ...) Apply funs to every column. Use with **funs**(). Also **mutate_if**(). `mutate_all(faithful, funs(log(), log2()))` `mutate_if(iris, is.numeric, funs(log()))`



mutate_at(tbl, cols, funs, ...) Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for **select**(). `mutate_at(iris, vars(-Species), funs(log()))`



add_column(data, ..., before = NULL, after = NULL) Add new column(s). Also **add_count()**, **add_tally**(). `add_column(mtcars, new = 1:32)`



rename(data, ...) Rename columns. `rename(iris, Length = Sepal.Length)`




Share your snippets and solutions during the coding session:

Go to <https://hackmd.io/Fjm4XuozRKSDyFNXuU6mqQ> and post your code in between backticks:

For example:

```
```\n\nlibrary(tidyverse)\n\nmy_data <- ... \n\n```
```

# The concept

We defined a number of challenges. If you were able to achieve a challenge, add a  to your laptop screen.

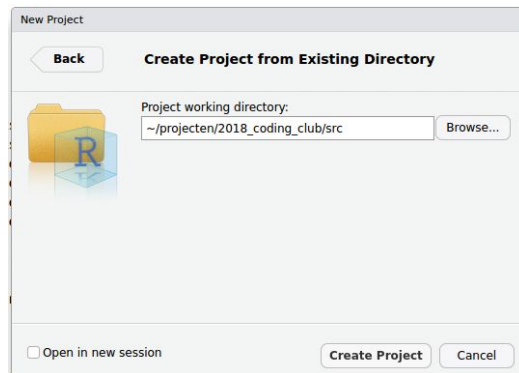
The objective is that **everyone** achieves !

- Someone has more  than you? **Ask for help!**
- Someone has less  than you? **Provide help!**

- Download coding club material and work locally, **not in sync** with the Google drive



- Create new Rstudio project in your local coding club folder (or in `src` folder, as you prefer)

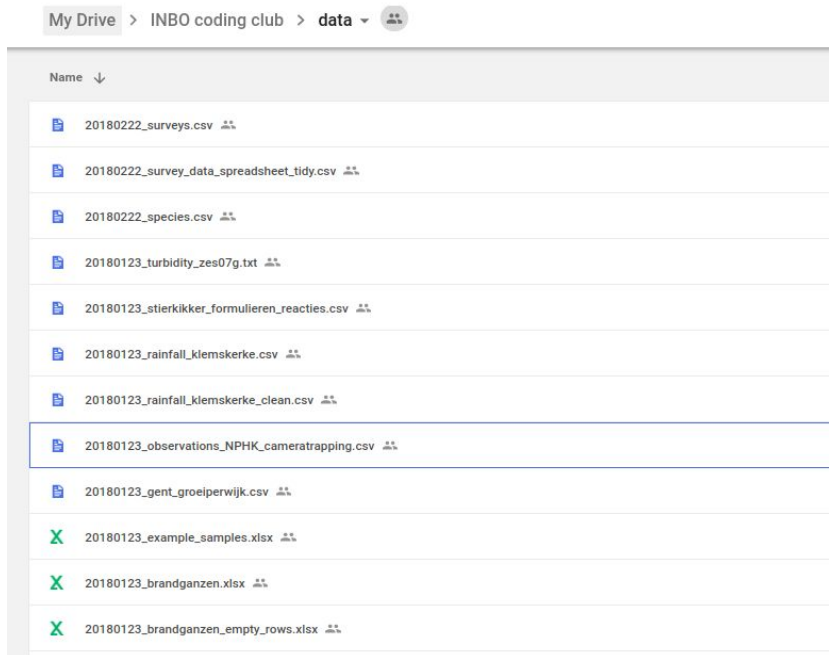




- Download coding club material and work locally, not in sync with the Google drive
- Create new Rstudio project in the **src** folder...
- Use relative paths to data files!

```
> library(readr)
```

```
> read_csv2("../data/20180123_gent_groeiperwijk.csv")
```



For this coding club:

20180426\_visdata\_cleaned.csv

20180123\_observations\_NPHK\_cameratrapping.csv



```
library(tidyverse)
vis_data <- read_csv(file = "../data/20180426_visdata_cleaned.csv")
```

copy-paste your  
solutions to [hackmd](#)

Tidyverse the following code using pipes and tidyverse functions:

1. Select a specific set of columns:

```
vis_data[, c("datum", "meetpuntomschrijving", "soort", "aantal", "gewicht")]
```

2. Filter the data to only `Zandplaat Kastel` for variable `meetomschrijving`:

```
vis_data[vis_data$meetpuntomschrijving == "Zandplaat Kastel",]
```

3. Subset the species (`soort`) to `snoekbaars`, `paling` and `spiering`:

```
vis_data[vis_data$soort %in% c("snoekbaars", "paling", "spiering"),]
```

4. Create a new column `year` derived from the `datum` column:

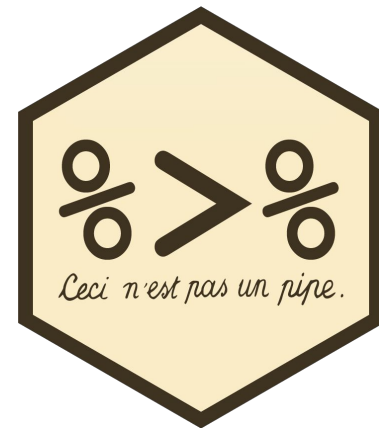
```
vis_data$year <- factor(lubridate::year(vis_data$datum))
```

5. Extract a sorted list of the the species names:

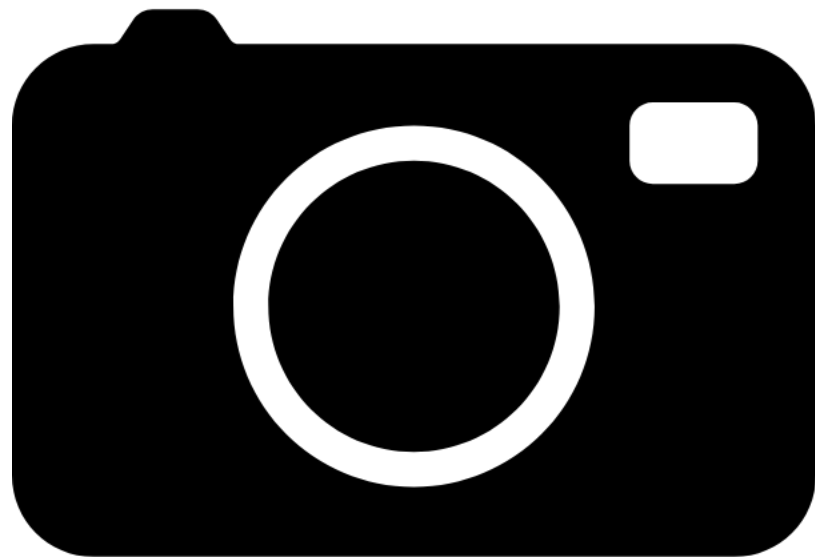
```
sort(unique(vis_data$soort))
```

# Piping recap

```
vis_data <- vis_data[, c("datum", "meetpuntomschrijving",
 "soort", "aantal", "gewicht")]
vis_data <- vis_data[vis_data$meetpuntomschrijving == "Zandplaat Kastel",]
vis_data <- vis_data[vis_data$soort %in% c("snoekbaars", "paling",
 "spiering"),]
vis_data <- vis_data$year <- factor(lubridate::year(vis_data$datum))
```









# List Column Workflow

Nested data frames use a **list column**, a list that is stored as a column vector of a data frame. A typical **workflow** for list columns:

## 1 Make a list column

Species	SL	SW	PL	PW
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
versicol	7.0	3.2	4.7	1.4
versicol	6.4	3.2	4.5	1.5
versicol	6.9	3.1	4.9	1.5
versicol	5.5	2.3	4.0	1.3
virginica	6.3	3.3	6.0	2.5
virginica	5.8	2.7	5.1	1.9
virginica	7.1	3.0	5.9	2.1
virginica	6.3	2.9	5.6	1.8

```
n_iris <- iris %>%
 group_by(Species) %>%
 nest()
```

Species	data
setosa	<tibble [50x4]>
versicol	<tibble [50x4]>
virginica	<tibble [50x4]>

## 2 Work with list columns

Species	data	model
setosa	<tibble [50x4]>	<S3: lms>
versicol	<tibble [50x4]>	<S3: lms>
virginica	<tibble [50x4]>	<S3: lms>

```
mod_fun <- function(df)
 lm(Sepal.Length ~ ., data = df)

m_iris <- n_iris %>%
 mutate(model = map(data, mod_fun))
```

## 3 Simplify the list column

Species	beta
setosa	2.35
versicol	1.89
virginica	0.69

```
b_fun <- function(mod)
 coefficients(mod)[1]

m_iris %>% transmute(Species,
 beta = map_dbl(model, b_fun))
```

Calculate the quantiles 0%, 50% and 100% of the fish spherical density distribution (an available function called `spherical_density_distribution`) for each combination of year (year) and location (meetpuntnummer).

You can start from the [20181129\\_challenge\\_3.R](#) script in the `src`-folder.

### SHORTCUTS - within a purrr function:

"name" becomes **function(x) x\$name**. e.g. `map(l, "a")` extracts \$a from each element of l

~ . becomes **function(x) x**. e.g. `map(l, ~ 2 + .)` becomes `map(l, function(x) 2 + x)`

~ .x.y becomes **function(.x, .y) .x.y**. e.g. `map2(l, p, ~ .x + .y)` becomes `map2(l, p, function(l, p) l + p)`

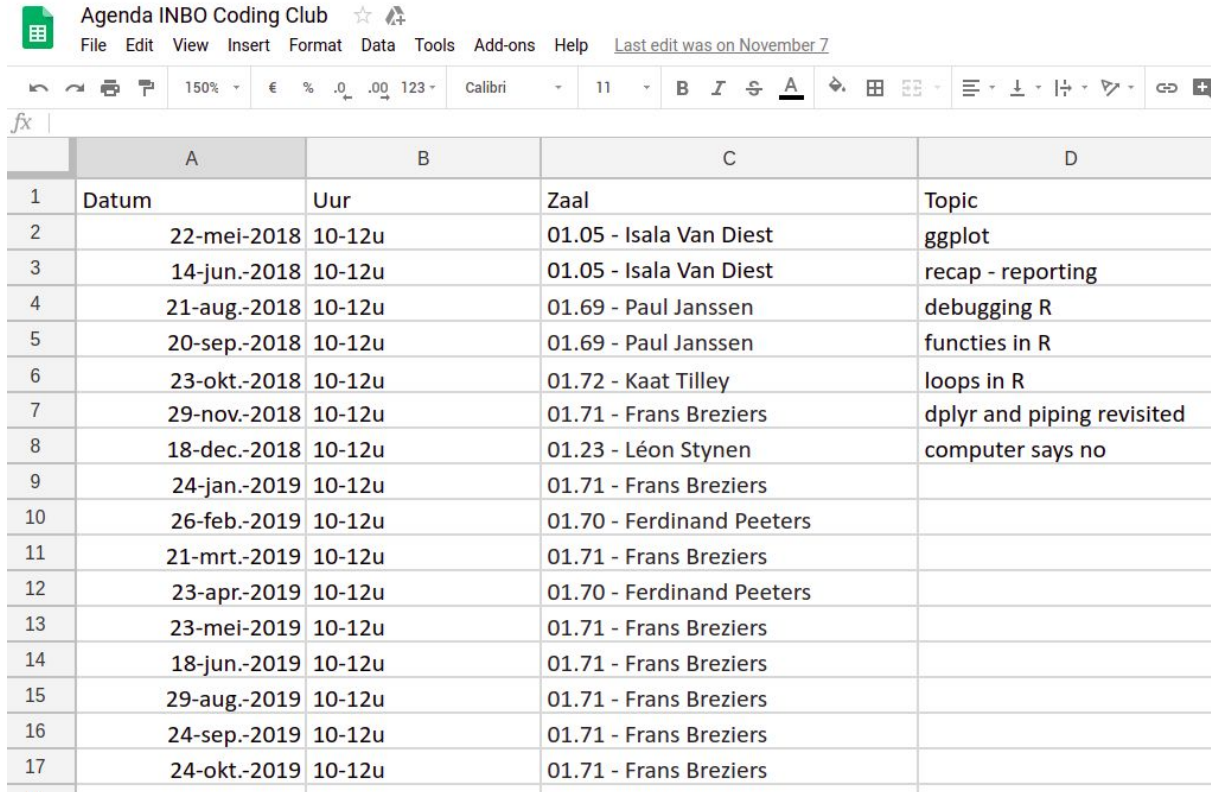
~ ..1 ..2 etc becomes **function(..1, ..2, etc) ..1 ..2 etc**. e.g. `pmap(list(a, b, c), ~ ..3 + ..1 - ..2)` becomes `pmap(list(a, b, c), function(a, b, c) c + a - b)`





TIME FOR REVIEW

# Check the [agenda for next year](#) (intranet page):



The screenshot shows a Google Sheets spreadsheet titled "Agenda INBO Coding Club". The spreadsheet has a header row with columns A, B, C, and D. The data rows contain dates, times, locations, and topics. The events are listed from 2018 to 2019.

	A	B	C	D
1	Datum	Uur	Zaal	Topic
2	22-mei-2018	10-12u	01.05 - Isala Van Diest	ggplot
3	14-jun.-2018	10-12u	01.05 - Isala Van Diest	recap - reporting
4	21-aug.-2018	10-12u	01.69 - Paul Janssen	debugging R
5	20-sep.-2018	10-12u	01.69 - Paul Janssen	functies in R
6	23-okt.-2018	10-12u	01.72 - Kaat Tilley	loops in R
7	29-nov.-2018	10-12u	01.71 - Frans Breziers	dplyr and piping revisited
8	18-dec.-2018	10-12u	01.23 - Léon Stynen	computer says no
9	24-jan.-2019	10-12u	01.71 - Frans Breziers	
10	26-feb.-2019	10-12u	01.70 - Ferdinand Peeters	
11	21-mrt.-2019	10-12u	01.71 - Frans Breziers	
12	23-apr.-2019	10-12u	01.70 - Ferdinand Peeters	
13	23-mei-2019	10-12u	01.71 - Frans Breziers	
14	18-jun.-2019	10-12u	01.71 - Frans Breziers	
15	29-aug.-2019	10-12u	01.71 - Frans Breziers	
16	24-sep.-2019	10-12u	01.71 - Frans Breziers	
17	24-okt.-2019	10-12u	01.71 - Frans Breziers	





Zaal: Herman Teirlinck - 01.23 - Léon Stynen

Datum: 2018-12-18, van 10:00 tot 12:00

*(registration announced via [DG\\_useR@inbo.be](mailto:DG_useR@inbo.be))*