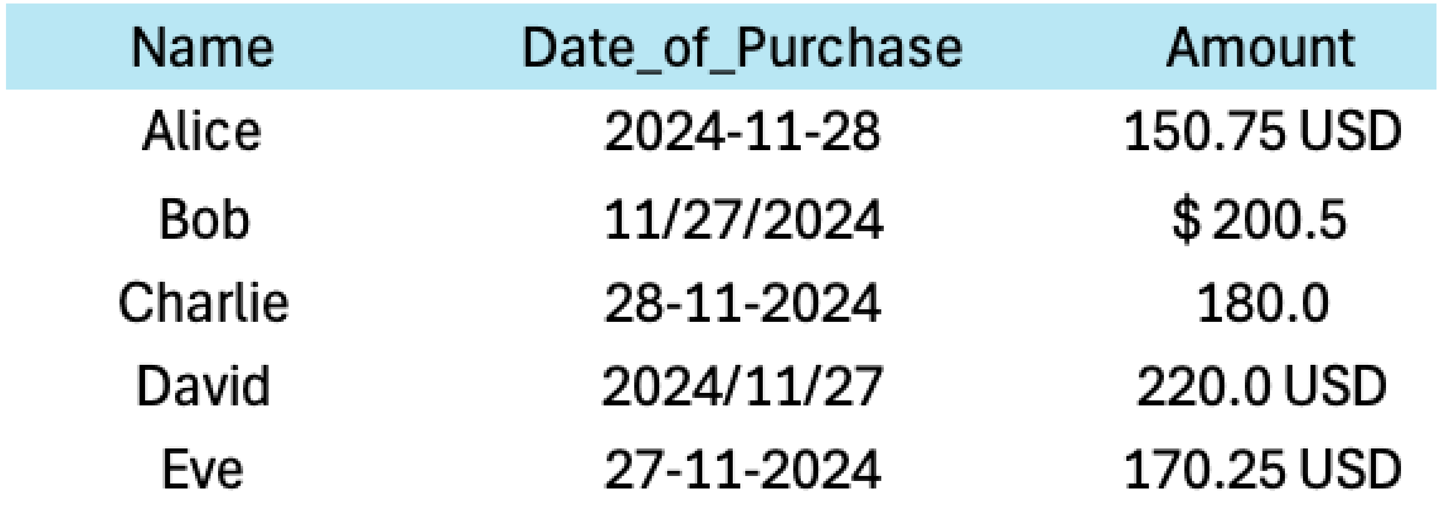
Week 2

Data Management



Name	Date_of_Purchase	Amount
Alice	2024-11-28	150.75 USD
Bob	11/27/2024	$ 200.5
Charlie	28-11-2024	180.0
David	2024/11/27	220.0 USD
Eve	27-11-2024	170.25 USD

Data is messy by nature: It may come from various sources (web scraping, APIs, surveys) and contain errors, duplicates, missing values, or inconsistent formatting.

Garbage In, Garbage Out: Poor quality data leads to inaccurate analysis and unreliable results.

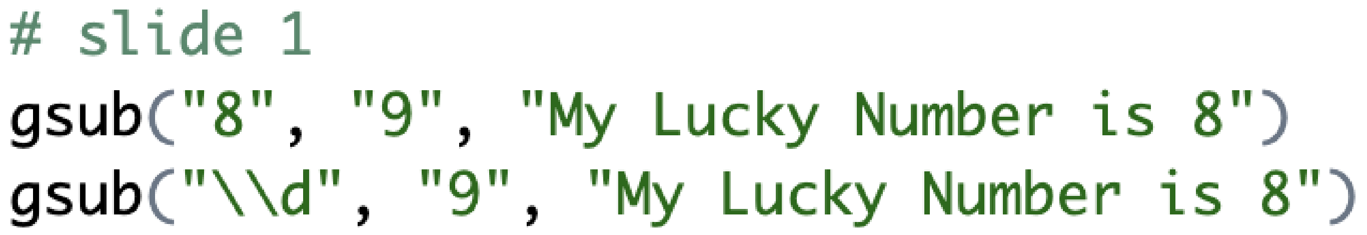
Wrangling:

“engagement in a long,
complicated dispute or argument.”

Cleaning Data

Review of Regex

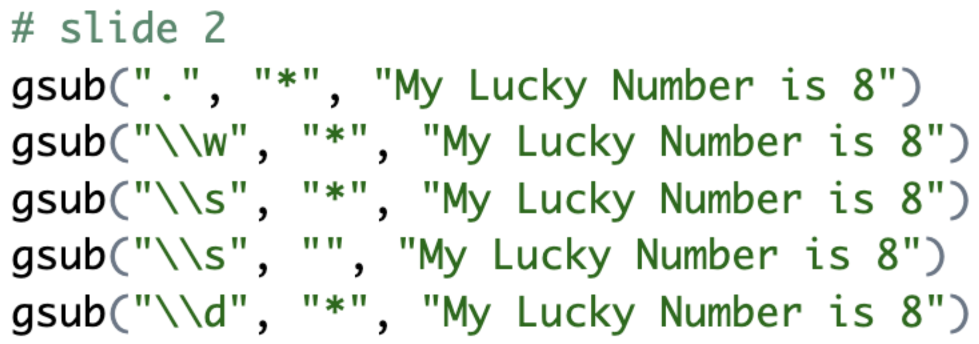
gsub("pattern", "what to replace with", "the string to analyze")



```r
# slide 1
gsub("8", "9", "My Lucky Number is 8")
gsub("\\d", "9", "My Lucky Number is 8")
```

Review of gsub

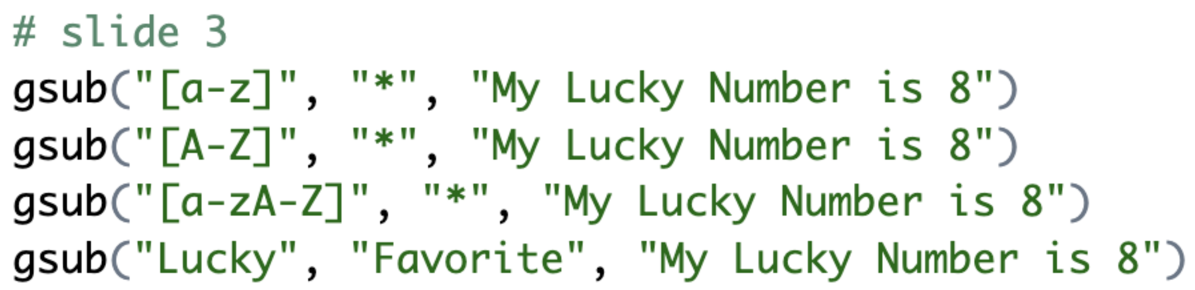
Code	Description
.	Any character
\\w	Any alpha numeric character
\\s	Any white space (including new line: \n)
\\d	Any digit



```r
# slide 2
gsub(".", "*", "My Lucky Number is 8")
gsub("\\w", "*", "My Lucky Number is 8")
gsub("\\s", "*", "My Lucky Number is 8")
gsub("\\s", "", "My Lucky Number is 8")
gsub("\\d", "*", "My Lucky Number is 8")
```

Review of gsub

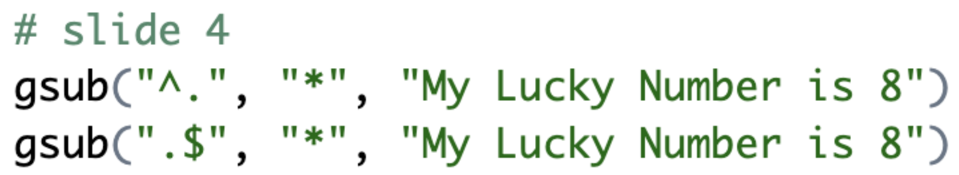
Code	Description
[a-z]	Lower case letters
[A-Z]	Upper case letters
[a-zA-Z]	Lower or upper case letters
Any word	Any digit



```
# slide 3
gsub("[a-z]", "*", "My Lucky Number is 8")
gsub("[A-Z]", "*", "My Lucky Number is 8")
gsub("[a-zA-Z]", "*", "My Lucky Number is 8")
gsub("Lucky", "Favorite", "My Lucky Number is 8")
```

Review of gsub

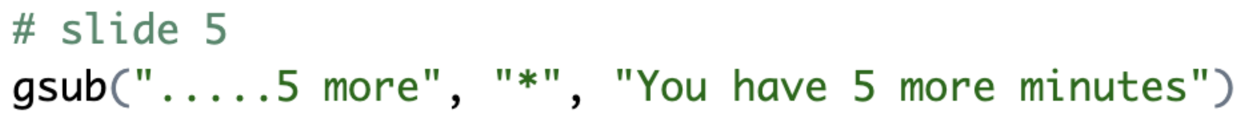
Code	Description
^	Match beginning of string
$	Match end of string



```r
# slide 4
gsub("^.", "*", "My Lucky Number is 8")
gsub(".$", "*", "My Lucky Number is 8")
```

Review of gsub

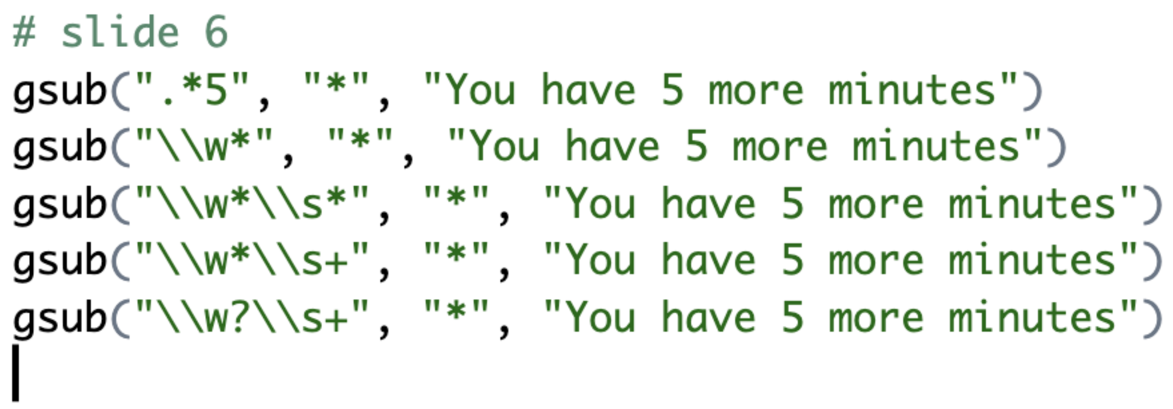
You can combine commands!



```
# slide 5
gsub(".....5 more", "*", "You have 5 more minutes")
```

Review of gsub

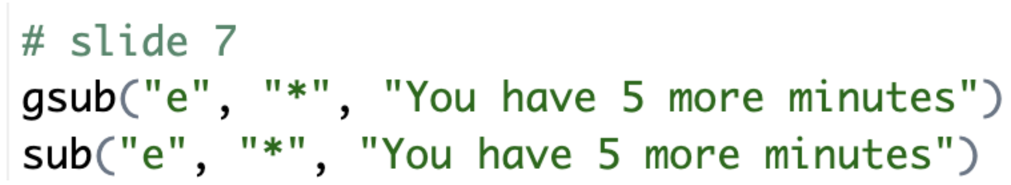
Code	Description
+	1 or more
*	0 or more
?	0 or 1



```
# slide 6
gsub(".*5", "*", "You have 5 more minutes")
gsub("\\w*", "*", "You have 5 more minutes")
gsub("\\w*\\s*", "*", "You have 5 more minutes")
gsub("\\w*\\s+", "*", "You have 5 more minutes")
gsub("\\w?\\s+", "*", "You have 5 more minutes")
```

gsub versus sub

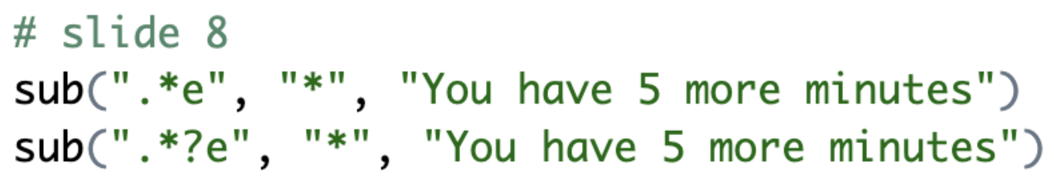
- gsub: replace all
- Sub: replace only 1



```
# slide 7
gsub("e", "*", "You have 5 more minutes")
sub("e", "*", "You have 5 more minutes")
```

Review of gsub

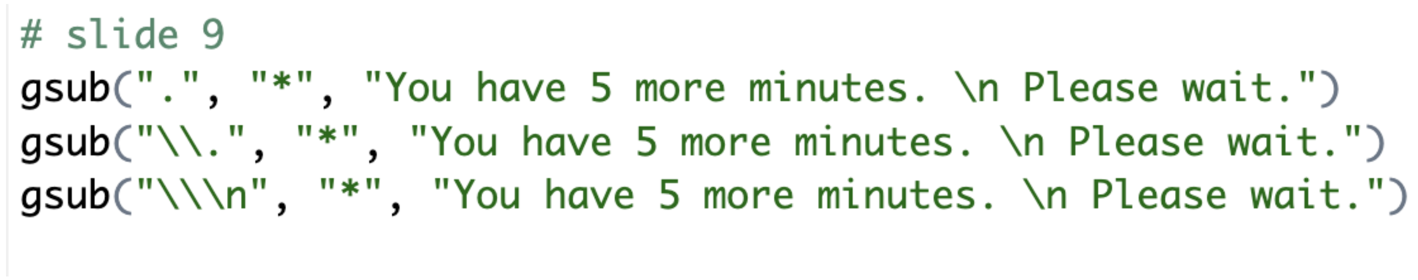
Code	Description
?	Also means lazy versus greedy!



```
# slide 8
sub(".*e", "*", "You have 5 more minutes")
sub(".*?e", "*", "You have 5 more minutes")
```

Escaping Commands

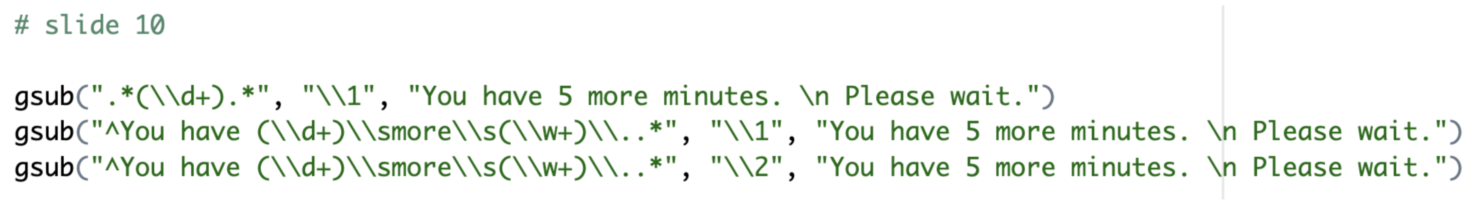
- Lets say I wanted a . to represent a period not ’any character’
  - \\.
- \ represents a special character in R, \\ is a special character in gsub. If I just want to match a \, **you therefore need three!**
  - \\\n – new line “\n” in the text



```
# slide 9
gsub(".", "*", "You have 5 more minutes. \n Please wait.")
gsub("\\.", "*", "You have 5 more minutes. \n Please wait.")
gsub("\\\n", "*", "You have 5 more minutes. \n Please wait.")
```

Review of Regex: You can do get instead of replace

gsub("(pattern to get)", "which one to get", "the string to analyze")



```
# slide 10

gsub(".*(\\d+).*", "\\1", "You have 5 more minutes. \n Please wait.")
gsub("^You have (\\d+)\\smore\\s(\\w+)\\..*", "\\1", "You have 5 more minutes. \n Please wait.")
gsub("^You have (\\d+)\\smore\\s(\\w+)\\..*", "\\2", "You have 5 more minutes. \n Please wait.")
```

Review of Regex: No match will return everything

```
#  slide 11
gsub("cow(\\d*)", "\\1", "You have 5 more minutes. \n Please wait.")
```

Wrangling Data

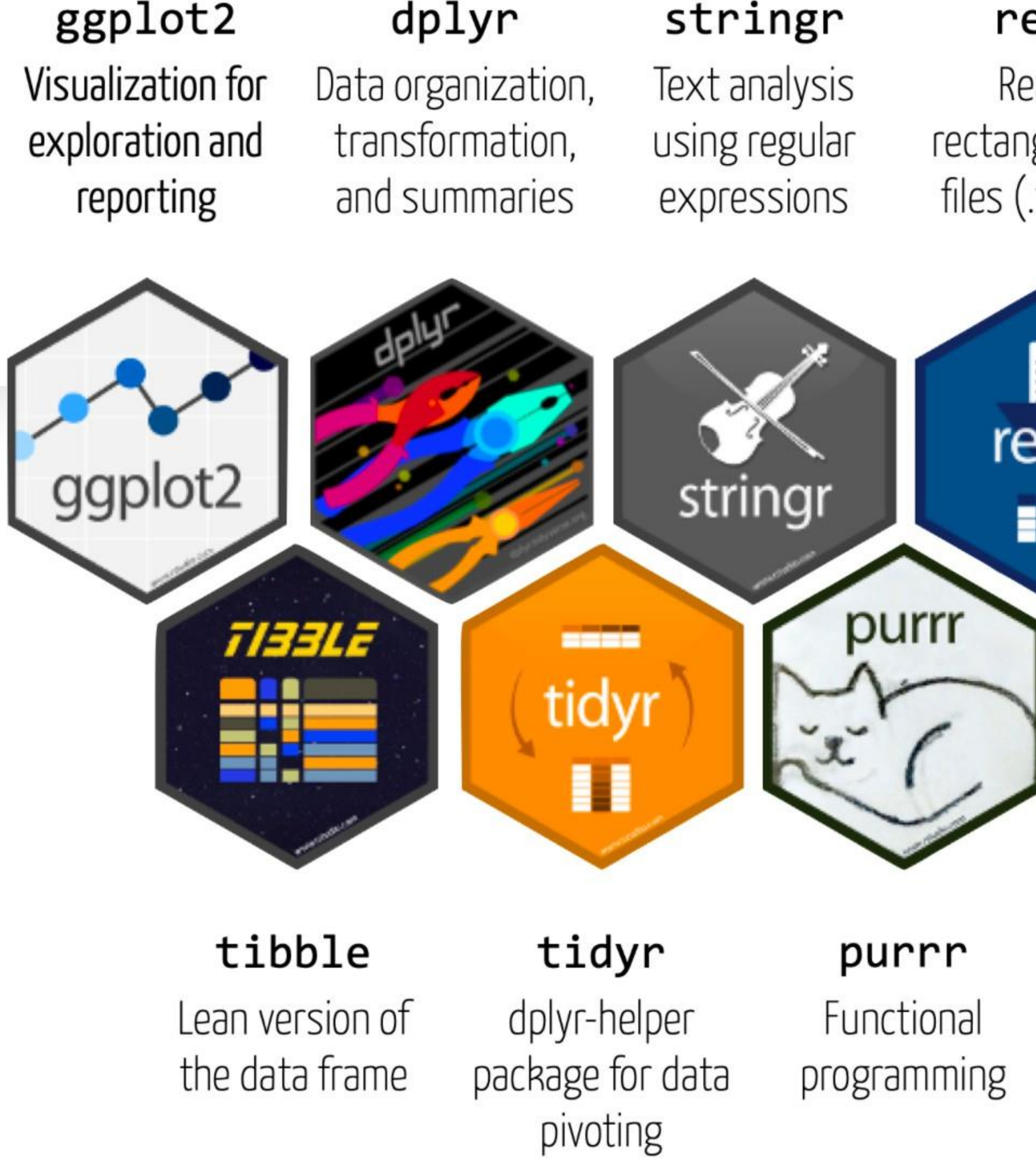
This sections learning goals

- Review of tidyverse (we’ve been using it for a few days now)
- Practice data manipulation with tidyverse for data visualization
- Do a practice that will help you with your homework!

Base R + Tidyverse + RStudio

Tidyverse

- A collection of user-friendly packages for analyzing tidy data
- An ecosystem for analytics and data science with common design principles
- A dialect of the R language


ggplot2
Visualization for exploration and reporting
dplyr
Data organization, transformation, and summaries
stringr
Text analysis using regular expressions
re
Re
rectang
files (
ggplot2
dplyr
stringr
re
TIBBLE
tidyr
purrr
tibble
Lean version of the data frame
tidyr
dplyr-helper package for data pivoting
purrr
Functional programming

%>%

The **novel pipe operator** from the **magrittr** package makes chaining commands easy.

It takes output from the left hand side and feeds it into the function that comes after the pipe.

```
# Without pipe operator
summarize(group_by(filter(mutate(demo_data, Total_Sales = Q1_Sales + Q2_Sales + Q3_Sales + Q4_Sales),
             Region == "North"), Category), mean_sales = mean(Total_Sales, na.rm = TRUE))

# With pipe operator
demo_data %>%
  mutate(Total_Sales = Q1_Sales + Q2_Sales + Q3_Sales + Q4_Sales) %>%
  filter(Region == "North") %>%
  group_by(Category) %>%
  summarize(mean_sales = mean(Total_Sales, na.rm = TRUE))
```

tibble

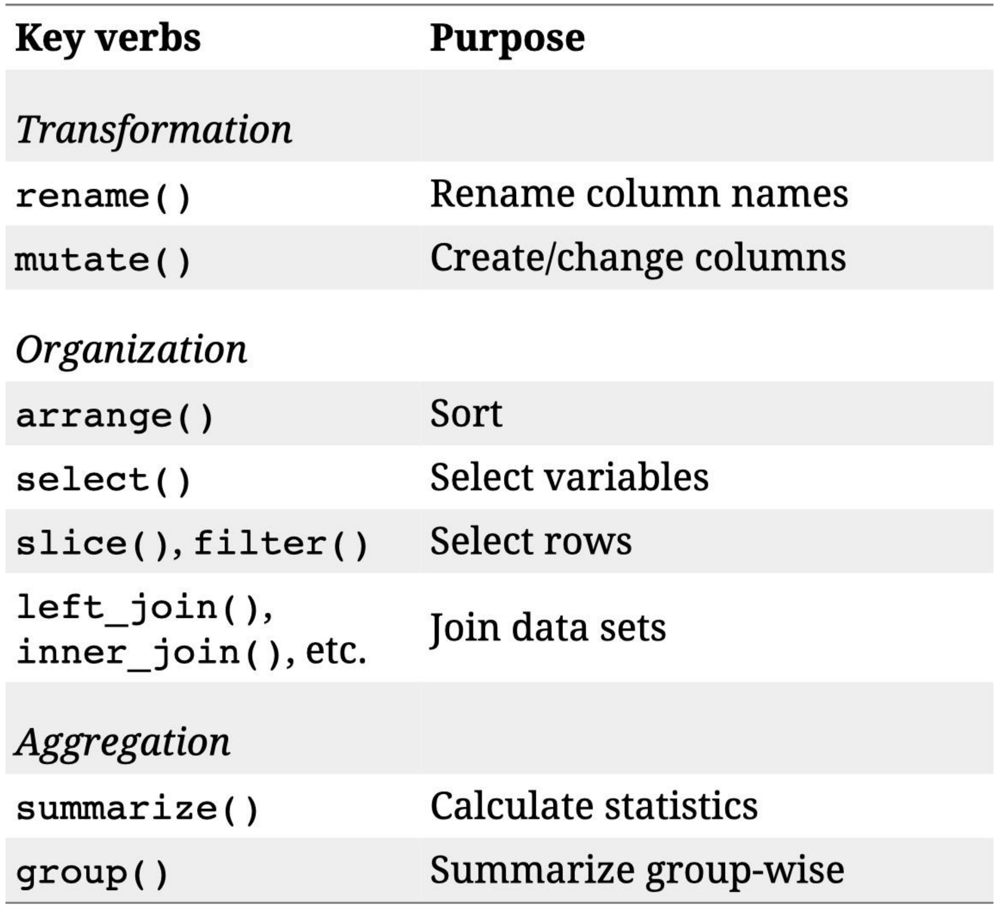
- Benefits over data.frame:
  - **Better print**: More informative and cleaner
  - More consistent subsetting

```
# Read in taxation
basel <- read_csv("1_Data/taxation.csv")

basel
```

```
## # A tibble: 357 × 10
##    year quarter     quarter_no     N
##   <dbl> <chr>            <dbl> <dbl>
## 1  2001 Altstadt …           1  1673
## 2  2001 Vorstädte            2  3204
## 3  2001 Am Ring              3  6579
## 4  2001 Breite               4  5433
## 5  2001 St. Alban            5  6179
## # … with 352 more rows, and 6 more
## #   variables: income_mean <dbl>,
## #   income_median <dbl>,
## #   income_gini <dbl>,
## #   wealth_mean <dbl>,
## #   wealth_median <dbl>,
## #   wealth_gini <dbl>
```

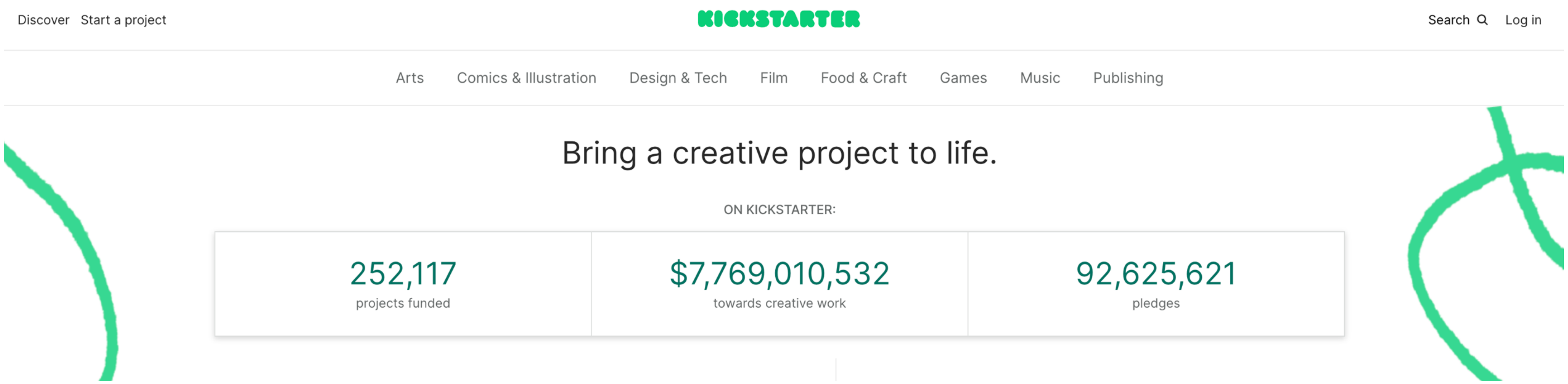
Dplyr

- Benefits over Base R:
  - **No more brackets**
  - **Data masking**
  - Tidy selection
  - Intuitively named functions



Key verbs	**Purpose**
Transformation	
`rename()`	Rename column names
`mutate()`	Create/change columns
Organization	
`arrange()`	Sort
`select()`	Select variables
`slice()`, `filter()`	Select rows
`left_join()`, `inner_join()`, etc.	Join data sets
Aggregation	
`summarize()`	Calculate statistics
`group()`	Summarize group-wise

Introduction to the Data



Bring a creative project to life.

ON KICKSTARTER:

252,117 projects funded

$7,769,010,532 towards creative work

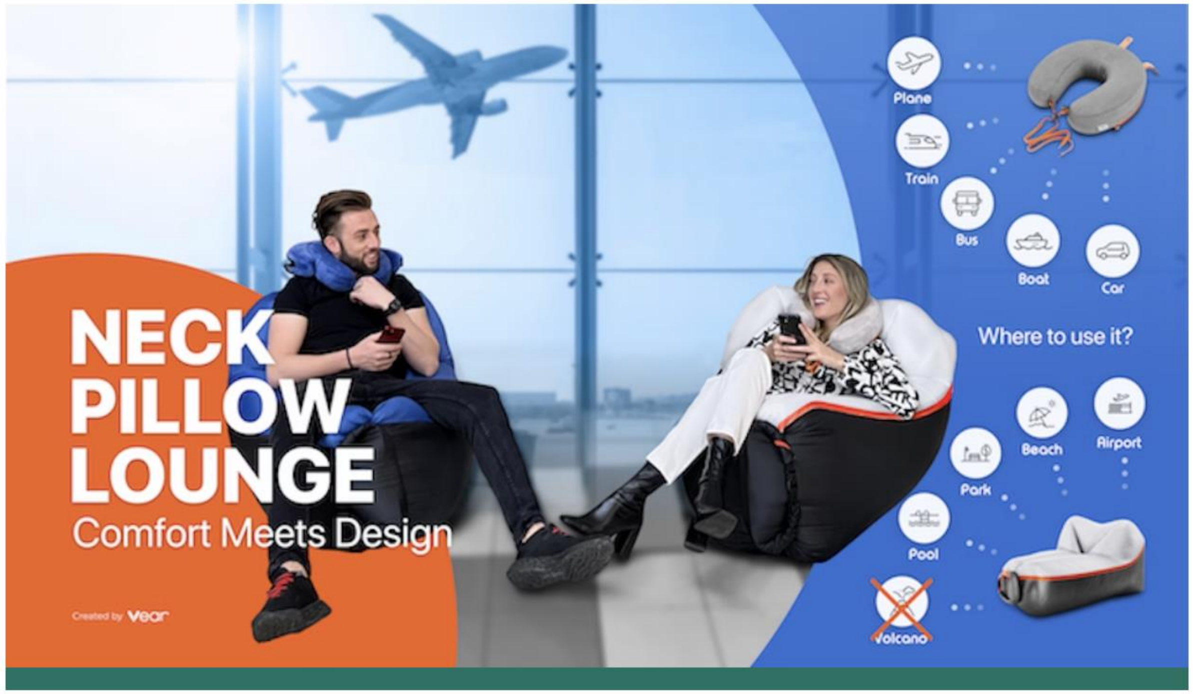
92,625,621 pledges

Design & Tech

From fine design to innovative tech, discover projects from creators working to build a more beautiful future.

Explore Design   Explore Technology

FEATURED PROJECT



✈️ Neck Pillow Lounge™ - For The Ultimate Travel Experience

A neck pillow that turns into an air lounge in seconds. Use it in airports, on planes, on buses, by the pool, etc. It even floats. 🌊

RECOMMENDED FOR YOU

BOLTZ Cutter- The First True Cordless...
9,364% funded
By BOLTZ

oladance OWS Sports Earphones: Unleash...
5,448% funded
By Oladance

Looking Glass Go
1,342% funded
By Looking Glass

‹ 1 2 3 ›

FRESH FAVORITES Discover more >

MakaGiC VS01 Intelligent Electric Vise for DIYer & Maker

Smart clamping | Adjustable intensity | Multiple modes | Expandable peripherals | Large capacity battery | All aluminum alloy body

By MakaGiC

Maliang Magic Pencil: Redefining XR Interactions

Unleash your digital creativity with our innovative XR pencil, powered by the world's most accurate tracking technology.

By Sensoryx AG

Skyted : Stay Connected in Silence

Make silent & confidential calls everywhere, anytime|No noise in, no voice out|Aerospace tech|Breathe in-out comfortably|Lightweight

By Stephane Hersen

MagFree Transform: 3-in-1 Fast Wireless Charger

Innovation Design | X2 Faster | 15W Fast Charges for iPhone 15 | 5W Fast Charges for Apple Watch | Versatile, Transformable, Unrivaled

By INVZI

mui Bc for Mo

Say hell home h body- a

By mui L

Games

From tabletop adventures to beloved revivals, discover the projects forging the future of gameplay.

Explore Games

FEATURED PROJECT

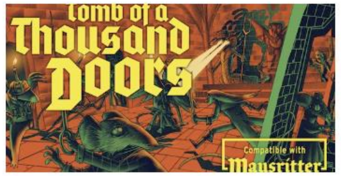
Secret Of The Sea - Deluxe Playing Cards

A deluxe playing cards deck about pirates and sailors

By FRIS Cards

RECOMMENDED FOR YOU

Make 100: Dual game
137% funded
By Zerua Games



Tomb of a Thousand Doors: A Mausritter...
239% funded
By Matthew Morris

The Last Lighthouse
A Scott Almes Solo Game
Plus New Royal Limited Expansions & Expansion Collection Wallet

The Last Lighthouse - A Scott Almes Simpl...
7,679% funded
By Jason Tagmire

< 1 2 3 >

Maliang Magic Pencil: Redefining XR Interactions

Unleash your digital creativity with our innovative XR pencil, powered by the world's most accurate tracking technology.

€1,875
pledged of €58,832 goal

13
backers

31
days to go

Back this project

Remind me

All or nothing. This project will only be funded if it reaches its goal by Tue, February 13 2024 5:00 PM CET.

Project We Love Hardware Zurich, Switzerland

Kickstarter connects creators with backers to fund projects.

Rewards aren't guaranteed, but creators must regularly update backers.

You're only charged if the project meets its funding goal by the campaign deadline.

Select

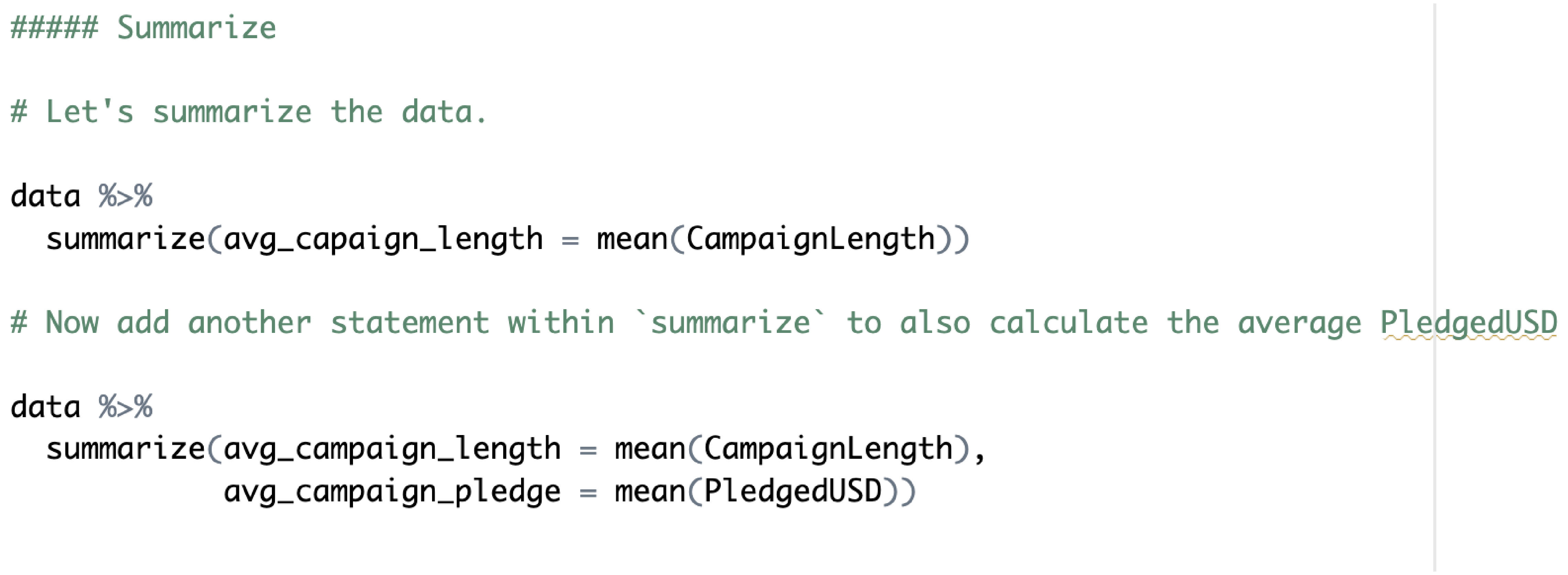
To select columns so that your dataset only includes variables you're interested in, you will use select().

- Reorder columns
- Remove columns
- Rename variables select (OldVariable == New)

```
# Let's select only. the name and blurb

data %>% select(Name, Blurb)

# Let's select only. the name and blurb, rename blurb to Description

data %>% select(Name, Description = Blurb)

# Remove id
data %>% select(-Id)
```

summarize ( )

- It allows us, for
- instance, to calculate the mean price of all of the diamonds within our dataset.



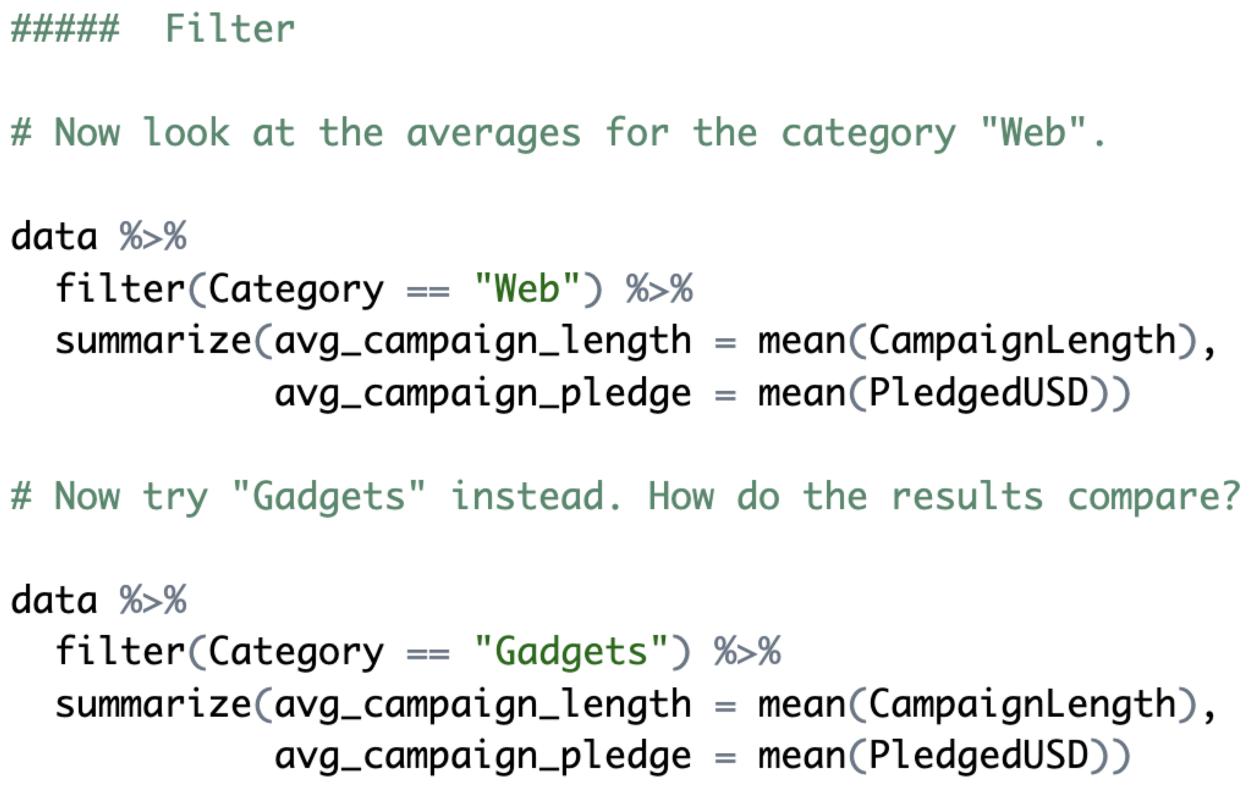
```
##### Summarize

# Let's summarize the data.

data %>%
  summarize(avg_capaign_length = mean(CampaignLength))

# Now add another statement within `summarize` to also calculate the average PledgedUSD

data %>%
  summarize(avg_campaign_length = mean(CampaignLength),
            avg_campaign_pledge = mean(PledgedUSD))
```

The verb summarise() is compatible with almost all the functions in R. Here is a short list of useful functions you can use together with summarise():

Objective	Function	Description
Basic	mean()	Average of vector x
	median()	Median of vector x
	sum()	Sum of vector x
variation	sd()	standard deviation of vector x
	IQR()	Interquartile of vector x
Range	min()	Minimum of vector x
	max()	Maximum of vector x
	quantile()	Quantile of vector x
Position	first()	Use with group_by() First observation of the group
	last()	Use with group_by(). Last observation of the group
	nth()	Use with group_by(). nth observation of the group
Count	n()	Use with group_by(). Count the number of rows
	n_distinct()	Use with group_by(). Count the number of distinct observations

Filter()

- Filter is row wise (not like the select on columns)
- Filter out or keep rows, cleaning the data

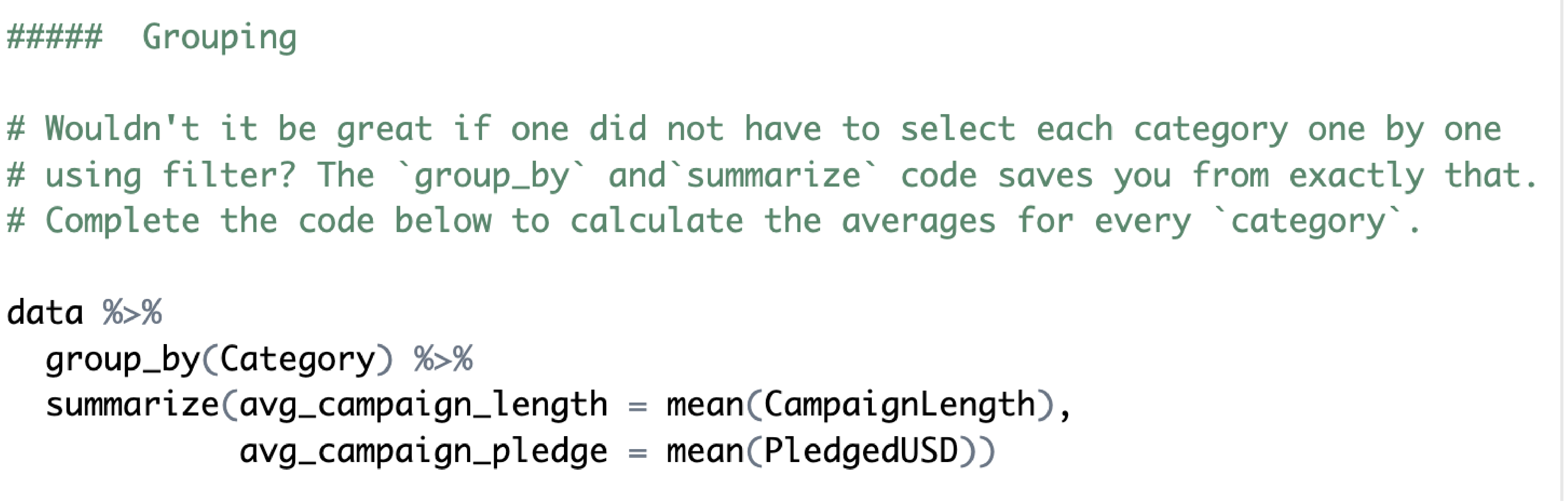


```
#####  Filter

# Now look at the averages for the category "Web".

data %>%
  filter(Category == "Web") %>%
  summarize(avg_campaign_length = mean(CampaignLength),
            avg_campaign_pledge = mean(PledgedUSD))

# Now try "Gadgets" instead. How do the results compare?

data %>%
  filter(Category == "Gadgets") %>%
  summarize(avg_campaign_length = mean(CampaignLength),
            avg_campaign_pledge = mean(PledgedUSD))
```

group by()

Group your data set for a certain categorical variable.

The group_by() function doesn’t change the dataframe data how it looks and it just returns the grouped tbl (tibble table) where we can perform summarise on.

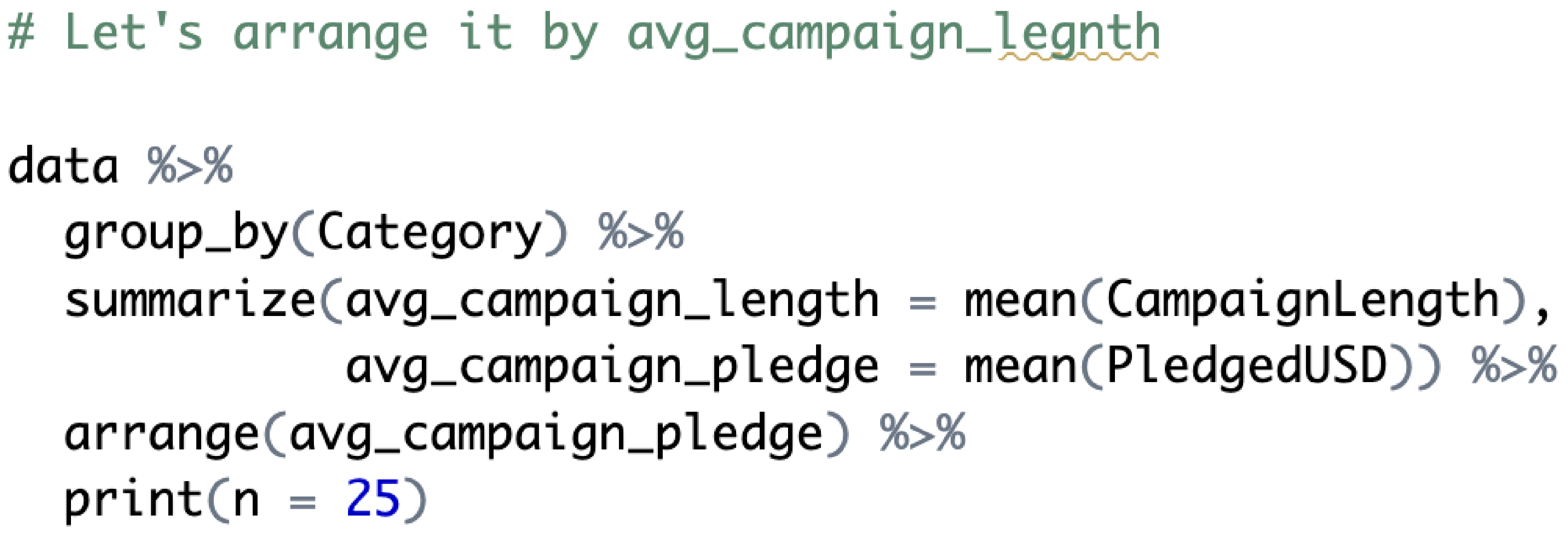


```
#####  Grouping

# Wouldn't it be great if one did not have to select each category one by one
# using filter? The `group_by` and`summarize` code saves you from exactly that.
# Complete the code below to calculate the averages for every `category`.

data %>%
  group_by(Category) %>%
  summarize(avg_campaign_length = mean(CampaignLength),
            avg_campaign_pledge = mean(PledgedUSD))
```

```r
# R has printed only 10 of the 25 categories due to a default setting for the
# `tibble` print. Overwrite this setting by telling R exactly how many rows you
# would like to see.


data %>%
  group_by(Category) %>%
  summarize(avg_campaign_length = mean(CampaignLength),
            avg_campaign_pledge = mean(PledgedUSD)) %>%
  print(n = 25)
```

Arrange()

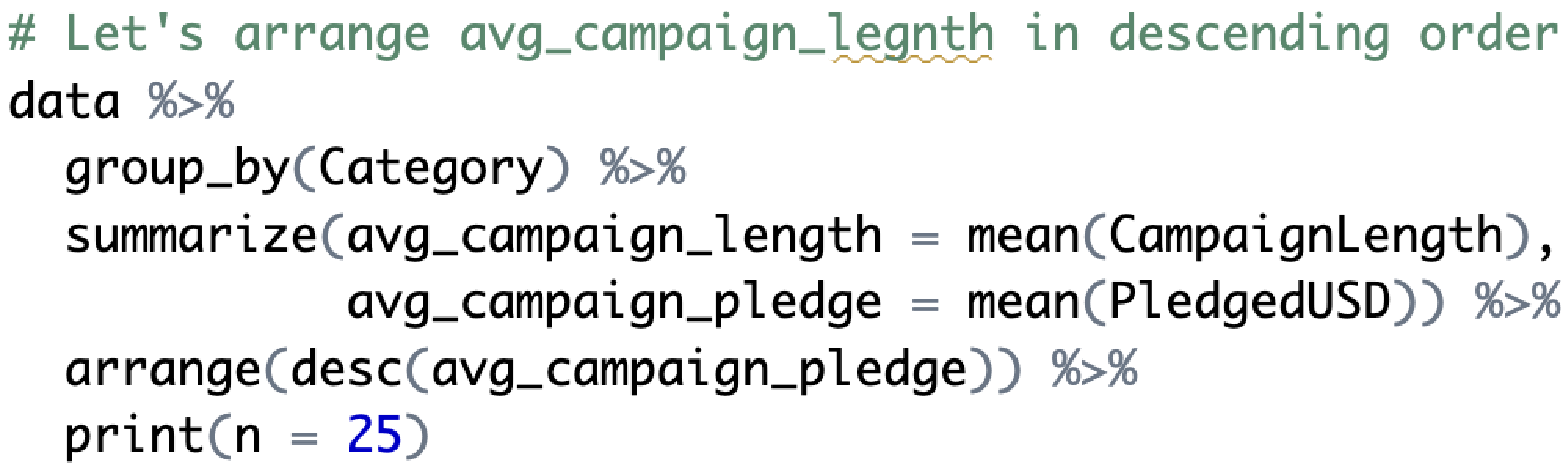
- Orders the rows of a data frame by the values of the select columns (by default, in ascending order)



```
# Let's arrange it by avg_campaign_legnth

data %>%
  group_by(Category) %>%
  summarize(avg_campaign_length = mean(CampaignLength),
            avg_campaign_pledge = mean(PledgedUSD)) %>%
  arrange(avg_campaign_pledge) %>%
  print(n = 25)
```

Desc()

Put in descending order



```r
# Let's arrange avg_campaign_legnth in descending order
data %>%
  group_by(Category) %>%
  summarize(avg_campaign_length = mean(CampaignLength),
            avg_campaign_pledge = mean(PledgedUSD)) %>%
  arrange(desc(avg_campaign_pledge)) %>%
  print(n = 25)
```

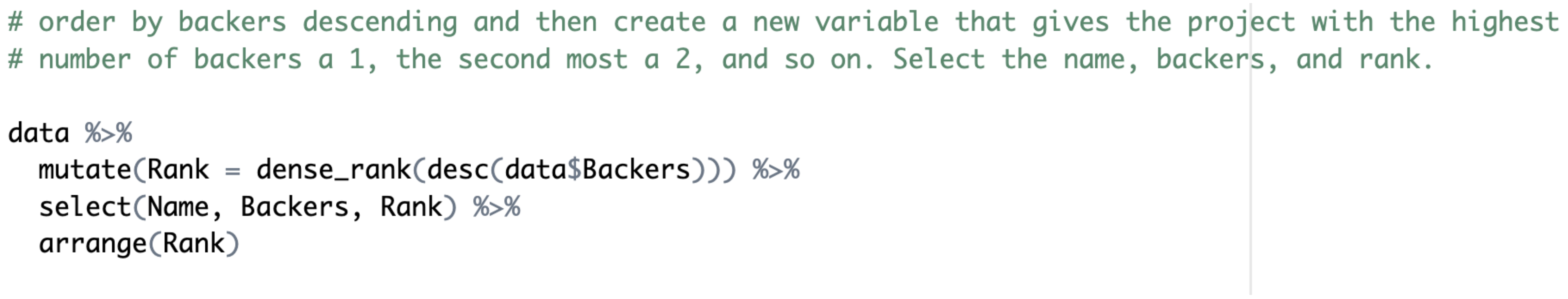
mutate ()

- Create, change a column
- mutate (m = mean(price))
    - This is nesting: one function in another

```
# create a new variable called, and select the name and percentageObtained,
# arrange desc by percentageObtained

data %>%
  mutate(percentageObtained = PledgedUSD / Goal) %>%
  select(Name, percentageObtained) %>%
  arrange(desc(percentageObtained))
```

dense_rank

- returns the rank of rows within a window partition, without any gaps.



```r
# order by backers descending and then create a new variable that gives the project with the highest
# number of backers a 1, the second most a 2, and so on. Select the name, backers, and rank.

data %>%
  mutate(Rank = dense_rank(desc(data$Backers))) %>%
  select(Name, Backers, Rank) %>%
  arrange(Rank)
```

Table()

- Summarize by categories (count or sum)

```r
# create a table that shows the count of each Status
table(data$Status)
```

Tidyr

Helpful when wrangling data.

The main functions we'll cover from tidyr are:

- unite() - combine contents of two or more columns into a single column
- separate() - separate contents of a column into two or more columns

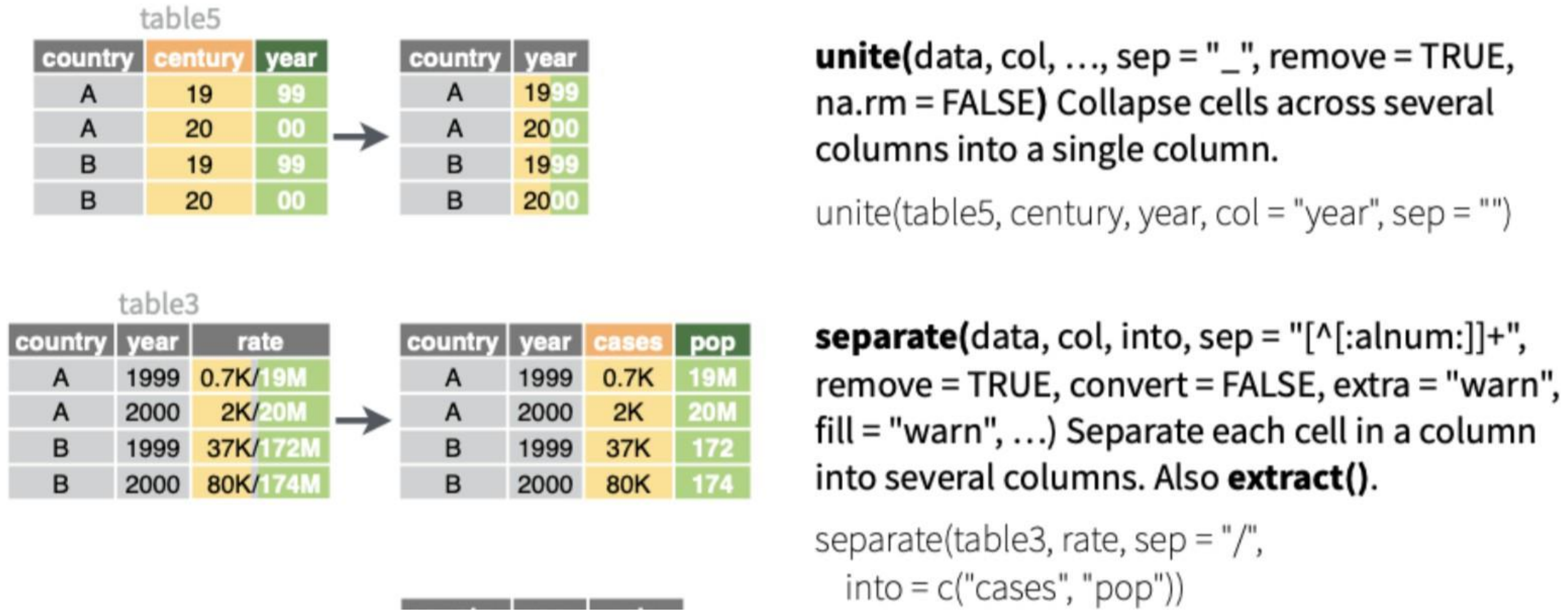


table5

country	century	year
A	19	99
A	20	00
B	19	99
B	20	00

country	year
A	1999
A	2000
B	1999
B	2000

unite(data, col, ..., sep = "_", remove = TRUE, na.rm = FALSE**)** Collapse cells across several columns into a single column.

unite(table5, century, year, col = "year", sep = "")

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174

separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...**)** Separate each cell in a column into several columns. Also **extract()**.

separate(table3, rate, sep = "/",
  into = c("cases", "pop"))

Long vs. Wide Data: What’s the Difference?

A dataset can be written in two different formats: **wide** and **long**.

A **wide** format contains values that *do not* repeat in the first column.

A **long** format contains values that *do* repeat in the first column.

Wide Format

Team	Points	Assists	Rebounds
A	88	12	22
B	91	17	28
C	99	24	30
D	94	28	31

Long Format

Team	Variable	Value
A	Points	88
A	Assists	12
A	Rebounds	22
B	Points	91
B	Assists	17
B	Rebounds	28
C	Points	99
C	Assists	24
C	Rebounds	30
D	Points	94
D	Assists	28
D	Rebounds	31

Notice that in the **wide** dataset, each value in the first column is unique.

Wide Format

Each value is unique in first column

Team	Points	Assists	Rebounds
A	88	12	22
B	91	17	28
C	99	24	30
D	94	28	31

Tidyr

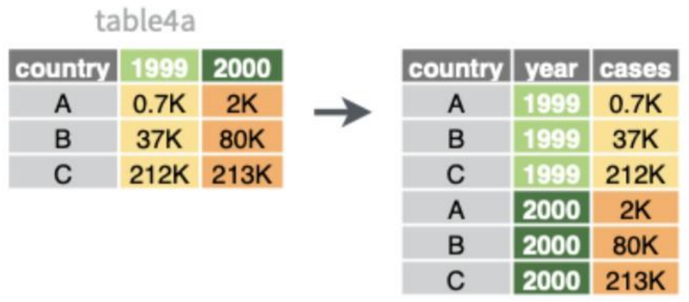


table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

pivot_longer(data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE**)**

"Lengthen" data by collapsing several columns into two. Column names move to a new names_to column and values to a new values_to column.

```
pivot_longer(table4a, cols = 2:3, names_to ="year",
  values_to = "cases")
```

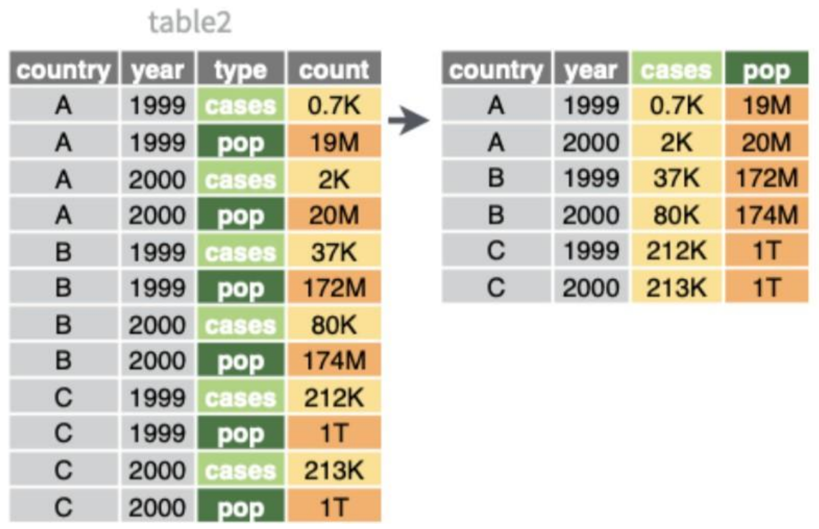


table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

pivot_wider(data, names_from = "name", values_from = "value"**)**

The inverse of pivot_longer(). "Widen" data by expanding two columns into several. One column provides the new column names, the other the values.

```
pivot_wider(table2, names_from = type,
  values_from = count)
```

Which one?

If you're visualizing multiple variables in a plot using statistical software such as R you typically must convert your data to a **long** format in order for the software to create the plot.

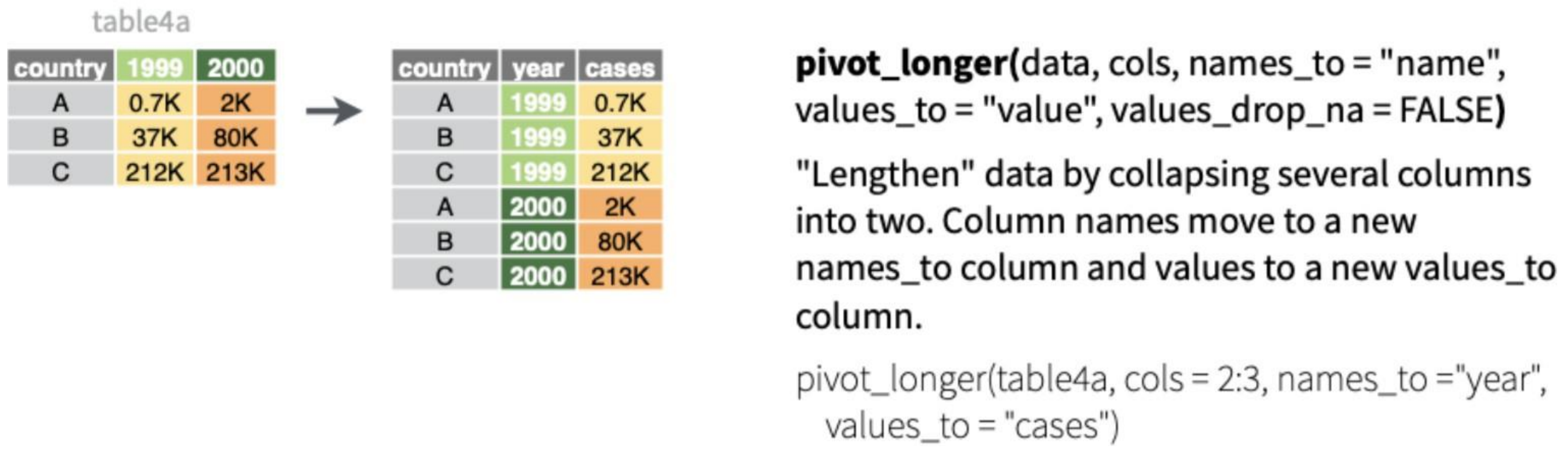


table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

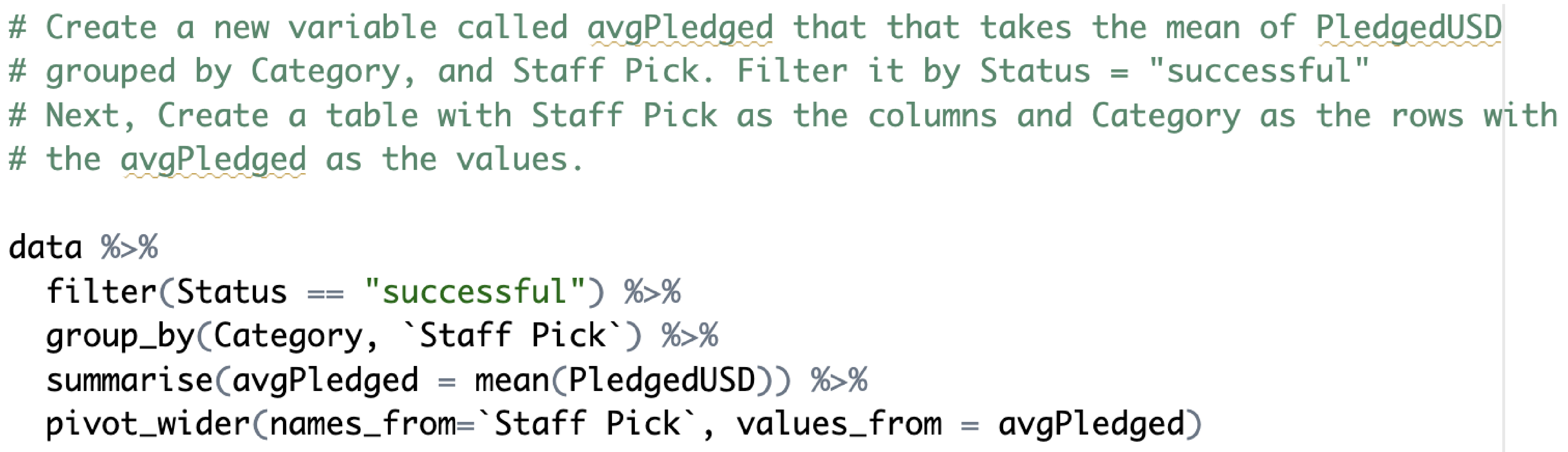
→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

```
pivot_longer(data, cols, names_to = "name",
values_to = "value", values_drop_na = FALSE)
```

"Lengthen" data by collapsing several columns into two. Column names move to a new names_to column and values to a new values_to column.

```
pivot_longer(table4a, cols = 2:3, names_to ="year",
  values_to = "cases")
```



```r
# Create a new variable called avgPledged that that takes the mean of PledgedUSD
# grouped by Category, and Staff Pick. Filter it by Status = "successful"
# Next, Create a table with Staff Pick as the columns and Category as the rows with
# the avgPledged as the values.

data %>%
  filter(Status == "successful") %>%
  group_by(Category, `Staff Pick`) %>%
  summarise(avgPledged = mean(PledgedUSD)) %>%
  pivot_wider(names_from=`Staff Pick`, values_from = avgPledged)
```