Introduction to the Tidyverse: Data wrangling

R-Ladies Frankfurt Meetup #2

Sandra Pintor

23rd May 2019

Agenda

1. Get to know each other

2. What is R-Ladies?

3. Introduction to the Tidyverse

05:00

Get to know each other

Who am I?

Where am I from?

What do I do?

Experience with R

Hobbies/Funny thing

01:00

What is R-Ladies?

R-Ladies

Worldwide organization that promotes

- **gender diversity** in the R-community
- via **meetups** and **mentorship**
- in a **friendly** and **safe** environment

R-Ladies

- R-Ladies is a user-run group, meaning it is all of us that keep it going.
- Everyone is welcome to present something, no matter their skill level.
- Presentations may be to showcase something you've been working on, to get feed-back on your approach or code, or just because you want to contribute and practise giving talks about coding.
- Everyone has something new to learn, and even a novice may have found an interesting way of doing someting an expert didn't think of.

Code of Conduct

- Public events are **always** for free
- The conceptual domain / scope is **R**-specifc
- Be aware and acknowledge **other**'s contribution
- Use neutral language
- Constructive, supportive and **gender-inclusive** environment to share, learn and network
- Mentoring and organization is exclusively reservaded for women

R-Ladies History

GABRIELA DE QUEIROZ

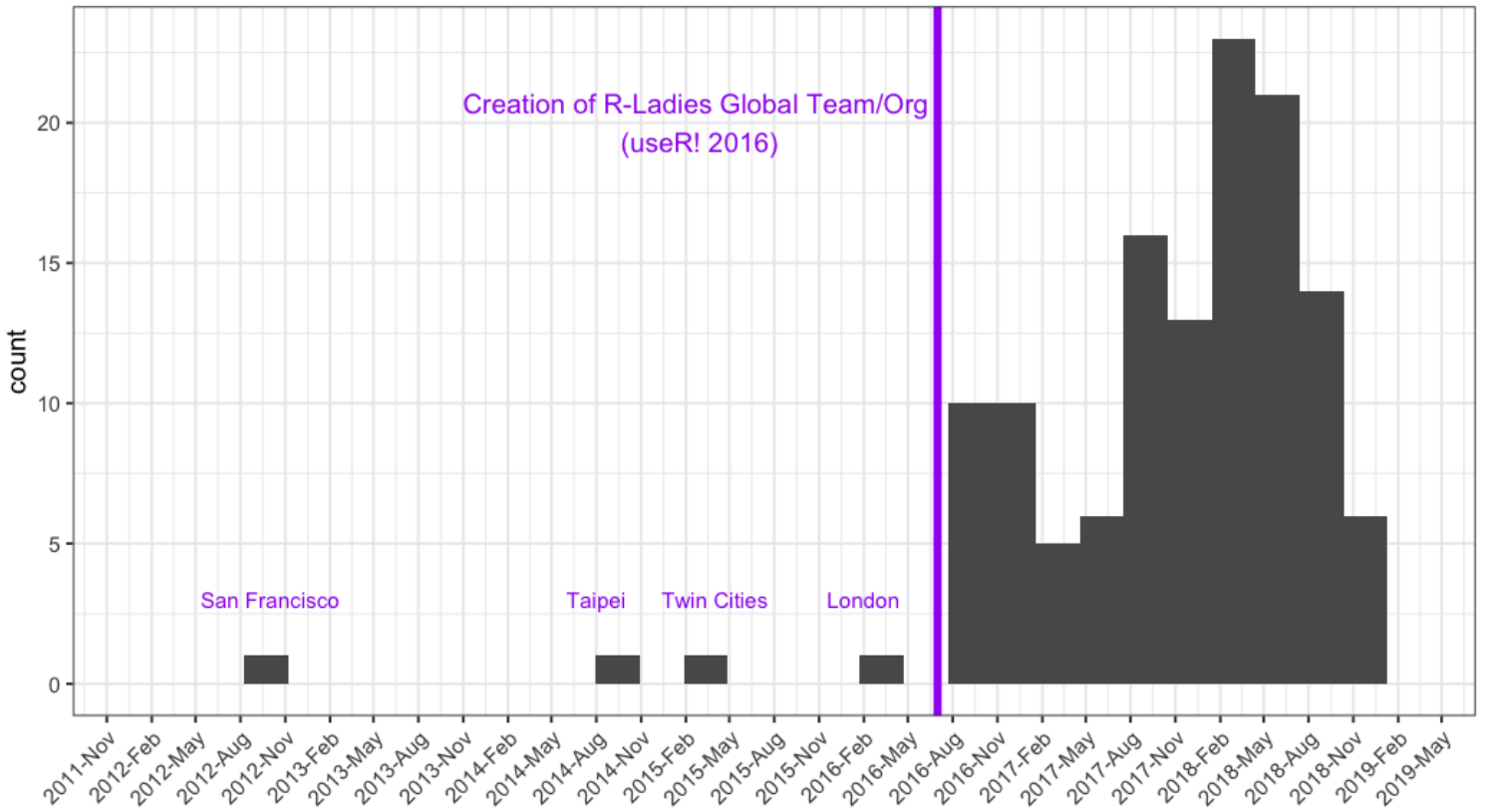
1st of October 2012

Gabriela de Queiroz founded R-Ladies

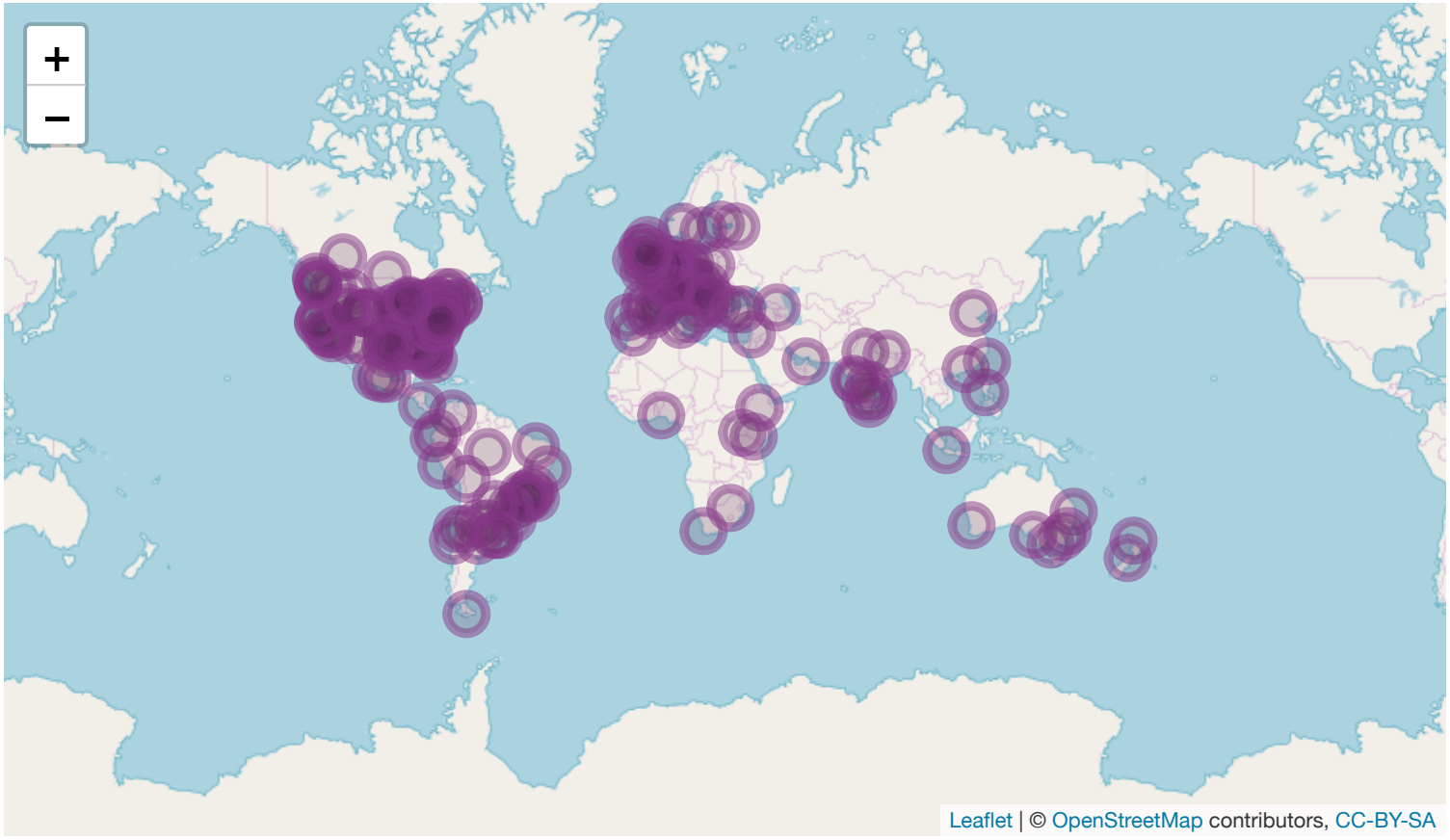
She wanted to give back to the community after going to several meetups and learning a lot for free.

The first meetup took place in San Francisco, California (United States).

R-Ladies Growth



R-Ladies Global



R-Ladies in Germany

R-Ladies in Germany

R-Ladies Frankfurt

Goals R-Ladies Frankfurt

Build an **active**, **supportive**, and **empowering** R community of women to:

- Learn R
- Teach R
- Share experiences
- Improve together
- Network

More women coding, developing and creating R packages and being involved in the R community

To keep in mind...

We all have different knowledge and level of experience 🤓

No matter if we're newbie or proficient R user, there's always something that we don't know! 🔎

No matter if you're newbie or proficient R user, don't be shy to ask, comment! 🙌

We're here to learn together! 💪

How to be engaged?

Participate in meetup

Give a tutorial

Mentor

...

Introduction to the Tidyverse

What are your expectations?

Objectives

1. To generally understand tidyverse and differences between tidyverse and base R
2. To recognize packages and functions to wrangle data in the tidyverse
3. To write code using tidyverse packages

Tidyverse

The tidyverse

"(...) **collection of R** 📦 designed for data science. All packages **share** an underlying **design philosophy**, **grammar**, and **data structures**."
(www.tidyverse.org)

Packages built on and in base R

Tidyverse:

- style guide
- design principles

Principles in the tidyverse

1. Consistent data structures: **data frame**. Usually, tidy data format:
    - Each variable must have its own column
    - Each observation must have its own row
    - Each value must have its own cell
2. Each function should solve one small and well-defined class of problems
3. Rely on function composition to simplify data science workflow by using, as an example, the `magrittr` **pipe** thus enhancing readability and avoiding the need to name interim objects

⚠ Note:

Tidyverse is a quite handy set of tools for solving data science problems and recommended to work with data up to 1-2 Gb. When you work with larger data (10-100 Gb), you should use `data.table` as well as other approaches to the data (Grolemund & Wickham, 2017).

Install tidyverse 


```r
# Install all tidyverse packages
install.packages("tidyverse")
```

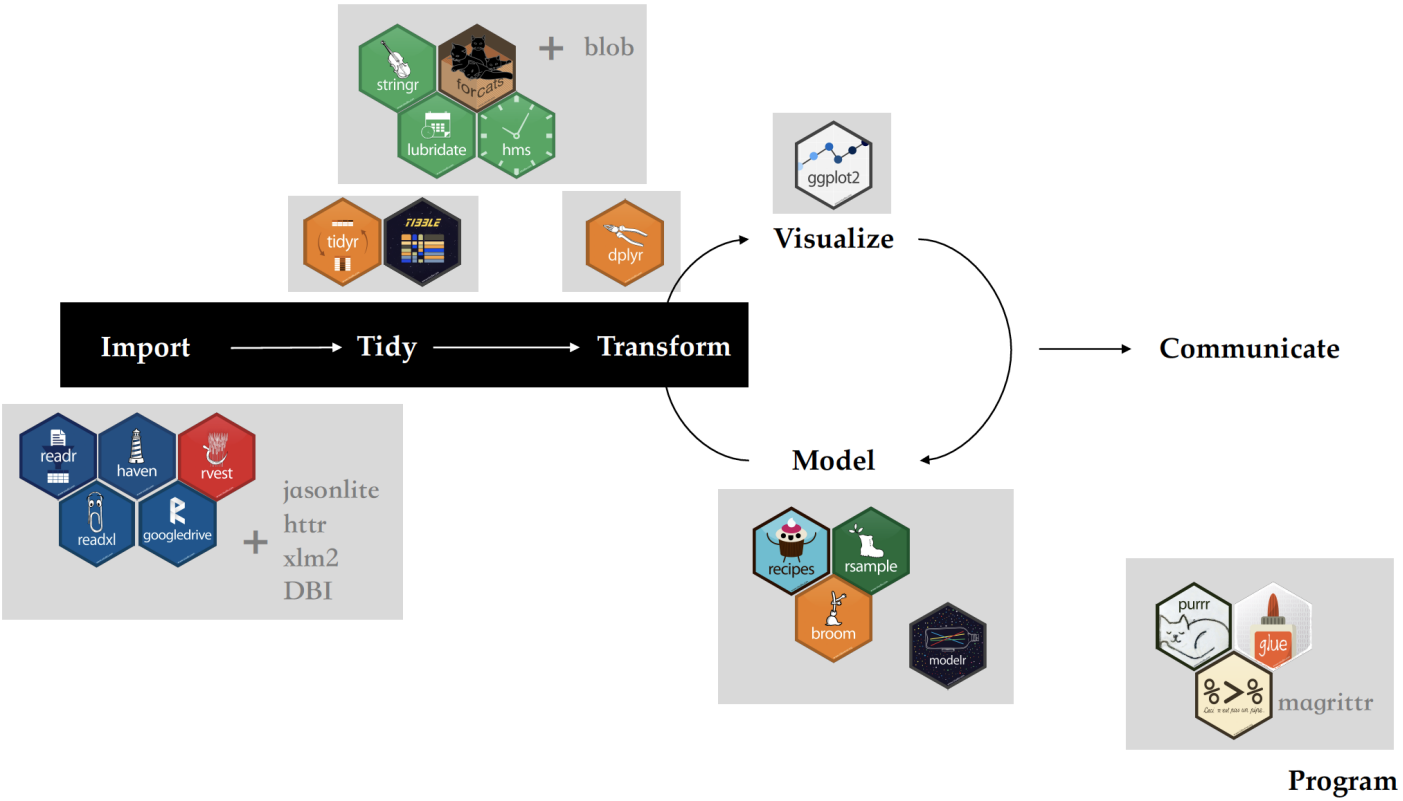
```r
# Equivalent of
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("hms")
install.packages("stringr")
install.packages("lubridate")
install.packages("forcats")
install.packages("DBI")
install.packages("haven")
install.packages("httr")
install.packages("jsonlite")
install.packages("readxl")
install.packages("rvest")
install.packages("xml2")
install.packages("modelr")
install.packages("broom")
```

---

source: Github tidyverse on 22 May 2019

Data science project with tidyverse

"The tidyverse is a **language** for solving **data science** challenges with **R code**."
(Hadley Wickham)



Load tidyverse 📦


```
# Load core packages
library(tidyverse)
```

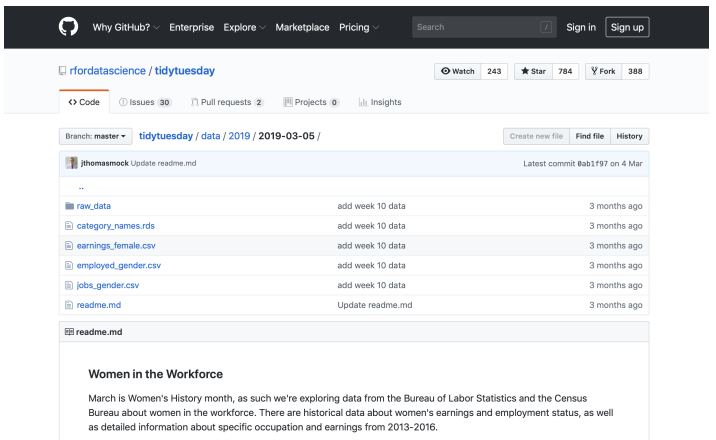
```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──

## ✔ ggplot2 3.1.1     ✔ purrr   0.3.2
## ✔ tibble  2.1.1     ✔ dplyr   0.8.1
## ✔ tidyr   0.8.3     ✔ stringr 1.4.0
## ✔ readr   1.3.1     ✔ forcats 0.4.0

## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

Data

Datasets

Source:
Github of TidyTuesday: A weekly data project in R from the R4DS online learning community

Datasets #10: 2019-03-05
Data Women in the Workforce



The data format is **csv**

Question?

? Between 2013 and 2016, were the earnings of women and men similar across occupations for full-time workers?

Import data

Import data with readr

- Loads flat files in R
- Imports rectangular data frames (columns are variables and rows are observations)
- `readr::read_csv()` is faster than `read.csv()`

```
# Import data from github
jobsGender <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytues

employedGender <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidy
```

Import data with readr


```
# Import data from github
jobsGender <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytues

# Print data
jobsGender
```

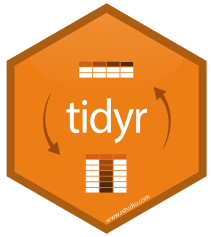
```
## # A tibble: 2,088 x 12
##     year occupation major_category minor_category total_workers
##    <dbl> <chr>      <chr>          <chr>                  <dbl>
##  1  2013 Chief exe… Management, B… Management           1024259
##  2  2013 General a… Management, B… Management            977284
##  3  2013 Legislato… Management, B… Management             14815
##  4  2013 Advertisi… Management, B… Management             43015
##  5  2013 Marketing… Management, B… Management            754514
##  6  2013 Public re… Management, B… Management             44198
##  7  2013 Administr… Management, B… Management            109703
##  8  2013 Computer … Management, B… Management            489048
##  9  2013 Financial… Management, B… Management            990611
## 10  2013 Compensat… Management, B… Management             14656
## # … with 2,078 more rows, and 7 more variables: workers_male <dbl>,
## #   workers_female <dbl>, percent_female <dbl>, total_earnings <dbl>,
## #   total_earnings_male <dbl>, total_earnings_female <dbl>,
## #   wage_percent_of_male <dbl>
```

Write to a file with readr


```
# Save data locally
write_csv(jobsGender, path = "./data/jobsGender.csv")
write_csv(employedGender, path = "./data/employedGender.csv")
```

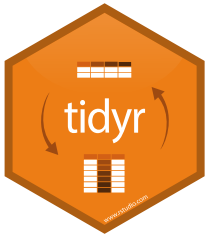
Tidy data

Reshape data with tidyr



- Creates a tidy dataset by reshaping the layout of tabular data
- Usually solves the following **problems**:
  - One variable spread across multiple columns
  - One observation scattered across multiple rows
- Important **verbs**:
  - `gather()` - gathers multiple columns into a new one, result "wide" cata becomes longer
  - `spread()` - spreads rows into multiple columns, result "long" data becomes wider
  - `separate()` - separates a column into multiple columns
  - `unite()` - unites multiple columns into one

Reshape data with tidyr




```
# Gather columns into two new columns
jobsTidy <- gather(jobsGender,
                   key = "workerGender",
                   value = "earnings",
                   c(workers_male, workers_female))

dplyr::glimpse(jobsTidy)

# Inspect values of variable workerGender
unique(jobsTidy$workerGender)
```

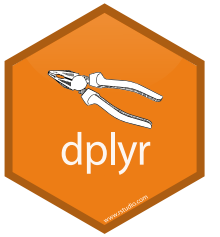
```
# Separate column workerGender into three new columns
jobsTidySep <- separate(jobsTidy,
                        col = workerGender,
                        into = c("totalEarnings", "earningsTotal", "gender"),
                        sep = "_",
                        remove = TRUE)

dplyr::glimpse(jobsTidySep)
```

It's your turn...

Do other variables in this dataset require such procedure?
Or other `tidyr` verb?

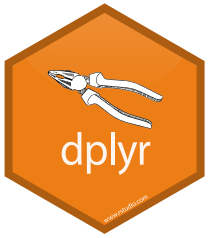
Transform data

Manipulate data with dplyr



- Grammar of data manipulation
- Transforms tabular data
- Basic single-table **verbs** for data manipulation:
  - `mutate()` - creats new variables that are functions of existing variables
  - `select()` - extracts variables
  - `filter()` - extract rows that meet logical criteria
  - `summarise()` / `summarize()` - calculate aggregate measures for groups
  - `arrange()` - reorder the rows
  - `group_by()` - group by one or more variables
- Some two-table **verbs** for data manipulation:
  - `inner_join()`
  - `left_join()`
  - `right_join()`
  - `full_join()`
  - `semi-join()`
  - `anti_join()`

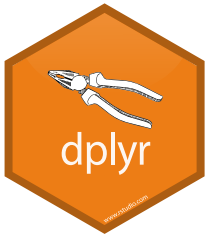
Manipulate data with dplyr




```
# Join two datasets
joinJobs <- inner_join(jobsTidySep,
                       employedGender,
                       by = "year")

dplyr::glimpse(joinJobs)
```

```
# Select variables of interest
joinJobsShort <- select(joinJobs, c(year:minor_category, earnings, gender, full_ti


dplyr::glimpse(joinJobsShort)
```

Manipulate data with dplyr




```
# Filter women
jobsFilter <- filter(joinJobsShort, gender == "female")
```

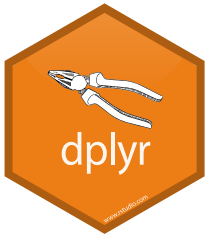
Logical operators

- == equal
- > greater than
- < less than
- >= greater than or equal
- <= less yhan or equal
- != not equal

Boolean operators

- & and
- | or
- ! not

Manipulate data with dplyr




```
# Summarise
summarise(jobsFilter, earnMean = mean(earnings, na.rm = TRUE))
```

Summary functions

- `min()`
- `max()`
- `mean()`
- `median()`
- `quantile()`
- `sd()`
- `var()`
- `IQR()`

The pipe operator: %>%

- `magrittr` package
- loads automatically
- code more readable when successive commands are required

```
function(object, arguments)
```

with pipe

```
object %>% function(arguments)
```

pipe can be read as **"then"**

Piped code


```
# Code with pipe
pipeSummary <- joinJobs %>%
  select(c(year:minor_category, earnings, gender, full_time_female, full_time_male
  filter(gender == "female") %>%
  group_by(occupation, major_category, minor_category) %>%
  summarise(earningsMean = mean(earnings, na.rm = TRUE)) %>%
  arrange(desc(earningsMean))
```

Bring it all together

It's your turn...

Question?

? Between 2013 and 2016, were the earnings of women and men similar across occupations for full-time workers?

15:00

Additional Resources

Books and Papers

- Grolemund, G., & Wickham, H. (2017). *R for Data Science.*
- Grolemund, G. (2019). *The Tidyverse Cookbook.*
- Ross, Z., Wickham, H., & Robinson, D. (2017). Declutter your R workflow with tidy tools. *PeerJ Preprints*.
- Wickham, H. (2014). Tidy data. *Journal of Statistical Software*, *59*(10), 1-23.

Other

- RStudio cheat sheets
- tidyverse website

Save the date!

19th June

18:00-21:00

Data visualization with R

📍 Frankfurt School of Finance and Management

Keep in touch!

frankfurt@rladies.org
https://www.meetup.com/rladies-frankfurt/
@RLadiesFRA

Slides created via the R package **xaringan** by Yihui Xie with the Rladies theme by Alison Hill.

Thank you!