

R programming (dplyr)

Welcome to the tidyverse

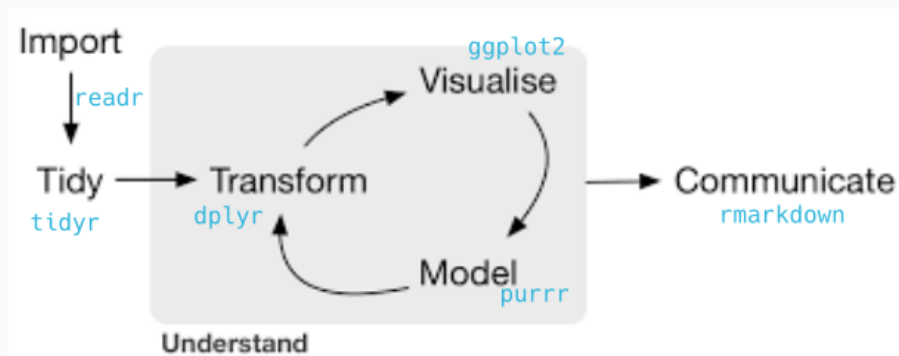
Víctor Granda (@MalditoBarbudo)
2019-03-26

The tidyverse

The **tidyverse** is a collection of R packages designed for data science, as a suite aimed at easing the data analysis in all its steps.

Created by Hadley Wickham, chief scientist of RStudio, and author of more than 30 R packages (`readr`, `ggplot2`, `plyr`, `devtools`, `roxygen2`, `rmarkdown`...)

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

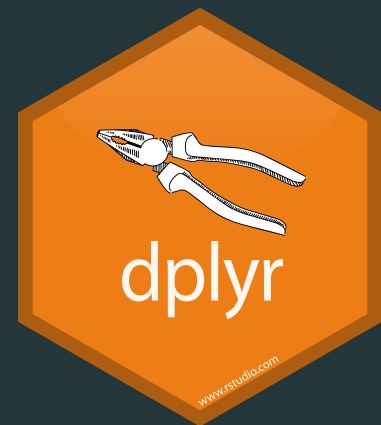


So what's exactly *in* the tidyverse?



- `ggplot2` a system for creating graphics, based on the Grammar of Graphics
- `readr` a fast and friendly way to read rectangular data (csv, txt...)
- `tibble` a tibble is a re-imagining version of the data frame, keeping what time has proven to be effective and throwing out what has not
- `stringr` provides a cohesive set of functions designed to make working with strings as easy as possible
- `forcats` provides a suite of useful tools that solve common problems with factors
- `dplyr` provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges
- `tidyr` provides a set of functions that help you get to tidy data
- `purrr` enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors

dplyr



5 main verbs of dplyr



- `filter`: keep the rows that match a condition
- `select`: keep columns by name
- `arrange`: sort rows
- `mutate`: transform existent variables or create new ones
- `summarise`: do some summary statistics and reduce data

(for most of the tidyverse)

```
verb(data, ...)
```

- first argument: data (as data.frame or tbl_df)
- the rest of arguments specify what to do with the data frame
- output is always another data frame (tbl_df or data.frame)
- unless we are assigning (`←`), never modifies the original data frame

`filter`

Let's work with some data. `dplyr` comes with some example data to get the feeling:

```
# install.packages(dplyr)
# install.packages(babynames)
library(dplyr)
library(babynames)
babynames
```

```
## # A tibble: 1,924,665 x 5
##   year sex  name      n  prop
##   <dbl> <chr> <chr>   <int> <dbl>
## 1  1880 F    Mary    7065 0.0724
## 2  1880 F    Anna    2604 0.0267
## 3  1880 F    Emma    2003 0.0205
## 4  1880 F  Elizabeth 1939 0.0199
## 5  1880 F   Minnie   1746 0.0179
## 6  1880 F  Margaret 1578 0.0162
## 7  1880 F    Ida     1472 0.0151
## 8  1880 F   Alice    1414 0.0145
## 9  1880 F   Bertha   1320 0.0135
## 10 1880 F    Sarah   1288 0.0132
## # ... with 1,924,655 more rows
```


Selecting rows (filter)



```
filter(babynames, name == 'Alice')
```

```
## # A tibble: 241 x 5
##   year sex  name      n      prop
##   <dbl> <chr> <chr> <int>    <dbl>
## 1  1880 F    Alice   1414  0.0145
## 2  1881 F    Alice   1308  0.0132
## 3  1881 M    Alice     7  0.0000646
## 4  1882 F    Alice   1542  0.0133
## 5  1883 F    Alice   1488  0.0124
## 6  1883 M    Alice     6  0.0000533
## 7  1884 F    Alice   1732  0.0126
## 8  1885 F    Alice   1681  0.0118
## 9  1885 M    Alice     9  0.0000776
## 10 1886 F    Alice   1811  0.0118
## # ... with 231 more rows
```

Selecting rows (filter)



```
filter(babynames, year > 2016)
```

```
## # A tibble: 32,469 x 5
##   year sex   name      n    prop
##   <dbl> <chr> <chr>   <int>  <dbl>
## 1  2017 F     Emma   19738 0.0105
## 2  2017 F    Olivia 18632 0.00994
## 3  2017 F     Ava   15902 0.00848
## 4  2017 F  Isabella 15100 0.00805
## 5  2017 F   Sophia 14831 0.00791
## 6  2017 F     Mia   13437 0.00717
## 7  2017 F  Charlotte 12893 0.00688
## 8  2017 F    Amelia 11800 0.00629
## 9  2017 F   Evelyn 10675 0.00569
## 10 2017 F  Abigail 10551 0.00563
## # ... with 32,459 more rows
```

Selecting rows (filter)



```
filter(babynames, name %in% c('Ada', 'Leon'))
```

```
## # A tibble: 411 x 5
##   year sex  name      n    prop
##   <dbl> <chr> <chr> <int>  <dbl>
## 1  1880 F    Ada     652 0.00668
## 2  1880 M    Leon    118 0.000997
## 3  1881 F    Ada     628 0.00635
## 4  1881 M    Leon    121 0.00112
## 5  1882 F    Ada     689 0.00596
## 6  1882 M    Leon    131 0.00107
## 7  1883 F    Ada     778 0.00648
## 8  1883 M    Leon    140 0.00124
## 9  1884 F    Ada     854 0.00621
## 10 1884 M    Leon    150 0.00122
## # ... with 401 more rows
```

Selecting rows (filter)









```
filter(  
  babynames,  
  sex = 'F',  
  prop > 0.07  
)
```

```
## # A tibble: 2 x 5  
##   year sex   name     n  prop  
##   <dbl> <chr> <chr> <int> <dbl>  
## 1  1880 F     Mary   7065 0.0724  
## 2  1882 F     Mary   8148 0.0704
```

Selecting rows (filter)



	a
	b
	a b
	a & b
	a & !b
	xor(a, b)

```
x > 1  
x >= 1  
x < 1  
x <= 1  
x != 1  
x == 1  
x %in% ("a", "b")
```

select

Selecting columns (select)



```
select(babynames, year)
```

```
## # A tibble: 1,924,665 x 1
##   year
##   <dbl>
## 1  1880
## 2  1880
## 3  1880
## 4  1880
## 5  1880
## 6  1880
## 7  1880
## 8  1880
## 9  1880
## 10 1880
## # ... with 1,924,655 more rows
```

Selecting columns (select)



```
select(babynames, -prop)
```

```
## # A tibble: 1,924,665 x 4
##   year sex   name      n
##   <dbl> <chr> <chr>   <int>
## 1  1880 F    Mary    7065
## 2  1880 F    Anna    2604
## 3  1880 F    Emma    2003
## 4  1880 F  Elizabeth 1939
## 5  1880 F   Minnie   1746
## 6  1880 F  Margaret  1578
## 7  1880 F    Ida     1472
## 8  1880 F   Alice    1414
## 9  1880 F  Bertha   1320
## 10 1880 F   Sarah    1288
## # ... with 1,924,655 more rows
```


Selecting columns (select)



```
select(babynames, sex, name)
```

```
## # A tibble: 1,924,665 x 2
##   sex    name
##   <chr> <chr>
## 1 F     Mary
## 2 F     Anna
## 3 F     Emma
## 4 F     Elizabeth
## 5 F     Minnie
## 6 F     Margaret
## 7 F     Ida
## 8 F     Alice
## 9 F     Bertha
## 10 F    Sarah
## # ... with 1,924,655 more rows
```

Selecting columns (select)



```
select(babynames, sex:n)
```

```
## # A tibble: 1,924,665 x 3
##   sex    name      n
##   <chr> <chr>    <int>
## 1 F      Mary     7065
## 2 F      Anna     2604
## 3 F      Emma     2003
## 4 F      Elizabeth 1939
## 5 F      Minnie    1746
## 6 F      Margaret  1578
## 7 F      Ida       1472
## 8 F      Alice     1414
## 9 F      Bertha    1320
## 10 F     Sarah     1288
## # ... with 1,924,655 more rows
```

Selecting columns (select)



Special functions:

- `starts_with(x)`: names that start with x
- `ends_with(x)`: names that end with x
- `contains(x)`: selects all variables whose name contains x
- `matches(x)`: selects all variables whose name contains the regular expression x
- `num_range("x", 1:5, width = 2)`: selects all variables (numerically) from x01 to x05
- `one_of ("x", "y", "z")`: selects variables provided in a character vector
- `everything()`: selects all variables

Selecting columns (select)



```
select(babynames, starts_with('n'))
```

```
## # A tibble: 1,924,665 x 2
##   name          n
##   <chr>      <int>
## 1 Mary        7065
## 2 Anna        2604
## 3 Emma        2003
## 4 Elizabeth   1939
## 5 Minnie      1746
## 6 Margaret    1578
## 7 Ida         1472
## 8 Alice       1414
## 9 Bertha      1320
## 10 Sarah       1288
## # ... with 1,924,655 more rows
```

arrange

Sorting rows (arrange)



```
arrange(babynames, prop)
```

```
## # A tibble: 1,924,665 x 5
##   year sex   name      n      prop
##   <dbl> <chr> <chr>    <int>    <dbl>
## 1  2007 M     Aaban        5 0.00000226
## 2  2007 M    Aareon        5 0.00000226
## 3  2007 M    Aaris        5 0.00000226
## 4  2007 M     Abd         5 0.00000226
## 5  2007 M Abdulazeez      5 0.00000226
## 6  2007 M Abdulhadi       5 0.00000226
## 7  2007 M Abdulhamid      5 0.00000226
## 8  2007 M Abdulkadir     5 0.00000226
## 9  2007 M Abdulraheem    5 0.00000226
## 10 2007 M Abdulrahim    5 0.00000226
## # ... with 1,924,655 more rows
```

Sorting rows (arrange)



```
arrange(babynames, desc(prop))
```

```
## # A tibble: 1,924,665 x 5
##   year sex   name      n  prop
##   <dbl> <chr> <chr>   <int> <dbl>
## 1  1880 M     John    9655 0.0815
## 2  1881 M     John    8769 0.0810
## 3  1880 M   William  9532 0.0805
## 4  1883 M     John    8894 0.0791
## 5  1881 M   William  8524 0.0787
## 6  1882 M     John    9557 0.0783
## 7  1884 M     John    9388 0.0765
## 8  1882 M   William  9298 0.0762
## 9  1886 M     John    9026 0.0758
## 10 1885 M     John    8756 0.0755
## # ... with 1,924,655 more rows
```

mutate

Transforming variables (mutate)



```
mutate(  
  babynames,  
  total = n / prop  
)
```

```
## # A tibble: 1,924,665 x 6  
##   year sex   name      n   prop total  
##   <dbl> <chr> <chr>   <int> <dbl> <dbl>  
## 1  1880 F     Mary    7065 0.0724 97605.  
## 2  1880 F     Anna    2604 0.0267 97605.  
## 3  1880 F     Emma    2003 0.0205 97605.  
## 4  1880 F   Elizabeth 1939 0.0199 97605.  
## 5  1880 F    Minnie   1746 0.0179 97605.  
## 6  1880 F   Margaret 1578 0.0162 97605.  
## 7  1880 F      Ida    1472 0.0151 97605.  
## 8  1880 F    Alice    1414 0.0145 97605.  
## 9  1880 F   Bertha   1320 0.0135 97605.  
## 10 1880 F    Sarah    1288 0.0132 97605.  
## # ... with 1,924,655 more rows
```

Transforming variables (mutate)



```
mutate(  
  babynames,  
  year_diff = 2018 - year,  
  months_diff = year_diff * 12  
)
```

```
## # A tibble: 1,924,665 x 7  
##   year sex  name      n  prop year_diff months_diff  
##   <dbl> <chr> <chr>   <int> <dbl>   <dbl>      <dbl>  
## 1  1880 F    Mary    7065 0.0724    138      1656  
## 2  1880 F    Anna    2604 0.0267    138      1656  
## 3  1880 F    Emma    2003 0.0205    138      1656  
## 4  1880 F  Elizabeth  1939 0.0199    138      1656  
## 5  1880 F   Minnie   1746 0.0179    138      1656  
## 6  1880 F  Margaret  1578 0.0162    138      1656  
## 7  1880 F     Ida    1472 0.0151    138      1656  
## 8  1880 F    Alice   1414 0.0145    138      1656  
## 9  1880 F   Bertha   1320 0.0135    138      1656  
## 10 1880 F    Sarah   1288 0.0132    138      1656  
## # ... with 1,924,655 more rows
```

summarise

Reducing variables (summarise)



```
summarise(babynames, max_prop = max(prop))
```

```
## # A tibble: 1 x 1
##   max_prop
##   <dbl>
## 1    0.0815
```

Reducing variables (summarise)



Summary functions

- `min(x)`, `max(x)`, `quantile(x, p)`
- `mean(x)`, `median(x)`,
- `sd(x)`, `var(x)`, `IQR(x)`
- `n()`, `n_distinct(x)`
- `sum(x > 10)`, `mean(x > 10)`

grouped summarise



Reducing variables (summarise)



Grouped summarise

```
by_year <- group_by(babynames, year)
by_year
```

```
## # A tibble: 1,924,665 x 5
## # Groups:   year [138]
##   year sex  name      n  prop
##   <dbl> <chr> <chr>   <int> <dbl>
## 1  1880 F    Mary    7065 0.0724
## 2  1880 F    Anna    2604 0.0267
## 3  1880 F    Emma    2003 0.0205
## 4  1880 F  Elizabeth 1939 0.0199
## 5  1880 F   Minnie   1746 0.0179
## 6  1880 F  Margaret 1578 0.0162
## 7  1880 F    Ida     1472 0.0151
## 8  1880 F   Alice    1414 0.0145
## 9  1880 F  Bertha   1320 0.0135
## 10 1880 F   Sarah    1288 0.0132
## # ... with 1,924,655 more rows
```

Reducing variables (summarise)



Grouped summarise

```
summarise(  
  by_year,  
  max_n = max(n)  
)
```

```
## # A tibble: 138 x 2  
##   year max_n  
##   <dbl> <dbl>  
## 1  1880  9655  
## 2  1881  8769  
## 3  1882  9557  
## 4  1883  8894  
## 5  1884  9388  
## 6  1885  9128  
## 7  1886  9889  
## 8  1887  9888  
## 9  1888 11754  
## 10 1889 11648  
## # ... with 128 more rows
```


Reducing variables (summarise)



Grouped summarise

```
by_year_sex <- group_by(babynames, year, sex)

summarise(
  by_year_sex,
  max_n = max(n)
)
```

```
## # A tibble: 276 x 3
## # Groups:   year [138]
##   year sex    max_n
##   <dbl> <chr> <dbl>
## 1  1880 F      7065
## 2  1880 M      9655
## 3  1881 F      6919
## 4  1881 M      8769
## 5  1882 F      8148
## 6  1882 M      9557
## 7  1883 F      8012
## 8  1883 M      8894
## 9  1884 F      9217
## 10 1884 M      9388
## # ... with 266 more rows
```

pipes



Data pipelines (%>%)



- Often, we want to use several verbs (filter, arrange, group_by, summarise...)
- Multiple operations are difficult to read, or require to create multiple intermediate objects:

```
year_1880 <- summarise(  
  group_by(  
    filter(  
      babynames, year = 1880  
    ),  
    sex  
  ),  
  max = max(n),  
  prop = max(prop)  
)
```

```
year_1880 <- filter(  
  babynames, year = 1880  
)  
year_1880_grouped <- group_by(  
  year_1880, sex  
)  
summarised_year_1880 <- summarise(  
  year_1880_grouped,  
  max = max(n),  
  prop = max(prop)  
)
```

Data pipelines (%>%)



- Alternative (cleaner and easy to read): *pipe* operator (%>%) from `magrittr` package
- The result of the left side is passed to the function in the right as first argument:

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

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

- In the tidyverse %>% makes each function to be applied to the data frame resulting from the previous step

`filter(df, color = 'blue')` is the same as `df %>% filter(color = 'blue')`

`mutate(df, double = 2*value)` is the same as `df %>% mutate(double = 2*value)`

Data pipelines (%>%)



Nested functions

```
year_1880 <- summarise(  
  group_by(  
    filter(  
      babynames, year = 1880  
    ),  
    sex  
  ),  
  max = max(n),  
  prop = max(prop)  
)
```

Data pipelines (%>%)



Nested functions

```
year_1880 <- summarise(  
  group_by(  
    filter(  
      babynames, year = 1880  
    ),  
    sex  
  ),  
  max = max(n),  
  prop = max(prop)  
)
```

Pipeline

```
year_1880 <- babynames %>%  
  filter(year = 1880) %>%  
  group_by(sex) %>%  
  summarise(  
    max = max(n),  
    prop = max(prop)  
  )
```

Applying all together



How do you do to get the names with the maximum proportion for each year and sex? We also want the total n for each year and sex.

Applying all together



How do you do to get the names with the maximum proportion for each year and sex? We also want the total n for each year and sex.

```
babynames %>%
  group_by(year, sex) %>%
  arrange(desc(prop)) %>%
  summarise(
    prop_max = max(prop),
    prop_total = sum(prop),
    name = first(name),
    n_total = sum(n),
    n_year = first(n)
  ) %>%
  select(
    name, sex, year, n_year, n_total, prop_max, prop_total
  ) %>%
  mutate(
    calc_prop = n_year/n_total
  )
```


Thank you!

 @MalditoBarbudo

 <https://github.com/MalditoBarbudo/>

 v.granda@creaf.uab.cat

Presentation repository:

 https://github.com/MalditoBarbudo/2019_dp1yr_girona_aeet

Acknowledgements:

Aitor Ameztegui [@multivac42](#)

University of Lleida