```
library(tidyverse)

rladies_stockholm %>%
  filter(city == 'Stockholm')
```

@ FOO CAFÉ

foocafe.org

Welcome to R-Ladies Stockholm!

Today's timeline:

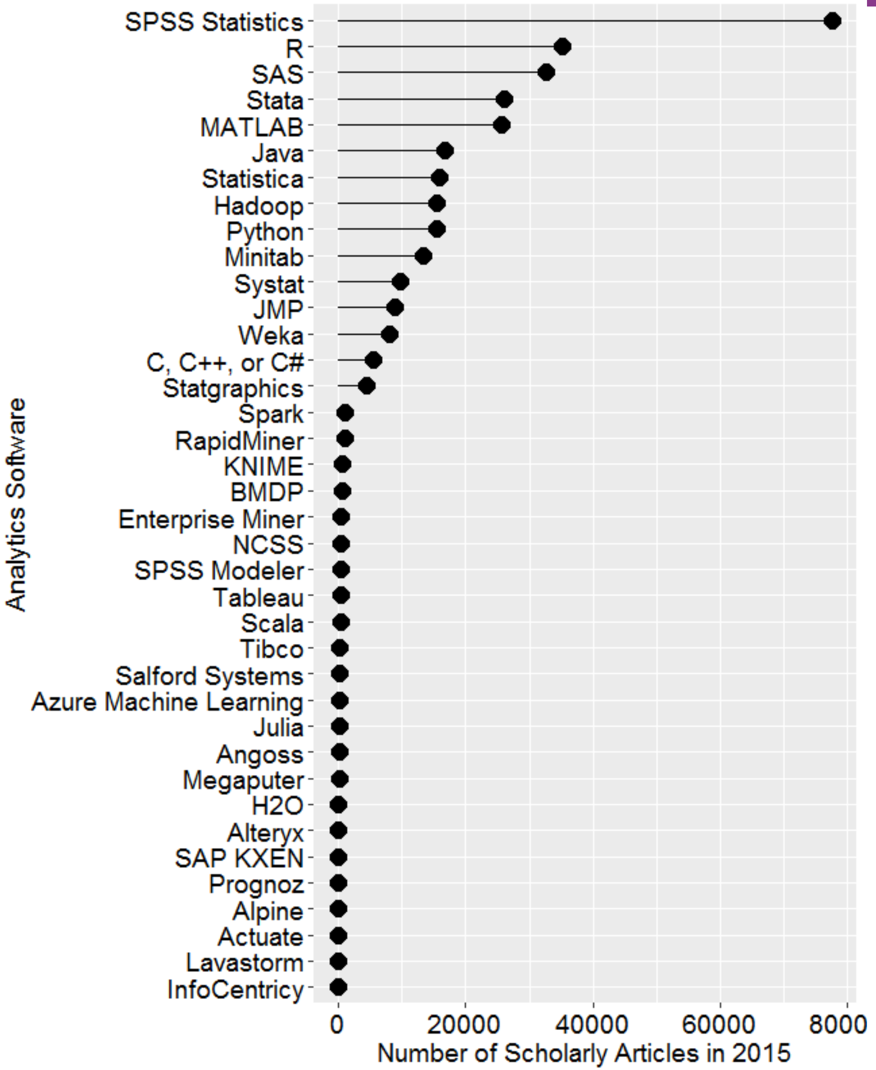
18.00	Introduction to the R-ladies
18.15	Dirty data
18.30	The universe of Tidyverse
18.50	Break and food 30 min
19.20	Focus: TidyR and Dplyr
19.45	Topics for future meetups
20.00	End

Who are the Stockholm R-Ladies?

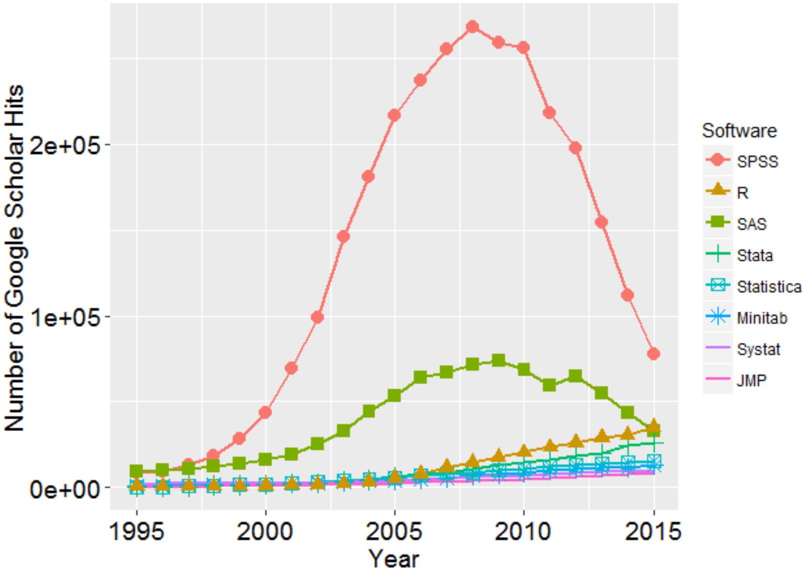
- Elisabeth Dahlqwist – PhD in Biostatistics
- Nina Lindell - Data Analyst
- Stefanie Möllberg - Data Analyst
- Yulia Leontyeva - Biostatistician
- Sophie Debonneville - Bioinformatician
- Maya Illipse - Post doc in image and signal processing
- Ines Illipse - Master student in statistical learning and data analysis
- Aminata Ndiaye - Biostatistician

Why R and the R community?

- R is an open source software - free
- R is rapidly growing and a competence often asked for
- Lots of online material and an active R-community makes it easy to get advanced
- R is often first with providing their users with new methods



http://r4stats.com/2016/06/08/r-passes-sas-in-scholarly-use-finally/



The number of scholarly articles found in each year by Google Scholar. Only the top six "classic" statistics packages are shown.

The R consortium and R-ladies

Call for proposals!

R consortium

Support to the R Foundation

satRdays are free/cheap accessible R conferences being organised by the community for the community.

R-hub

a centralised tool for checking R packages.

R-Ladies

R Forwards

R Foundation taskforce on women and other under-represented groups

The User Group Program all over the world. Ex **Stockholm R User Group (SRUG)**

Why R-Ladies?

- Founded in San Francisco 2012 by **Gabriela de Queiroz** https://rladies.org/

- **Mission:** *"to achieve proportionate representation by encouraging, inspiring, and empowering people of genders currently underrepresented in the R community"*
  History of R: https://rss.onlinelibrary.wiley.com/doi/10.1111/j.1740-9713.2018.01169.x

- **Important question 1:** Is diversity in the R community important?
- **Important question 2:** Is R-ladies the way to do it?

R-ladies Global at UseR2017

R-ladies - steady growing!

https://gqueiroz.shinyapps.io/rshinylady/

R-ladies Stockholm

- Welcome all proficiency levels - will host meetups for new and senior users.
- Peer-education → we educate each other
- Share our R-experience in a relaxed and friendly environment
- Contribute to the R-community

Today's topic: Cleaning dirty data in R

Cleaning data

- A problem shared by most - the dark side of working with data
- Regarded as trivial but this is not very true
- Problems and tools - Intro to Tidyverse
- Focus on Dplyr and TidyR to clean data

What is dirty data?

> “Tidy datasets are all alike, but every messy dataset is messy in its own way.” — Hadley Wickham

- Also, depends on how you want your data to be!
- In general:
    - Missing
    - Misleading coding
    - Wrongly imputed - combinations of variables that don’t make sense

How do we get dirty data?

- **In the data collection stage:**
  - coding of variables
  - database imputation errors
  - missing values, duplicates

Cleaning data

- The complexity of dirty data - missing data, imputation and measurement error are areas of research in itself

- The same dataset can be cleaned different by different people - the importance of documenting the cleaning process!

- Follow guidelines of best practice

Four all these aspect the Tidyverse package aims to play a role, more about that later!

Basic checks

- Plot variables - do they look as you would expect?
- Cross-tabulation of variables that should give similar answers
- Systematic pattern in the missingness - data collection as cause for systematic missingness or error
- Use the code book and get a basic understanding of the data collection
- If you get cleaned data - ask for the documentation of the data cleaning …. and check it

Subjectivity in data cleaning

Problem: Same individual answer Question 1 over three rounds of visits… but the answers are not consistent...

ID	Q1.0.1	Q1.1.1	Q1.2.1
1	1	1	1
2	1	2	2
3	2	3	NA
4	NA	1	2

- **We want to know THE answer to Q1!**
- **Median over the individuals?**
- **The last and most “updated” value?**
- **The first value?**

The answer depends on that variable! Imagine you have 50-100 variables to do this for……

We need tools for data wrangling!

The Tidyverse gives you:

- Code written for humans to read

- Same syntax and operators over different packages - not always the case in open source

- This ease the documentation of your workflow with the data

- The core tidy data principles:
    1. Variable make up the columns
    2. Observations make up the rows
    3. Values go into cells

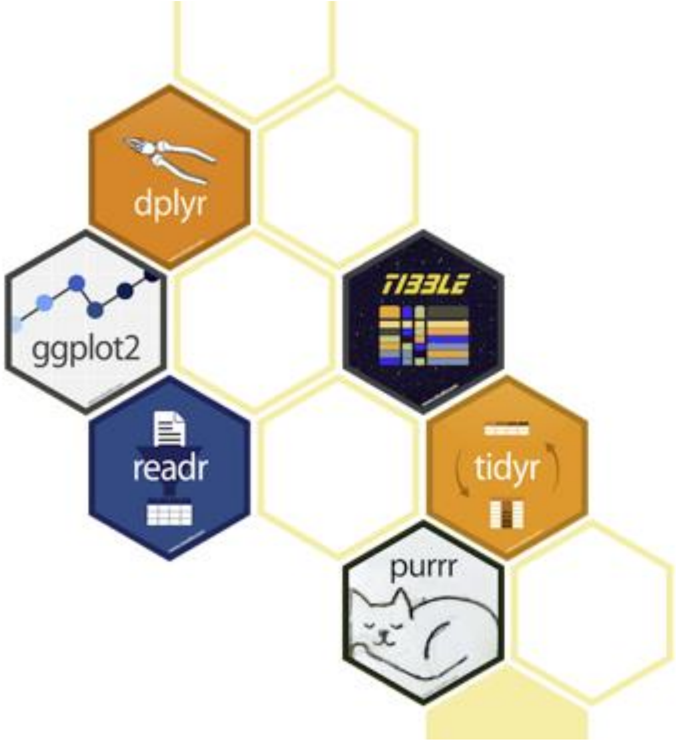
Intro to the Tidyverse by Sophie Debonneville

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

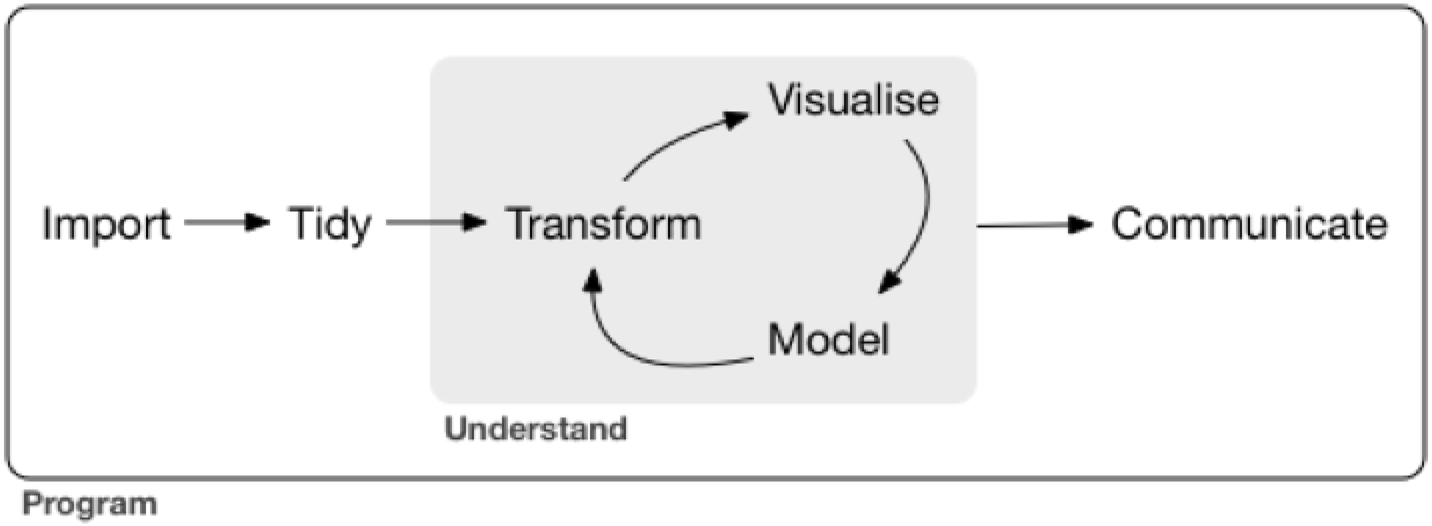
-- Attaching packages --- tidyverse 1.2.1 --
v ggplot2 2.2.1   v purrr   0.2.4
v tibble 1.4.2   v dplyr   0.7.4
v tidyr   0.8.0   v stringr 1.2.0
v readr   1.1.1   v forcats 0.3.0

The Tidyverse

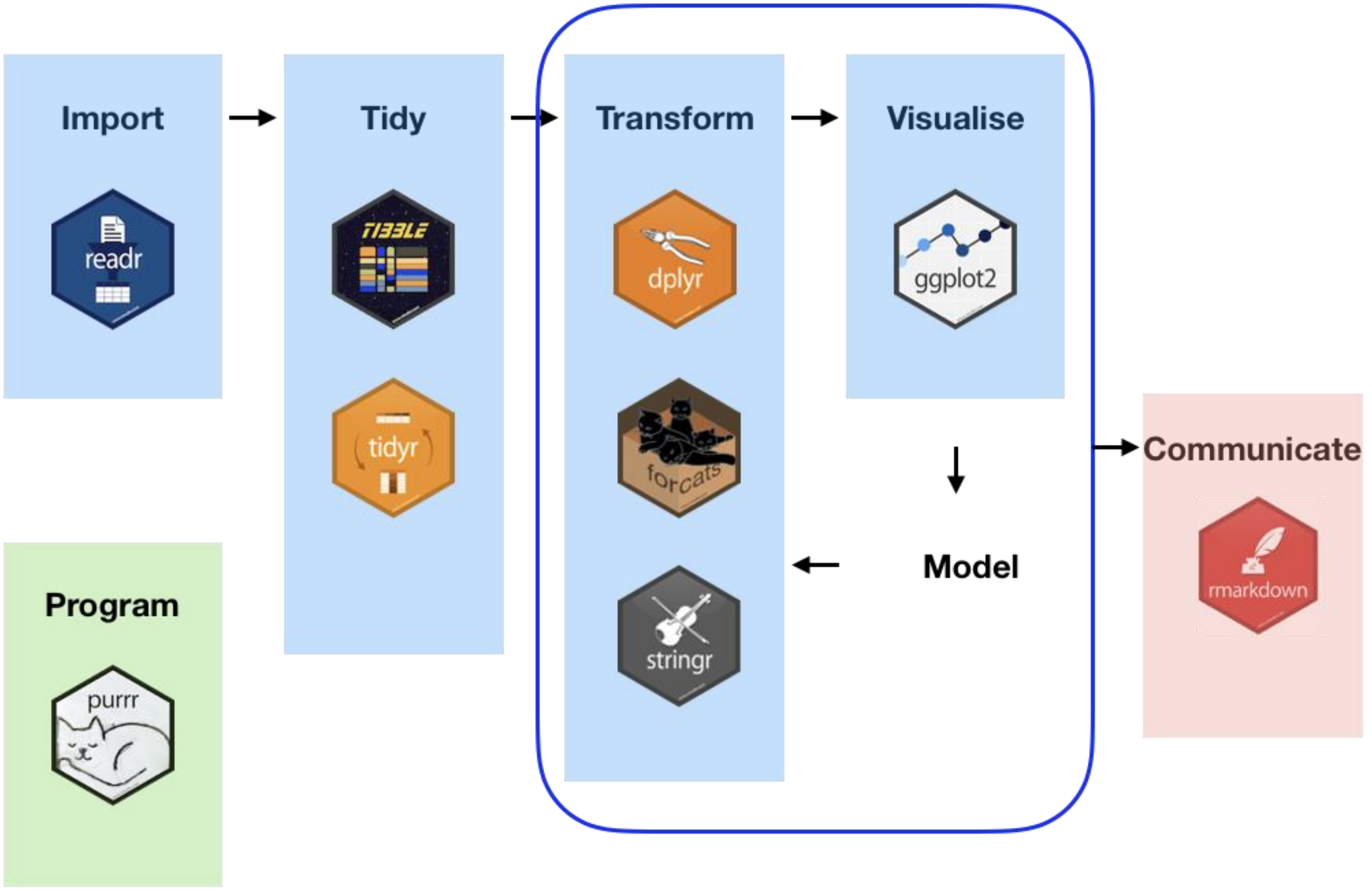
- Collection of R packages for data science
- Developed by Hadley Wickham and others from the Rstudio team



Workflow in data science



http://r4ds.had.co.nz/


Import
readr
Tidy
TIBBLE
tidyr
Transform
dplyr
forcats
stringr
Visualise
ggplot2
Model
Communicate
rmarkdown
Program
purrr

La Trahison des images (1928-1928, René Magritte)

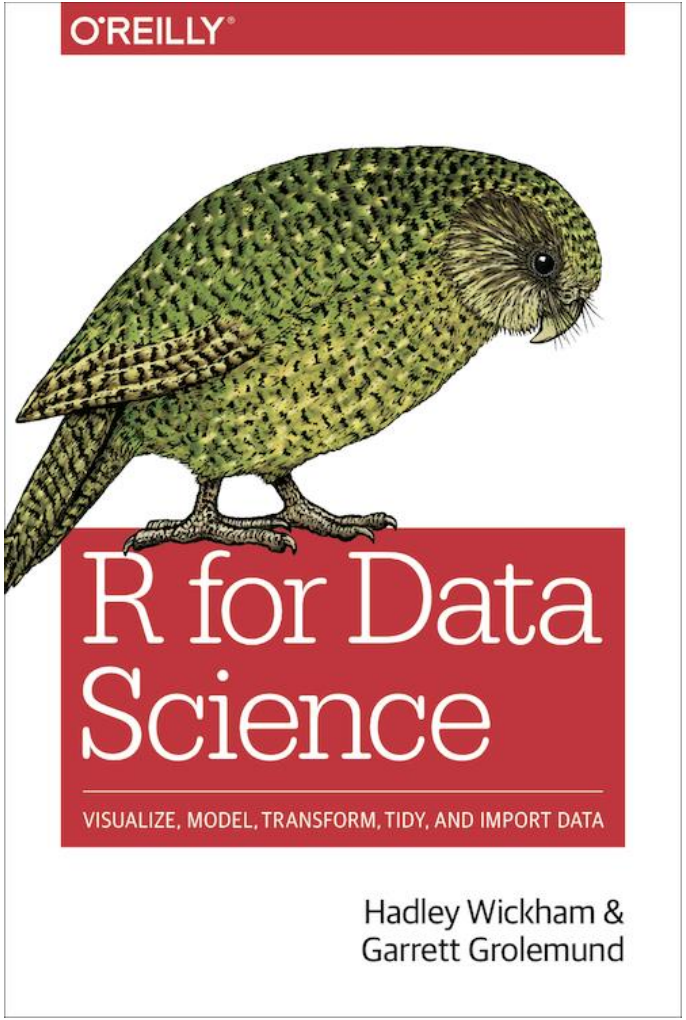
A basic building block: The pipe operator %>%

- comes from magrittr package
- replace `f(x)` with `x %>% f()`

```
summary(mtcars)   mtcars %>% summary()
```

```
function1(function2(function3(data)))

data %>% function1() %>% function2() %>% function3()
```



Online text: http://r4ds.had.co.nz/

Online tutorials:
https://www.datacamp.com/community/tutorials/tidyverse-tutorial-r

http://www.storybench.org/getting-started-with-tidyverse-in-r/

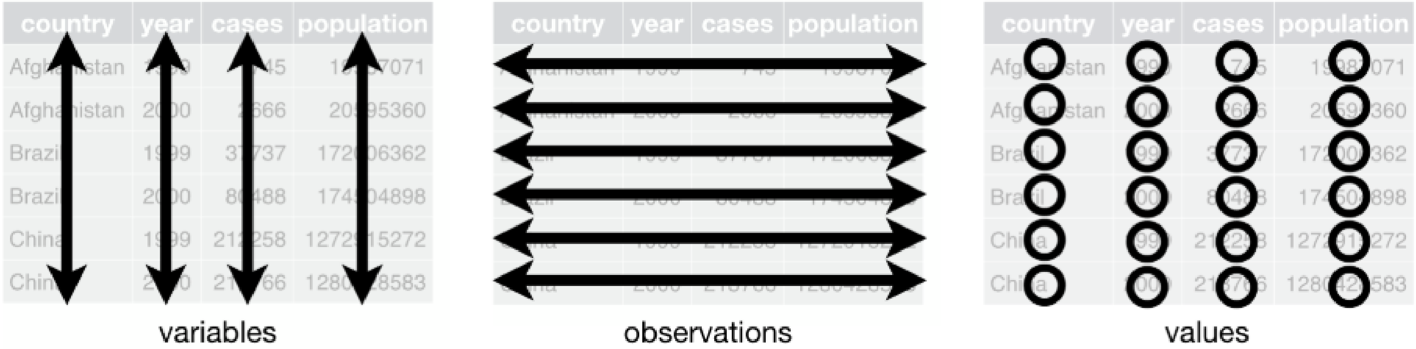
Intro and tutorial of tidyr and dplyr by Stefanie Möllberg

tidyr + dplyr = Clean data!

What defines a tidy data set?

There are three interrelated rules which make a dataset tidy:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.



What defines a tidy data set?

```
table1
#> # A tibble: 6 x 4
#>   country      year  cases population
#>   <chr>       <int>  <int>      <int>
#> 1 Afghanistan  1999    745   19987071
#> 2 Afghanistan  2000   2666   20595360
#> 3 Brazil       1999  37737  172006362
#> 4 Brazil       2000  80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```

```
table2
#> # A tibble: 12 x 4
#>   country      year type            count
#>   <chr>       <int> <chr>           <int>
#> 1 Afghanistan  1999 cases             745
#> 2 Afghanistan  1999 population   19987071
#> 3 Afghanistan  2000 cases            2666
#> 4 Afghanistan  2000 population   20595360
#> 5 Brazil       1999 cases           37737
#> 6 Brazil       1999 population 172006362
#> # ... with 6 more rows
```

```
table3
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/20595360
#> 3 Brazil       1999 37737/172006362
#> 4 Brazil       2000 80488/174504898
#> 5 China        1999 212258/1272915272
#> 6 China        2000 213766/1280428583
```

How can you make an untidy data set become tidy?

TidyR:

gather() & spread()

separate() & unite()

nest & unnest()

Dplyr:

filter()

select()

mutate()

arrange()

group_by() & summarize()

join()

tidyr

www.rstudio.com

TidyR: gather & spread

gather(): gather variables into one column

Valuable when you need your data on either long or wide format
Valuable when you have values as column headers

What's needed:

key = name of new column

value = name of the new value column

which columns that should be gathered

table4a %>%

gather(key = “year”, value = “cases”, 1999:2000)

```
table4a
#> # A tibble: 3 x 3
#>   country     `1999` `2000`
#> * <chr>        <int>  <int>
#> 1 Afghanistan    745   2666
#> 2 Brazil       37737  80488
#> 3 China       212258 213766
```

TidyR: gather & spread

spread(): spread values into variables (columns)

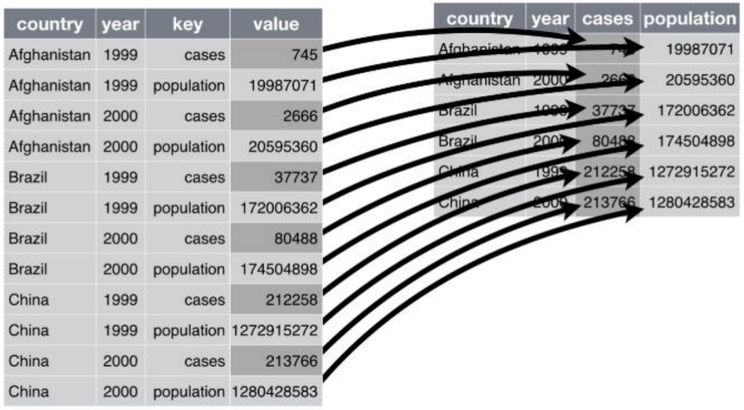
What's needed:

key = name of variable that contains values to be spread

value = the column that contains the values to spread

```
table4a %>%

spread(key = “key”, value = “value”)
```



TidyR: separate & unite

separate(): splitting values in a cell, into more than one cell

Valuable when you have more than one value in the same cell or when you want to split values to make analyses

What's needed:

col = column that contains value that needs to be separated

into = name of new variables created when separating

sep = pattern to separate with

```
table %>%
separate(col = rate,
into = c("cases", "population"), sep = "/")
```

```
table3
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/20595360
#> 3 Brazil       1999 37737/172006362
#> 4 Brazil       2000 80488/174504898
#> 5 China        1999 212258/1272915272
#> 6 China        2000 213766/1280428583
```

country	year	rate
Afghanistan	1999	**745** / 19987071
Afghanistan	2000	**2666** / 20595360
Brazil	1999	**37737** / 172006362
Brazil	2000	**80488** / 174504898
China	1999	**212258** / 1272915272
China	2000	**213766** / 1280428583

table3

country	year	cases	population
Afghanistan	1999	**745**	19987071
Afghanistan	2000	**2666**	20595360
Brazil	1999	**37737**	172006362
Brazil	2000	**80488**	174504898
China	1999	**212258**	1272915272
China	2000	**213766**	1280428583

TidyR: separate & unite

unite(): adding values together in one cell

What's needed:

col = name of new variable

... = name of variables to unite

sep = separator to use between values

```
table3 %>%
unite(col = year, century, year, sep = “”)
```

country	year	rate
Afghanistan	1999	745 / 19987071
Afghanistan	2000	2666 / 20595360
Brazil	1999	37737 / 172006362
Brazil	2000	80488 / 174504898
China	1999	212258 / 1272915272
China	2000	213766 / 1280428583

country	century	year	rate
Afghanistan	19	99	745 / 19987071
Afghanistan	20	0	2666 / 20595360
Brazil	19	99	37737 / 172006362
Brazil	20	0	80488 / 174504898
China	19	99	212258 / 1272915272
China	20	0	213766 / 1280428583

table6

TidyR: nest & unnest

nest(): creates a list of data frames containing the nested variables (collapses the rows)
unnest(): in a list, making each element it's own row

What's needed:

... = specification of columns to nest / unnest

```
iris %>%
nest(Sepal.Length, Sepal.Width,
     Petal.Length, Petal.Width)

iris %>%
unnest(test, data)
```

	Species	data
1	setosa	list(Sepal.Length = c(5.1, 4.9, 4.7, 4.6, 5, 5.4, 4.6, 5, 4.4, 4....
2	versicolor	list(Sepal.Length = c(7, 6.4, 6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, ...
3	virginica	list(Sepal.Length = c(6.3, 5.8, 7.1, 6.3, 6.5, 7.6, 4.9, 7.3, 6.7...

dplyr

www.rstudio.com

Dplyr: filter

filter(): subsets data based on conditions

```
1: table1 %>%
   filter(country == “Afghanistan”)

2: table1 %>%
   filter(country == “Afghanistan”
                    & year == 1999)

3: table1 %>%
   filter(year >= 2000)
```

		country	year	cases	population
#>		<chr>	<int>	<int>	<int>
#>	2	Afghanistan	2000	2666	20595360
#>	4	Brazil	2000	80488	174504898
#>	6	China	2000	213766	1280428583

Dplyr: select

select(): keeps variables mentioned and removes other

1: table1 %>%
select(country, year)

2: table1 %>%
select(-population)

3: table1 %>%
select(contains(“c”))

```
#>   country       cases
#>   <chr>         <int>
#> 1 Afghanistan     745
#> 2 Afghanistan    2666
#> 3 Brazil        37737
#> 4 Brazil        80488
#> 5 China        212258
#> 6 China        213766
```

Dplyr: mutate

mutate(): adds new variables to your data set

1: Creating a new variable:

```
table1 %>%
mutate(population2 = population * 2)
```

2: Overwriting existing variable

```
table1 %>%
mutate(population = population * 2)
```

```
table1
#> # A tibble: 6 x 4
#>   country      year  cases population
#>   <chr>       <int>  <int>      <int>
#> 1 Afghanistan  1999    745   19987071
#> 2 Afghanistan  2000   2666   20595360
#> 3 Brazil       1999  37737  172006362
#> 4 Brazil       2000  80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```

Dplyr: arrange

arrange(): sorts values in ascending or descending order

```
1: table1 %>%
   arrange(population)


2: table1 %>%
   arrange(desc(population))
```

```
table1
#> # A tibble: 6 x 4
#>   country      year  cases population
#>   <chr>       <int>  <int>      <int>
#> 1 Afghanistan  1999    745   19987071
#> 2 Afghanistan  2000   2666   20595360
#> 3 Brazil       1999  37737  172006362
#> 4 Brazil       2000  80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```

Dplyr: group_by & summarize

Group_by (): takes an existing tbl and converts it into a grouped tbl where operations are performed "by group“

Summarize(): produces summary statistics. When used in combination with group_by() it creates summary statistics on the groups

```
# A summary applied to ungrouped tbl returns a single row
mtcars %>%
  summarise(mean = mean(disp), n = n())
#>       mean  n
#> 1 230.7219 32

# Usually, you'll want to group first
mtcars %>%
  group_by(cyl) %>%
  summarise(mean = mean(disp), n = n())
#> # A tibble: 3 x 3
#>     cyl  mean     n
#>   <dbl> <dbl> <int>
#> 1     4  105.    11
#> 2     6  183.     7
#> 3     8  353.    14
```

Dplyr: join

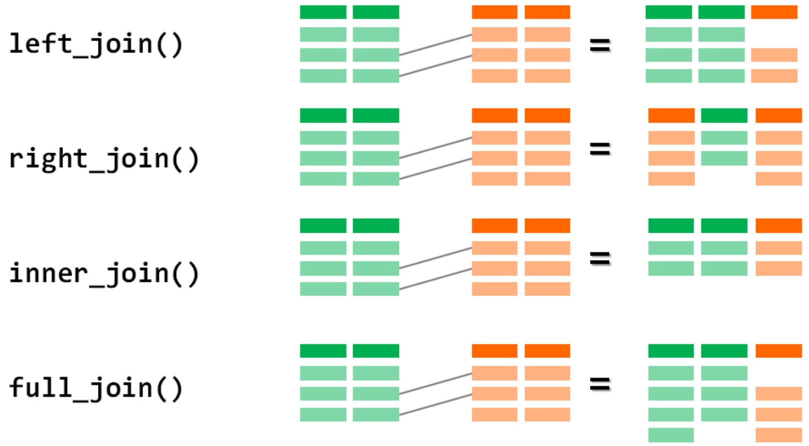
join(): lets you join data sets based on key

left_join()

right_join()

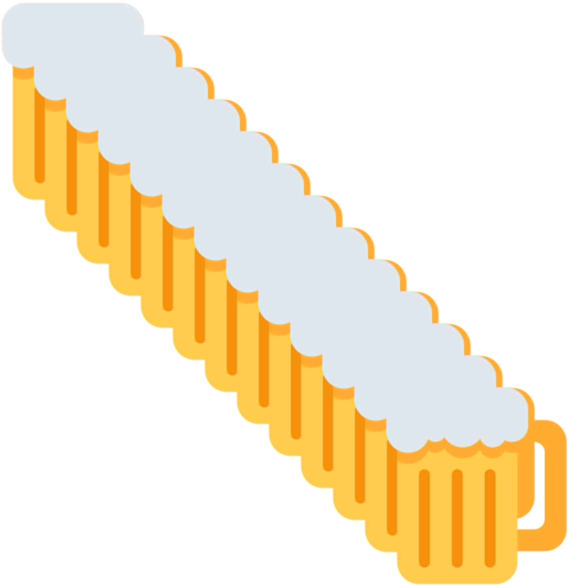
inner_join()

full_join()



Putting the functions to use!

Let's analyze some beers!



Our data sets

Our two interconnected data sets:

1. A data set with 2 410 types of beers
2. A data set with breweries 558 breweries

	count	abv	ibu	id	name	style		
1	107	0.056	4	1350	Summer Solstice	Cream Ale		
2	113	0.056	4	753	Summer Solstice Cerveza Crema (2009)	Cream Ale		
3	118	0.056	4	77	Summer Solstice (2011)	Cream Ale		
4	172	0.092	5	704	Devils Tramping Ground Tripel	Tripel		
5	874	0.044	5	2520	Yo Soy Un Berliner	Berliner Weissbier		
6	962	0.050	5	1604	Chickawawa Lemonale	Fruit / Vegetable Beer		
7	1683	0.044	5	2370	18th Anniversary Gose	Gose	128	12
8	2340	0.040	5	1312	Westbrook Gose	Gose	384	12
9	115	0.069	6	523	Winter Solstice	Winter Warmer	171	12
10	992	0.045	6	2375	Mr. Blue Sky	American Pale Wheat Ale	124	16
11	1461	0.034	6	1417	Weiss Trash Culture	Berliner Weissbier	410	12
12	363	0.053	7	1144	Samuel Adams Summer Ale	American Pale Wheat Ale	300	12
13	1509	0.051	7	128	O'Fallon Wheach	Fruit / Vegetable Beer	442	12
14	1861	0.032	7	2266	Rad	Fruit / Vegetable Beer	46	16
15	2006	0.052	7	1225	Point Nude Beach Summer Wheat	American Pale Wheat Ale	131	16
16	2007	0.050	7	816	Point Nude Beach Summer Wheat	American Pale Wheat Ale	131	12
17	2008	0.050	7	772	Point Nude Beach Summer Wheat (2011)	American Pale Wheat Ale	131	12
18	2013	0.050	7	141	Point Nude Beach Summer Wheat (2010)	American Pale Wheat Ale	131	12
19	154	0.045	8	534	Dirty Blonde Ale	American Blonde Ale	72	12

Showing 1 to 20 of 2,410 entries

	count	name	city		
358	1	NorthGate Brewing	Minneapolis		
12	2	Against the Grain Brewery	Louisville		
268	3	Jack's Abby Craft Lagers	Framingham		
322	4	Mike Hess Brewing Company	San Diego		
203	5	Fort Point Beer Company	San Francisco		
138	6	COAST Brewing Company	Charleston	SC	
229	7	Great Divide Brewing Company	Denver	CO	
484	8	Tapistry Brewing	Bridgman	MI	7
59	9	Big Lake Brewing	Holland	MI	8
498	10	The Mitten Brewing Company	Grand Rapids	MI	9
97	11	Brewery Vivant	Grand Rapids	MI	10
386	12	Petoskey Brewing	Petoskey	MI	11
71	13	Blackrocks Brewery	Marquette	MI	12
384	14	Perrin Brewing Company	Comstock Park	MI	13
551	15	Witch's Hat Brewing Company	South Lyon	MI	14
204	16	Founders Brewing Company	Grand Rapids	MI	15
194	17	Flat 12 Bierwerks	Indianapolis	IN	16
507	18	Tin Man Brewing Company	Evansville	IN	17
67	19	Black Acre Brewing Co.	Indianapolis	IN	18

Showing 1 to 20 of 558 entries

Using TidyR to create a longer format

- Would like to change to a long format

```
breweries_longformat <- breweries %>%
  select(-count, -id) %>%
  gather(key = region, value = cases, -name)
```

	name	region	cases
1	10 Barrel Brewing Company	city	Bend
2	10 Barrel Brewing Company	state	OR
3	18th Street Brewery	city	Gary
4	18th Street Brewery	state	IN
5	2 Towns Ciderhouse	city	Corvallis
6	2 Towns Ciderhouse	state	OR
7	21st Amendment Brewery	city	San Francisco
8	21st Amendment Brewery	state	CA
9	3 Daughters Brewing	city	St Petersburg
10	3 Daughters Brewing	state	FL
11	4 Hands Brewing Company	city	Saint Louis
12	4 Hands Brewing Company	state	MO
13	450 North Brewing Company	city	Columbus
14	450 North Brewing Company	state	IN
15	7 Seas Brewing Company	city	Gig Harbor
16	7 Seas Brewing Company	state	WA
17	7venth Sun	city	Dunedin
18	7venth Sun	state	FL
19	Abita Brewing Company	city	Abita Springs

Showing 1 to 20 of 1,116 entries

Using TidyR to unite variables

- Would like to add the city and state to the brewery name
- Can achieve using unite

```
beers_united_names <- breweries %>%
  unite(col = 'name_city_state', name, city, state,
        sep = ', ')
```

	count	name_city_state	id
1	1	NorthGate Brewing, Minneapolis, MN	0
2	2	Against the Grain Brewery, Louisville, KY	1
3	3	Jack's Abby Craft Lagers, Framingham, MA	2
4	4	Mike Hess Brewing Company, San Diego, CA	3
5	5	Fort Point Beer Company, San Francisco, CA	4
6	6	COAST Brewing Company, Charleston, SC	5
7	7	Great Divide Brewing Company, Denver, CO	6
8	8	Tapistry Brewing, Bridgman, MI	7
9	9	Big Lake Brewing, Holland, MI	8
10	10	The Mitten Brewing Company, Grand Rapids, MI	9
11	11	Brewery Vivant, Grand Rapids, MI	10
12	12	Petoskey Brewing, Petoskey, MI	11
13	13	Blackrocks Brewery, Marquette, MI	12
14	14	Perrin Brewing Company, Comstock Park, MI	13
15	15	Witch's Hat Brewing Company, South Lyon, MI	14
16	16	Founders Brewing Company, Grand Rapids, MI	15
17	17	Flat 12 Bierwerks, Indianapolis, IN	16
18	18	Tin Man Brewing Company, Evansville, IN	17
19	19	Black Acre Brewing Co., Indianapolis, IN	18

Showing 1 to 20 of 558 entries

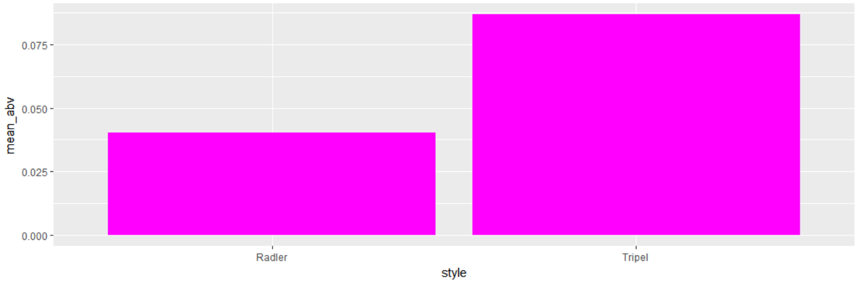
Mean abv for two different beer styles

- Would like to know the abv on two different styles of beers
- Can achieve with a spread and some dplyr functions

```
beers_abv <- beers %>%
  select(-ibu, -brewery_id, -id, -ounces) %>%
  filter(style == "Radler" | style == "Tripel") %>%

  group_by(style) %>%
  summarize(mean_abv = mean(abv)) %>%
  ungroup()


ggplot(data = beers_abv, aes(x=style, y=mean_abv)) +
  geom_bar(stat="identity", fill = 'purple')
```



Common beer names

- Would like to know the most common beer names

```
beers_split_words <- beers %>%
  dplyr::mutate(name = tolower(name)) %>%
  dplyr::mutate(words_in_name = strsplit(name, ' ')) %>%

  tidyr::unnest(words_in_name) %>%

  dplyr::group_by(words_in_name) %>%
  dplyr::summarize(count = n()) %>%

  dplyr::arrange(desc(count)) %>%
  dplyr::filter(!words_in_name %in% stop_words,
                !words_in_name %in% beer_types)
```

	words_in_name	count
1	hop	52
2	summer	48
3	style	26
4	point	25
5	old	21
6	(current)	19
7	big	19
8	river	18
9	winter	18
10	great	16
11	light	16
12	oktoberfest	16
13	barrel	15
14	beach	14
15	honey	14
16	island	14
17	mountain	14
18	pumpkin	14
19	cream	13

Showing 1 to 20 of 2,390 entries

Breweries per state

- Would like to know nr of breweries per state

```
breweries_per_state <- breweries %>%
  dplyr::group_by(city) %>%
  dplyr::summarize(nr_breweries = n()) %>%
  dplyr::ungroup() %>%
  dplyr::arrange(desc(nr_breweries))
```

	city	nr_breweries
1	Portland	17
2	Boulder	9
3	Chicago	9
4	Seattle	9
5	Austin	8
6	Denver	8
7	San Diego	8
8	Bend	6
9	San Francisco	5
10	Anchorage	4
11	Brooklyn	4
12	Cincinnati	4
13	Columbus	4
14	Indianapolis	4
15	Albuquerque	3
16	Athens	3
17	Aurora	3
18	Baltimore	3
19	Charlotte	3

Showing 1 to 20 of 384 entries

The state with the least diversified beer styles

- Would like to know the share of each beer style in the different states, and which of the states that has the least diversified styles

```
breweries_and_beers <- breweries %>%
  right_join(beers, by = c("id" = "brewery_id")) %>%

  group_by(state, style) %>%
  summarize(nr_of_beers = n()) %>%

  group_by(state) %>%
  mutate(style_share = nr_of_beers / sum(nr_of_beers)) %>%
  arrange(desc(style_share))
```

	state	style	nr_of_beers	style_share
1	DE	American IPA	1	0.50000000
2	DE	American Pale Ale (APA)	1	0.50000000
3	WV	American Black Ale	1	0.50000000
4	WV	American Pale Ale (APA)	1	0.50000000
5	NH	Berliner Weissbier	3	0.37500000
6	NJ	American IPA	3	0.37500000
7	ND	American IPA	1	0.33333333
8	ND	American Pale Ale (APA)	1	0.33333333
9	ND	Scottish Ale	1	0.33333333
10	TN	American IPA	2	0.33333333
11	FL	American IPA	19	0.32758621
12	WA	American IPA	21	0.30882353
13	NM	American IPA	4	0.28571429
14	AK	American IPA	7	0.28000000
15	NH	American IPA	2	0.25000000
16	VA	American IPA	10	0.25000000
17	CA	American IPA	45	0.24590164
18	MA	American IPA	19	0.23170732
19	VT	American Double / Imperial IPA	6	0.22222222
20	MO	American IPA	9	0.21428571

Showing 1 to 20 of 989 entries

Future meetup topics:

- Shiny
- R and Python
- Statistical modelling in R
- Machine learning in R
- Version control with GitHub
- Visualize your data with ggplot2
- Tools for reproducible research in R
- Make your own R package
- Suggestions?

Thank you for listening! And thanks to Foo café for having us!

Get in contact with us!

- Twitter: @RLadiesSTHLM
- Facebook: R-Ladies Stockholm
- Meetup: meetup.com/rladies-Stockholm
- Mail: rladies.stockholm@gmail.com

☞ And check out slides and material here:
https://github.com/rladies/meetup-presentations_stockholm

foocafe.org