INBO CODING CLUB

20 Maart 2018

Herman Teirlinck,
01.17 - Clara Peeters

What have I done?!?

```r
library(rvest)
library(dplyr)
library(magrittr)

waarnemingen <- read_html("https://waarnemingen.be/")
waarnemingen %>%
    html_nodes("table") %>%
    .[[3]] %>%
    html_table() %>%
    set_colnames(c("intro", "datum", "count", "soort", "photo", "gebied")) %>%
    select(datum, count, soort, gebied) %>%
    slice(-1)
```

Note: Rvest is a ' *tidyverse-adjacent*' package

DATA MANIPULATION

PART 2

Data Wrangling with dplyr and tidyr Cheat Sheet

RStudio

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:

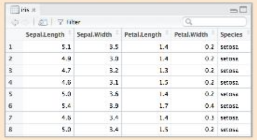
```
Source: local data frame [150 x 5]

  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).



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::**%>%**

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

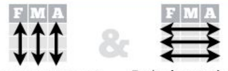
```
        x %>% f(y)  is the same as  f(x, y)
  y %>% f(x, ., z)  is the same as  f(x, y, z )
```

"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

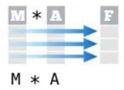
In a tidy data set:



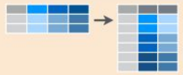
Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

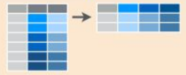


Reshaping Data - Change the layout of a data set



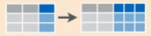
tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.



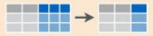
tidyr::**spread(pollution, size, amount)**

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).

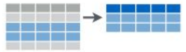
dplyr::**arrange(mtcars, desc(mpg))**

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)**

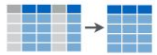
Select rows by position.

dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

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

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(".")**)
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.

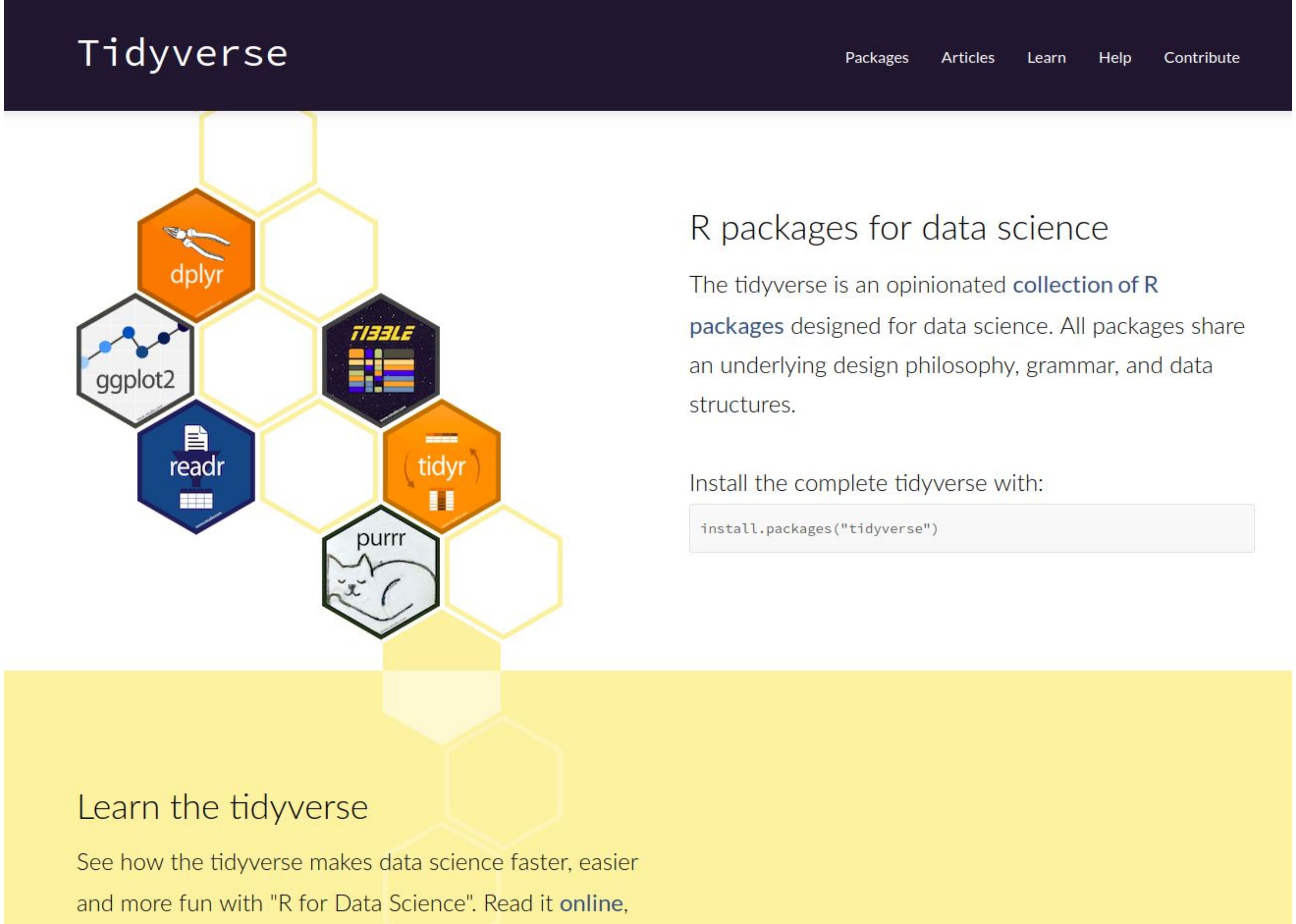
select(iris, **one_of(c("Species", "Genus"))**)
Select columns whose names are in a group of names.

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.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com   devtools::install_github("rstudio/EDAWR") for data sets   Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15



Tidyverse

Packages Articles Learn Help Contribute

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,

Install the package suite:

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

Load the package suite:

```
library(tidyverse)
```

TIDY?!?

See https://inbo.github.io/dwc-in-R/tidy.html#14

Share your snippets during the coding session!

Go to https://hackmd.io/7Yd3NsCFTwqHbRnHZbhlzg and post your code in between backticks:

For example:

~~~
```
library(dplyr)
my_data <- ...
```
~~~

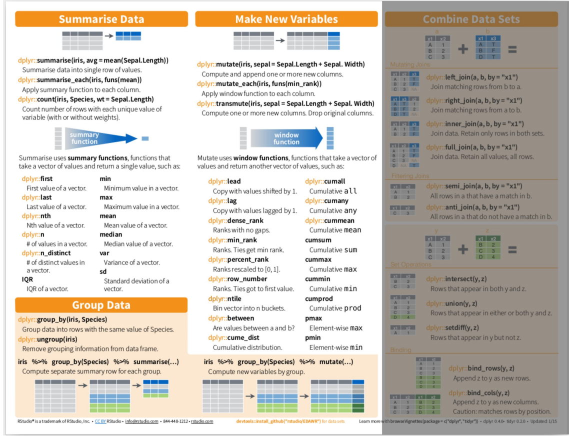
Excel might contain column names with capital letters, spaces, etc., which can be annoying to select:

```
brandganzen <- read_excel("./data/20180123_brandganzen.xlsx")

brandganzen %>% select(`Locatie vangst`) # Ugh
```

With `janitor` your column names can be cleaned (lowercase, underscores instead of spaces). In addition, you can remove empty rows:

```
library(janitor) # Also tidyverse, but not loaded by default

brandganzen %>%
  remove_empty_rows() %>% # Additional step to remove empty rows
  clean_names() -> brandganzen

brandganzen %>% select(locatie_vangst)
```

recap/showcase



Summarise Data

dplyr::**summarise(iris, avg = mean(Sepal.Length))**
Summarise data into single row of values.
dplyr::**summarise_each(iris, funs(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).

summary function

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first** First value of a vector.
dplyr::**last** Last value of a vector.
dplyr::**nth** Nth value of a vector.
dplyr::**n** # of values in a vector.
dplyr::**n_distinct** # of distinct values in a vector.
IQR IQR of a vector.
min Minimum value in a vector.
max Maximum value in a vector.
mean Mean value of a vector.
median Median value of a vector.
var Variance of a vector.
sd Standard deviation of a vector.

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.

Make New Variables

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

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

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. Ties got to first value.
dplyr::**ntile** Bin vector into n buckets.
dplyr::**between** Are values between a and b?
dplyr::**cume_dist** Cumulative distribution.
dplyr::**cumall** Cumulative `all`
dplyr::**cumany** Cumulative `any`
dplyr::**cummean** Cumulative `mean`
cumsum Cumulative `sum`
cummax Cumulative `max`
cummin Cumulative `min`
cumprod Cumulative `prod`
pmax Element-wise `max`
pmin Element-wise `min`

iris %>% group_by(Species) %>% mutate(...)
Compute new variables by group.

Combine Data Sets

Mutating Joins

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.

Filtering Joins

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.

Set Operations

dplyr::**intersect(y, z)**
Rows that appear in both y and z.
dplyr::**union(y, z)**
Rows that appear in either or both y and z.
dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

Binding

dplyr::**bind_rows(y, z)**
Append z to y as new rows.
dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com
devtools::install_github("rstudio/EDAWR") for data sets
Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

`20180222_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
...
```

recap/showcase

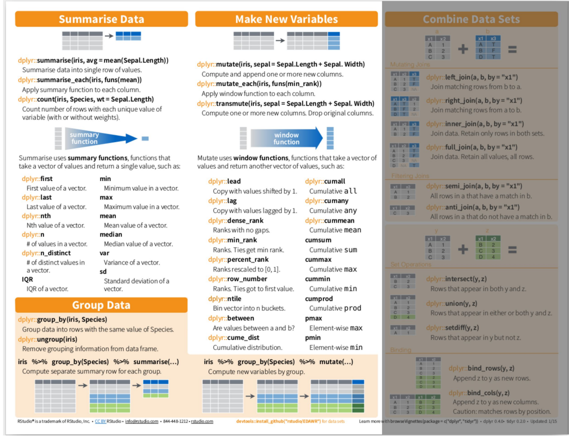
20180222_survey_data_spreadsheet_tidy.csv

1. Show min, max, mean w nd
   species an object (df)
   `w _species_sex`

group_by & summarise

2. Execute the following:
   a. Rename column 'weight_in_g' to
      'weight'
   b. Replace values in kg
   c. country' with value 'US'
   d. Save as new object 'data_kg_US'

rename & mutate



Summarise Data

dplyr::**summarise(iris, avg = mean(Sepal.Length))**
Summarise data into single row of values.
dplyr::**summarise_each(iris, funs(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).

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first** First value of a vector.	**min** Minimum value in a vector.
dplyr::**last** Last value of a vector.	**max** Maximum value in a vector.
dplyr::**nth** Nth value of a vector.	**mean** Mean value of a vector.
dplyr::**n** # of values in a vector.	**median** Median value of a vector.
dplyr::**n_distinct** # of distinct values in a vector.	**var** Variance of a vector.
IQR IQR of a vector.	**sd** Standard deviation of a vector.

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.

Make New Variables

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

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::**lead** Copy with values shifted by 1.	dplyr::**cumall** Cumulative `all`
dplyr::**lag** Copy with values lagged by 1.	dplyr::**cumany** Cumulative `any`
dplyr::**dense_rank** Ranks with no gaps.	dplyr::**cummean** Cumulative `mean`
dplyr::**min_rank** Ranks. Ties get min rank.	**cumsum** Cumulative `sum`
dplyr::**percent_rank** Ranks rescaled to [0, 1].	**cummax** Cumulative `max`
dplyr::**row_number** Ranks. Ties got to first value.	**cummin** Cumulative `min`
dplyr::**ntile** Bin vector into n buckets.	**cumprod** Cumulative `prod`
dplyr::**between** Are values between a and b?	**pmax** Element-wise `max`
dplyr::**cume_dist** Cumulative distribution.	**pmin** Element-wise `min`

iris %>% group_by(Species) %>% mutate(...)
Compute new variables by group.

Combine Data Sets

Mutating Joins

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.

Filtering Joins

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.

Set Operations

dplyr::**intersect(y, z)**
Rows that appear in both y and z.
dplyr::**union(y, z)**
Rows that appear in either or both y and z.
dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

Binding

dplyr::**bind_rows(y, z)**
Append z to y as new rows.
dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com devtools::install_github("rstudio/EDAWR") for data sets Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

20180123_brandganzen.xlsx

1. How many adult geese p u
   count (x):

filter & group_by & count

```
              63
2 Onbekend    19
3 Vrouw      119
```

2. How many different catching s
   were used in
   `L

group_by & summarise

```
                              1
 LEBERGEN                     1
...
```

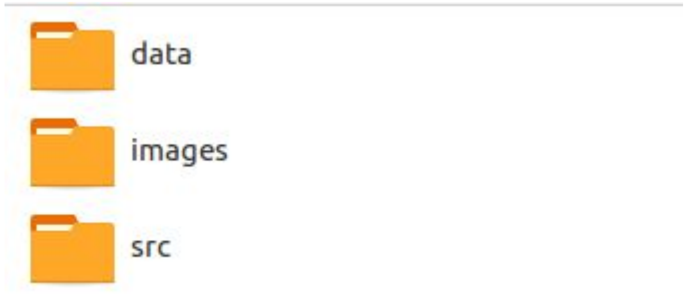
The concept

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

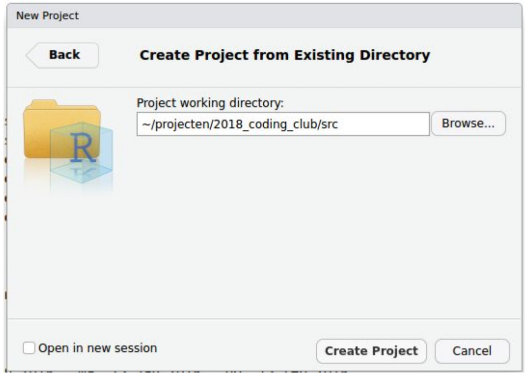
The objective is that **everyone** achieves !

- 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 the **/src** folder

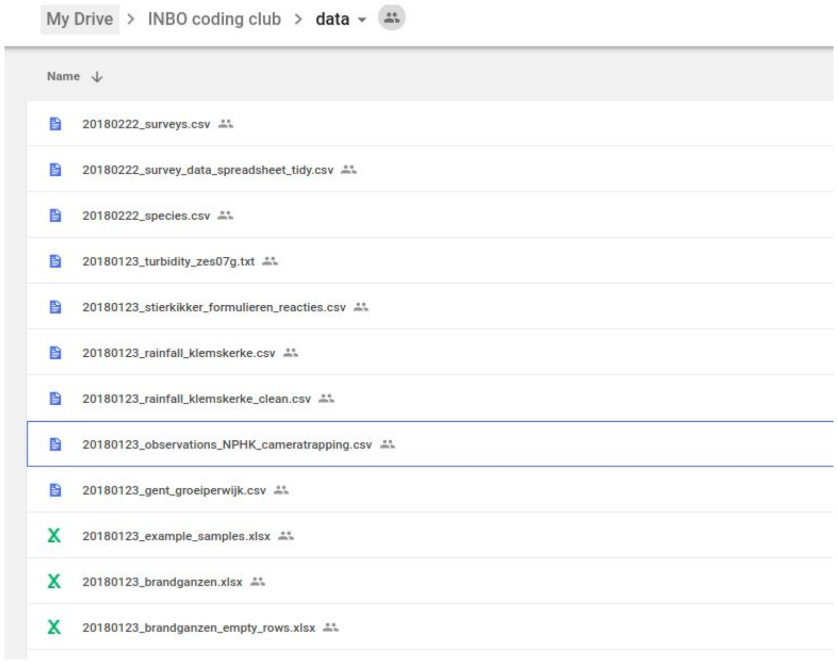


Project working directory:

```
~/projecten/2018_coding_club/src
```

- 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")
```


My Drive > INBO coding club > data

Name

20180222_surveys.csv
20180222_survey_data_spreadsheet_tidy.csv
20180222_species.csv
20180123_turbidity_zes07g.txt
20180123_stierkikker_formulieren_reacties.csv
20180123_rainfall_klemskerke.csv
20180123_rainfall_klemskerke_clean.csv
20180123_observations_NPHK_cameratrapping.csv
20180123_gent_groeiperwijk.csv
20180123_example_samples.xlsx
20180123_brandganzen.xlsx
20180123_brandganzen_empty_rows.xlsx

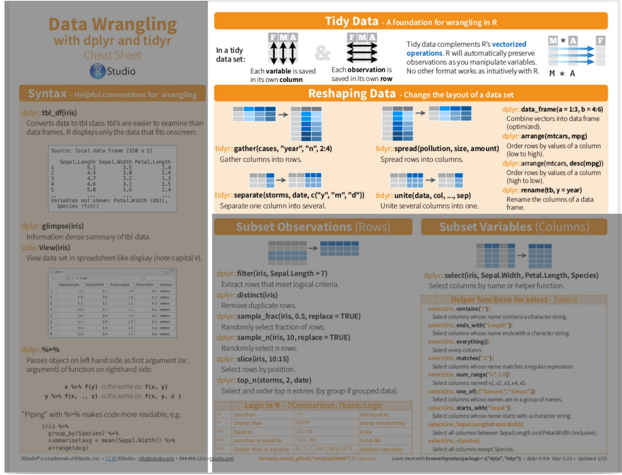
Read in the data set

[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 Ledeberg                               1999   -  4
10 Macharius - Heirnis                    1999     47
# ... with 265 more rows
```



Data Wrangling
with dplyr and tidyr
Cheat Sheet
RStudio

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).

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	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)
```

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

Tidy Data - A foundation for wrangling in R

In a tidy data set: Each **variable** is saved in its own **column** & Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with 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).

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

devtools::install_github("rstudio/EDAWR") for data sets

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(".")**)
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.

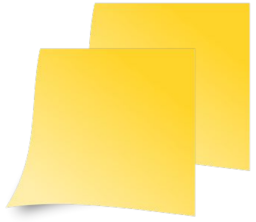
select(iris, **one_of(c("Species", "Genus"))**)
Select columns whose names are in a group of names.

select(iris, **starts_with("Sepal")**)
Select columns whose name starts with a character string.

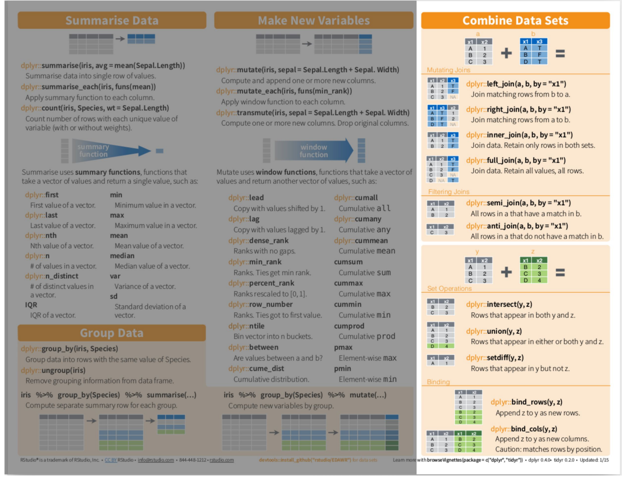
select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15



Read in the 20180222_surveys.csv and the 20180222_species.csv data.



Summarise Data

dplyr::**summarise(iris, avg = mean(Sepal.Length))**
Summarise data into single row of values.
dplyr::**summarise_each(iris, funs(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).

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first** First value of a vector.	**min** Minimum value in a vector.
dplyr::**last** Last value of a vector.	**max** Maximum value in a vector.
dplyr::**nth** Nth value of a vector.	**mean** Mean value of a vector.
dplyr::**n** # of values in a vector.	**median** Median value of a vector.
dplyr::**n_distinct** # of distinct values in a vector.	**var** Variance of a vector.
IQR IQR of a vector.	**sd** Standard deviation of a vector.

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.

Make New Variables

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

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::**lead** Copy with values shifted by 1.	dplyr::**cumall** Cumulative `all`
dplyr::**lag** Copy with values lagged by 1.	dplyr::**cumany** Cumulative `any`
dplyr::**dense_rank** Ranks with no gaps.	dplyr::**cummean** Cumulative `mean`
dplyr::**min_rank** Ranks. Ties get min rank.	**cumsum** Cumulative `sum`
dplyr::**percent_rank** Ranks rescaled to [0, 1].	**cummax** Cumulative `max`
dplyr::**row_number** Ranks. Ties got to first value.	**cummin** Cumulative `min`
dplyr::**ntile** Bin vector into n buckets.	**cumprod** Cumulative `prod`
dplyr::**between** Are values between a and b?	**pmax** Element-wise `max`
dplyr::**cume_dist** Cumulative distribution.	**pmin** Element-wise `min`

iris %>% group_by(Species) %>% mutate(…)
Compute new variables by group.

Combine Data Sets

Mutating Joins

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.

Filtering Joins

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.

Set Operations

dplyr::**intersect(y, z)**
Rows that appear in both y and z.
dplyr::**union(y, z)**
Rows that appear in either or both y and z.
dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

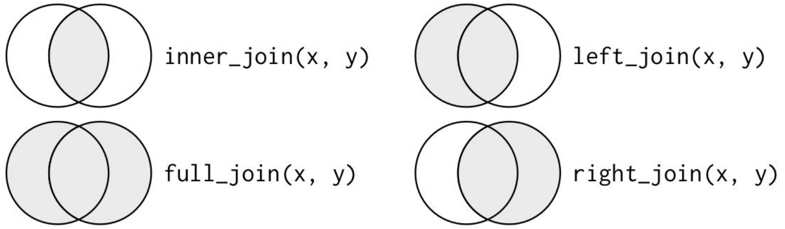
Binding

dplyr::**bind_rows(y, z)**
Append z to y as new rows.
dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com   devtools::install_github("rstudio/EDAWR") for data sets   Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

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...



Data used from: *Ernest, Morgan; Brown, James; Valone, Thomas; White, Ethan P. (2017): Portal Project Teaching Database. figshare.*

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-deploymentID 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
```

More challenges!

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>
```

Stackoverflow is your friend: [*https://stackoverflow.com/questions/41609912/remove-rows-where-all-variables-are-na-using-dplyr*](https://stackoverflow.com/questions/41609912/remove-rows-where-all-variables-are-na-using-dplyr)

For the `20180123_rainfall_klemskerke_clean.csv` data, calculate the yearly rainfall sum from 2012 till 2016:

```
 year                       value
   <dttm>                     <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
```

TIME FOR REVIEW

INBO CODING CLUB

Zaal: Herman Teirlinck - 01.21 - Jeanne Brabants

Datum: 26/04/2018, van 10:00 tot 12:00

(registratie aangekondigd via DG_useR@inbo.be)