

22 Februari 2018

Herman Teirlinck,
01.71 - Frans Breziers



DATA
MANIPULATION

Rstudio tips & tricks

1. open a new script in Rstudio
2. type ``ts``
3. press the TAB-button
4. press ENTER

You know other interesting shortcuts/tips? Add them to the shared board:

<https://hackmd.io/s/S1CfkMovz>

(you received the link in a mail ;-)

Data Wrangling with dplyr and tidyr

Cheat Sheet



Syntax - Helpful conventions for wrangling

dplyr: `tbl_df(iris)`

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1             5.1           3.5         1.4
2             4.9           3.0         1.4
3             4.7           3.2         1.3
4             4.6           3.1         1.5
5             5.0           3.6         1.4
..          ...           ...         ...
Variables not shown: Petal.Width (dbl),
Species (fctr)
```

dplyr: `glimpse(iris)`

Information dense summary of tbl data.

utils: `View(iris)`

View data set in spreadsheet-like display (note capital V).

```
View(iris)
  Sepal.Length Sepal.Width Petal.Length Species
1             5.1           3.5         1.4 setosa
2             4.9           3.0         1.4 setosa
3             4.7           3.2         1.3 setosa
4             4.6           3.1         1.5 setosa
5             5.0           3.6         1.4 setosa
6             4.8           3.4         1.4 setosa
7             5.1           3.5         1.4 setosa
```

dplyr: `%>%`

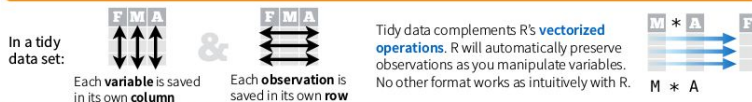
Passes object on left hand side as first argument (or argument) of function on righthand side.

`x %>% f(y)` is the same as `f(x, y)`
`y %>% f(x, ., z)` is the same as `f(x, y, z)`

"Piping" with `%>%` makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

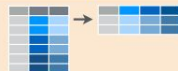
Tidy Data - A foundation for wrangling in R



Reshaping Data - Change the layout of a data set



tidyr: `gather(cases, "year", "n", 2:4)`
 Gather columns into rows.



tidyr: `spread(pollution, size, amount)`
 Spread rows into columns.



tidyr: `separate(storms, date, c("y", "m", "d"))`
 Separate one column into several.



tidyr: `unite(data, col, ..., sep)`
 Unite several columns into one.

dplyr: `data_frame(a = 1:3, b = 4:6)`
 Combine vectors into data frame (optimized).
dplyr: `arrange(mtcars, mpg)`
 Order rows by values of a column (low to high).
dplyr: `arrange(mtcars, desc(mpg))`
 Order rows by values of a column (high to low).
dplyr: `rename(tb, y = year)`
 Rename the columns of a data frame.

Subset Observations (Rows)



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

dplyr: `distinct(iris)`
 Remove duplicate rows.

dplyr: `sample_frac(iris, 0.5, replace = TRUE)`
 Randomly select fraction of rows.

dplyr: `sample_n(iris, 10, replace = TRUE)`
 Randomly select n rows.

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

dplyr: `top_n(storms, 2, date)`
 Select and order top n entries (by group if grouped data).

Subset Variables (Columns)

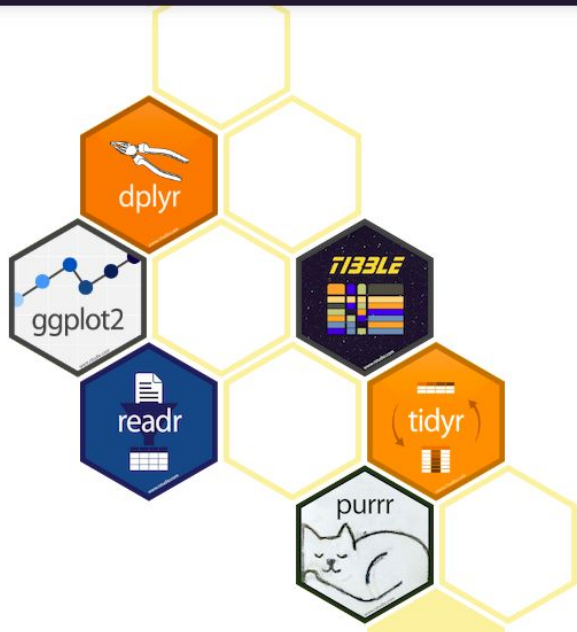


dplyr: `select(iris, Sepal.Width, Petal.Length, Species)`
 Select columns by name or helper function.

Helper functions for select - ?select

- `select(iris, contains("l"))`
Select columns whose name contains a character string.
- `select(iris, ends_with("Length"))`
Select columns whose name ends with a character string.
- `select(iris, everything())`
Select every column.
- `select(iris, matches("t"))`
Select columns whose name matches a regular expression.
- `select(iris, num_range("x", 1-5))`
Select columns named x1, x2, x3, x4, x5.
- `select(iris, one_of("Species", "Genus"))`
Select columns whose names are in a group of names.
- `select(iris, starts_with("Sepal"))`
Select columns whose name starts with a character string.
- `select(iris, Sepal.Length:Petal.Width)`
Select all columns between Sepal.Length and Petal.Width (inclusive).
- `select(iris, -Species)`
Select all columns except Species.

Logic in R - ?Comparison, ?base:Logic			
<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
>=	Less than or equal to	is.na	Is not NA
>>	Greater than or equal to	%, , !, xor, any, all	Boolean operators



R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

```
install.packages("tidyverse")
```

Learn the tidyverse

See how the tidyverse makes data science faster, easier and more fun with "R for Data Science". Read it [online](#),

The **tidyverse** is an opinionated collection of **R packages** for **data science**.

All packages share an underlying **design philosophy, grammar** and **data structures**

Install the package suite:

```
install.packages("tidyverse")
```

Load the package suite:

```
library(tidyverse)
```

Share your snippets during the coding session!


Go to <https://hackmd.io/s/S1CfkMovz> and post your code in between backticks:

For example:

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



Het  concept

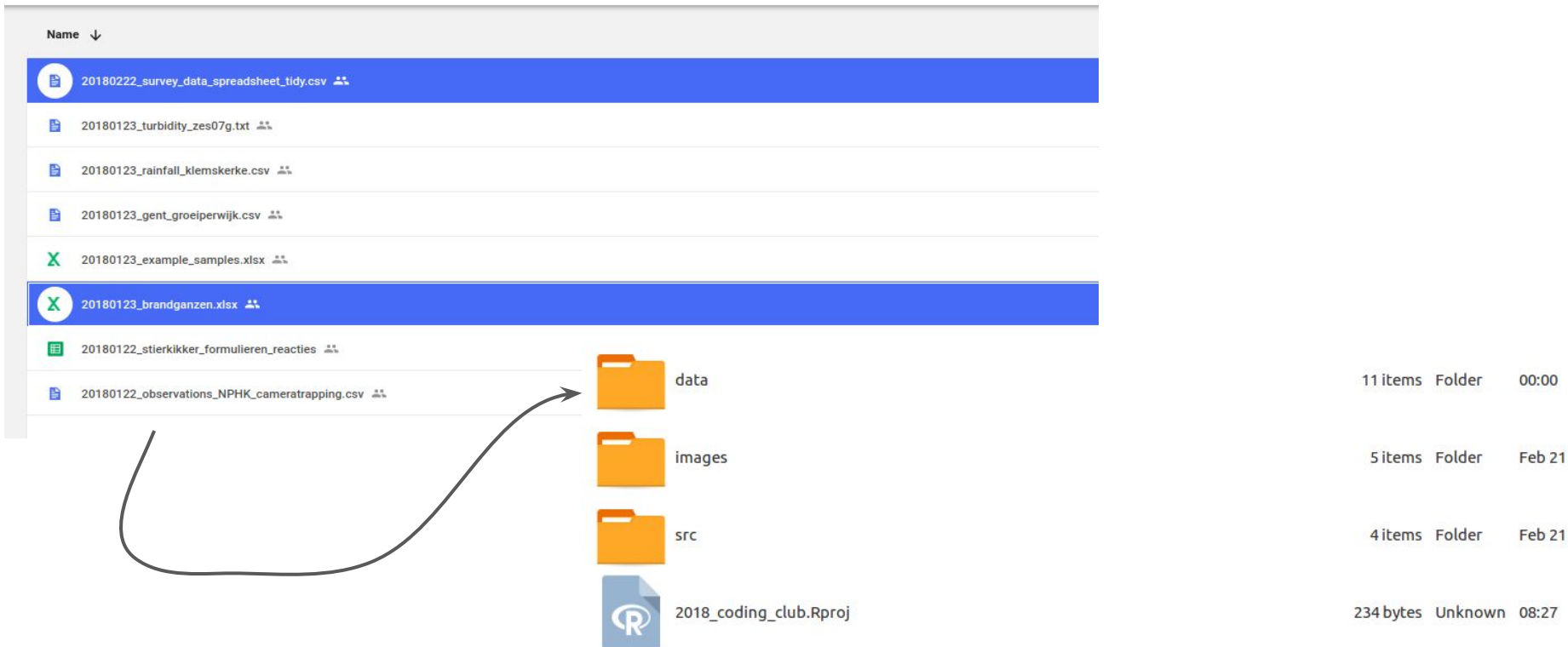
We bepaalden een aantal challenges. Als je zelf een challenge hebt bereikt, voeg dan een  toe aan je laptopscherm.

Het doel is dat **iedereen**  behaalt

- iemand met meer  dan jij? **Vraag hulp!**
- iemand met minder  dan jij? **Geef hulp!**

- Work locally, not in sync with the google drive!
- Create R project and use relative paths to data files: e.g. filename <- “./folder1/folder2/filename”

My Drive > INBO coding club > data ▾



Name	Size	Type	Modified
20180222_survey_data_spreadsheet_tidy.csv		File	
20180123_turbidity_zes07g.txt		File	
20180123_rainfall_klemskerke.csv		File	
20180123_gent_groeperwijk.csv		File	
20180123_example_samples.xlsx		File	
20180123_brandganzen.xlsx		File	
20180122_stierkikker_formulieren_reacties		File	
20180122_observations_NPHK_cameratrapping.csv		File	
data	11 items	Folder	00:00
images	5 items	Folder	Feb 21
src	4 items	Folder	Feb 21
2018_coding_club.Rproj	234 bytes	Unknown	08:27

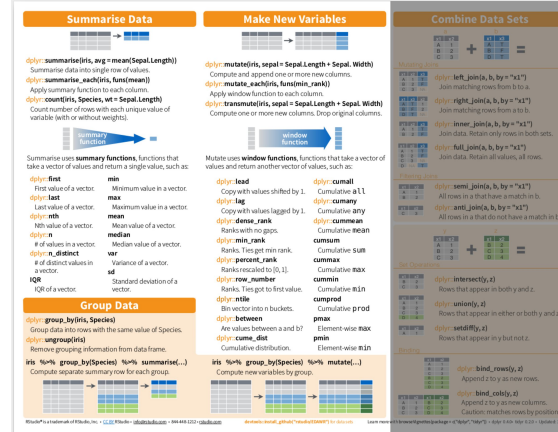
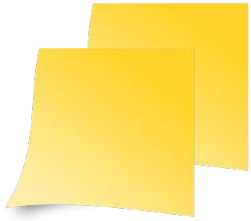
# Import data and let's start!

To do the first challenges, import data from:

1. [20180222\\_survey\\_data\\_spreadsheet\\_tidy.csv](#)
2. [20180123\\_brandganzen.xlsx](#)

Take a `glimpse()` on your freshly loaded data...





# survey\_data\_spreadsheet\_tidy.csv

1. Show min, max, mean weight per sex and species and save as a new object (df) ``weight_per_species_sex``

2. Execute the following:

- a. Rename column 'weight\_in\_g' to 'weight'
- b. Replace 'weight' values with values in kg
- c. Add column 'country' with value US
- d. Save as new object 'data\_kg\_US'

# 20180123\_brandganzen.xlsx

1. How many adult geese per sex can you count (consider 'onbekend' as a sex)?

```
Geslacht n
 <chr> <int>
1 Man 63
2 Onbekend 19
3 Vrouw 119
```

2. How many different catching methods were used in each location?

```
`Locatie vangst` n_methods
 <chr> <int>
1 DEINZE 1
2 DESTELBERGEN 1
...
```

%>%

magrittr

*Ceci n'est pas un pipe.*



# Read in the data set

[20180123\\_gent\\_groeiperwijk.csv](#)

This is NOT a *tidy* data set!

Make this a tidy data set:

wijk	year	growth
<chr>	<int>	<int>
1 Binnenstad	1999	- 36
2 Bloemekenswijk	1999	12
3 Brugse Poort - Rooigem	1999	85
4 Dampoort	1999	107
5 Drongen	1999	3
6 Elisabethbegijnhof - Papegaai	1999	- 4
7 Gentbrugge	1999	4
8 Kanaaldorpen en -zone	1999	5
9 Ledeborg	1999	- 4
10 Macharius - Heirnis	1999	47
# ... with 265 more rows		

The collage contains several cheat sheets:

- Data Wrangling with dplyr and tidy:** Explains the 'tidy' data set concept where each variable is a column and each observation is a row. It lists helpful conventions for wrangling.
- Tidy Data - A foundation for wrangling in R:** Discusses vectorized operations in R and how they differ from other languages like MATLAB.
- Reshaping Data:** Details functions like `gather()>spread()`, `long_to_wide()`, and `wide_to_long()` for changing data layout.
- Subset Observations (Rows):** Lists functions like `filter()`, `select()`, `distinct()`, `sample_n()`, and `slice()` for row selection.
- Subset Variables (Columns):** Lists functions like `select()`, `rename()`, `rename_with()`, and `select_at()` for column selection.



Read in the [20180222\\_surveys.csv](#) and the [20180222\\_species.csv](#) data.

**Summarise Data**

- `dplyr::summarise(iris, avg = mean(Sepal.Length))` Summarize data into single row of values.
- `dplyr::summarise_each(iris, fun=mean)` Apply summary function to each column.
- `dplyr::count(iris, Species, wt = Sepal.Length)` Count number of rows with each unique value of variable (with or without weights).

**Make New Variables**

- `dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)` Compute and append one or more new columns.
- `dplyr::mutate_each(iris, fun=mean)` Apply window function to each column.
- `dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)` Compute one or more new columns. Drop original columns.

**Join Data Sets**

- `dplyr::left_join(a, b, by = "x1")` Join matching rows from b to a.
- `dplyr::right_join(a, b, by = "x1")` Join matching rows from a to b.
- `dplyr::inner_join(a, b, by = "x1")` Join data. Retain only rows in both sets.
- `dplyr::full_join(a, b, by = "x1")` Join data. Retain all values, all rows.
- `dplyr::semi_join(a, b, by = "x1")` All rows in a that have a match in b.
- `dplyr::anti_join(a, b, by = "x1")` All rows in a that do not have a match in b.

**Group Data**

- `dplyr::group_by(iris, Species)` Group data into rows with the same value of Species.
- `dplyr::ungroup(iris)` Remove grouping information from data frame.
- `iris %>% group_by(Species) %>% summarise(...)` Compute separate summary row for each group.

**Other Functions**

- `dplyr::lead`: Copy with values shifted by 1.
- `dplyr::lag`: Copy with values lagged by 1.
- `dplyr::dense_rank`: Ranks with no gaps.
- `dplyr::min_rank`: Ranks. Ties get min rank.
- `dplyr::percent_rank`: Ranks rescaled to [0, 1].
- `dplyr::row_number`: Ranks. Tie get to first value.
- `dplyr::ntile`: Bin vector into n buckets.
- `dplyr::between`: Are values between a and b?
- `dplyr::cumc_dist`: Cumulative distribution.
- `dplyr::cumall`: Cumulative all 1.
- `dplyr::cumany`: Cumulative any 1.
- `dplyr::cummean`: Cumulative mean.
- `dplyr::cumsum`: Cumulative sum.
- `dplyr::cumsum`: Cumulative sum.
- `dplyr::cummax`: Cumulative MAX.
- `dplyr::cummin`: Cumulative MIN.
- `dplyr::pnorm`: Element-wise max.
- `dplyr::pmin`: Element-wise min.
- `dplyr::bind_rowwise(z)`: Append z by as new rows.
- `dplyr::bind_colwise(z)`: Append z to y as new columns. Caution: matches rows by position.

Join the species information columns (genus, species , taxa) to the survey data set, using the common identifier.

Compare the result when applying the different commands to join the data...



**More challenges!**

For the [20180123\\_observations\\_NPHK\\_cameratrapping.csv](#) data:

- count the observed humans for each month:

```
sequenceMonth humans_observed
 <int> <int>
1 5 1
2 6 1
3 7 38
4 8 153
5 9 38
6 10 25
```

- add an additional column with the counts for each animal/samplingPoint combination

```
sequenceDay sequenceMonth sequenceYear deploymentSamplingPoint animalVernacularName animalCount point_animal_counts
 <int> <int> <int> <chr> <chr> <int> <int>
1 7 7 2017 JW_0090 Ass 1 12
2 6 7 2017 JW_0090 Ass 3 12
3 7 7 2017 JW_0090 Ass 1 12
4 15 7 2017 JW_0090 Ass 1 12
5 16 7 2017 JW_0090 Ass 2 12
6 27 7 2017 JW_0090 Ass 2 12
7 27 7 2017 JW_0090 Ass 1 12
8 27 7 2017 JW_0090 Ass 1 12
9 13 5 2017 JW_0020 Beech Marten 1 3
10 13 5 2017 JW_0020 Beech Marten 1 3
... with 3,750 more rows
```

For the [stierkikker](#) formulieren data, derive all the columns concerning `blankvoorn` and remove those rows for which all values are NA:

```
A tibble: 95 x 12
 `Fuik 1 - Bijvangs...` `Fuik 2 - Bijvangs...` `Fuik 3 - Bijvangs...` `Fuik 4 - Bijvangs...` `Fuik 5 - Bijvang...` `Fuik 6 - Bijvang...`
 <chr> <chr> <chr> <chr> <chr> <chr>
1 NA 5-10cm NA NA NA NA
2 ?? ?? NA NA NA NA
3 ?? NA NA NA NA NA
4 ?? NA NA NA NA NA
5 ?? NA NA NA NA NA
6 ?? NA NA NA NA NA
7 5-10cm NA NA NA NA NA
8 5-10cm NA NA NA NA NA
9 5-10cm 5-10cm 5-10cm NA NA NA
10 NA 5-10cm 5-10cm 5-10cm 5-10cm 5-10cm
... with 85 more rows, and 6 more variables: `Fuik 7 - Bijvangst [Blankvoorn]` <chr>, `Fuik 8 - Bijvangst
[Blankvoorn]` <chr>, `Fuik 9 - Bijvangst [Blankvoorn]` <chr>, `Fuik 10 - Bijvangst [Blankvoorn]` <chr>, `Fuik 11 -
Bijvangst [Blankvoorn]` <chr>, `Fuik 12 - Bijvangst [Blankvoorn]` <chr>
```

For the [20180123\\_rainfall\\_klemskerke\\_clean.csv](#) data,  
calculate the yearly rainfall sum from 2012 till 2016:

	year		value
	<dtm>		<dbl>
1	2012-01-01 00:00:00		934
2	2013-01-01 00:00:00		701
3	2014-01-01 00:00:00		727
4	2015-01-01 00:00:00		789
5	2016-01-01 00:00:00		775

# More tidyverse/dplyr?

- Workshop on 13 maart @HT

Register at

<http://www.vib.be/en/training/research-training/courses/Pages/Elixir-tidyverse-Intro-March2018-Bru.aspx>

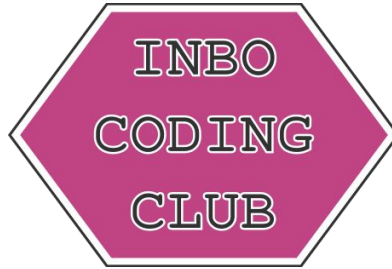
- More tidyverse courses/webinars/...:

<https://inbo-tutorials.netlify.com/data-handling/tidyverse/>

- R for data scientists: <http://r4ds.had.co.nz/>



TIME FOR REVIEW



Zaal: Herman Teirlinck - 01.17 - Clara Peeters

Datum: 20/03/2018, van 10:00 tot 12:00

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