Data management

Data wrangling

Matilde Trevisani

What is in a dataset?

Dataset terminology

- Each row is an **observation**
- Each column is a **variable**

`starwars` is a data from the `dplyr` pkg

```
starwars
```

```
## # A tibble: 87 × 14
##   name    height  mass hair_color skin_color eye_color birth_year
##   <chr>    <int> <dbl> <chr>      <chr>      <chr>           <dbl>
## 1 Luke S…    172    77 blond      fair       blue             19
## 2 C-3PO      167    75 <NA>       gold       yellow          112
## 3 R2-D2       96    32 <NA>       white, bl… red              33
## 4 Darth …    202   136 none       white      yellow           41.9
## 5 Leia O…    150    49 brown      light      brown            19
## 6 Owen L…    178   120 brown, gr… light      blue             52
## # ℹ 81 more rows
## # ℹ 7 more variables: sex <chr>, gender <chr>, homeworld <chr>,
## #   species <chr>, films <list>, vehicles <list>,
## #   starships <list>
```

What's in the Star Wars data?

Take a `glimpse` at the data:

```
glimpse(starwars)
```

```
## Rows: 87
## Columns: 14
## $ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth V…
## $ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 1…
## $ mass       <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, …
## $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, gr…
## $ skin_color <chr> "fair", "gold", "white, blue", "white", "lig…
## $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", …
## $ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, N…
## $ sex        <chr> "male", "none", "none", "male", "female", "m…
## $ gender     <chr> "masculine", "masculine", "masculine", "masc…
## $ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine",…
## $ species    <chr> "Human", "Droid", "Droid", "Human", "Human",…
## $ films      <list> <"A New Hope", "The Empire Strikes Back", "…
## $ vehicles   <list> <"Snowspeeder", "Imperial Speeder Bike">, <…
## $ starships  <list> <"X-wing", "Imperial shuttle">, <>, <>, "TI…
```

What does each row represent? What does each column represent?

```
?starwars
```

starwars {dplyr} R Documentation

Starwars characters

Description

This data comes from SWAPI, the Star Wars API, https://swapi.dev/

Usage

```
starwars
```

Format

A tibble with 87 rows and 14 variables:

name
: Name of the character

height
: Height (cm)

mass
: Weight (kg)

hair_color,skin_color,eye_color
: Hair, skin, and eye colors

birth_year
: Year born (BBY = Before Battle of Yavin)

Luke Skywalker

```
height = 172 cm
weight = 77 kg
eye_color = "blue"
skin_color = "fair"
species = "Human"
sex = "male"
gender = "masculine"
homeworld = "Tatooine"
```

```
name = "Luke Skywalker"
hair_color = "blond"
birth_year = 19 BBY
films = c("The Empire Strikes Back",
          "Revenge of the Sith",
          "Return of the Jedi",
          "A New Hope",
          "The Force Awakens")
vehicles = c("Snowspeeder",
             "Imperial Speeder Bike")
starships = c("X-wing",
              "Imperial shuttle")
```

Domanda (rispondere alla fine delle slides):

Come estraggo il *profilo* di Luke? E, in particolare, se volessi solo sapere in quali film della saga è stato protagonista?

```
starwars ...
```

> How many rows and columns does this dataset have?

```
nrow(starwars) # number of rows
```

```
## [1] 87
```

```
ncol(starwars) # number of columns
```

```
## [1] 14
```

```
dim(starwars)  # dimensions (row column)
```

```
## [1] 87 14
```

Exploratory data analysis

What is EDA?

- Exploratory data analysis (EDA) is an approach to analysing data sets to summarize its main characteristics
- Often, this is visual -- this is what we'll focus on next
- But we might also calculate summary statistics and perform data wrangling /manipulation/transformation at (or before) this stage of the analysis -- this is what we'll focus on first

> Data wrangling is the process of converting raw data into a usable form. It may also be called data munging or data remediation.

Tidy data

> Happy families are all alike; every unhappy family is unhappy in its own way.
>
> Leo Tolstoy

> Le serie di dati ordinate sono tutte uguali, ma ogni serie di dati disordinata è disordinata a modo suo".
>
> Hadley Wickham

Characteristics of tidy data:

1. Ogni variabile deve avere la propria colonna.
2. Ogni osservazione deve avere la sua riga.
3. Ogni valore deve avere la propria cella.

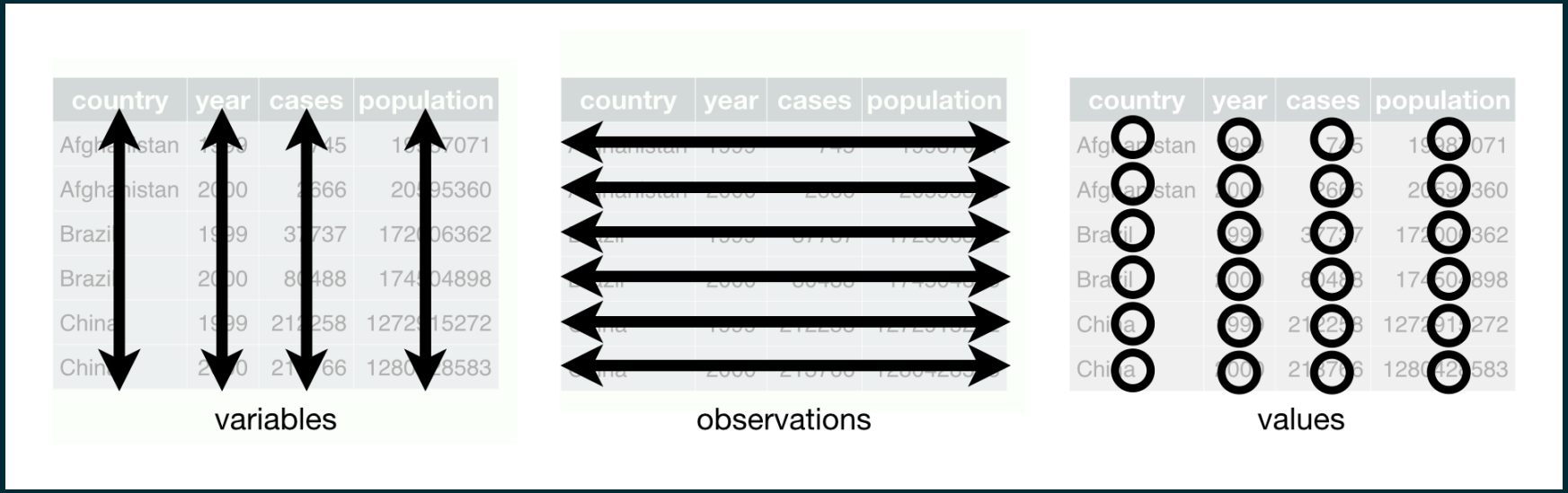
- Ogni tipo di unità (situazione) d'osservazione forma una tabella (`tibble`)

Characteristics of untidy data:

!@#$%^&*()

Tidy data

(dati ordinati, organizzati)



Una volta che avete dati ordinati con gli strumenti forniti dai pacchetti del `tidyverse`, passerete molto meno tempo a gestire i dati (eventualmente cambiando rappresentazione quando necessario), permettendovi di dedicare più tempo alle domande analitiche che avete a disposizione.

What makes this data not tidy?

Airplanes on Hand in the AAF, By Major Type: Jul 1939 to Aug 1945

End of Month	Total	Very Heavy Bombers	Heavy Bombers	Medium Bombers	Light Bombers	Fighters	Recon-naissance	Transports	Trainers	Communi-cations
1939										
Jul	2,402	-	16	400	276	494	356	118	735	7
Aug	2,440	-	18	414	276	492	359	129	745	7
[Germany invades Poland, 1 Sep 1939]										
Sep	2,473	-	22	428	278	489	359	136	754	7
Oct	2,507	-	27	446	277	490	365	137	758	7
Nov	2,536	-	32	458	275	498	375	136	755	7
Dec	2,546	-	39	464	274	492	378	131	761	7
1940										
Jan	2,588	-	45	466	271	464	409	128	798	7
Feb	2,658	-	49	470	271	458	415	128	860	7
Mar	2,709	-	54	468	267	453	415	125	920	7
Apr	2,806	-	54	468	263	451	416	125	1,022	7
May	2,906	-	54	470	259	459	410	124	1,123	7
Jun	2,966	-	54	478	166	477	414	127	1,243	7
[France surrenders to Germany, 25 Jun 1940] [Battle of Britain begins, 10 July 1940]										
Jul	3,102	-	56	483	161	500	410	128	1,357	7
Aug	3,295	-	65	485	158	539	407	128	1,506	7

Source: Army Air Forces Statistical Digest, WW II

What makes this data not tidy?

	A	AA	AB	AC	AD	AE	AF	AG	AH
1	Estimated HIV Prevalence% - (Ages 15-49)	2004	2005	2006	2007	2008	2009	2010	2011
2	Abkhazia								
3	Afghanistan						0.06	0.06	0.06
4	Akrotiri and Dhekelia								
5	Albania								
6	Algeria	0.1	0.1	0.1	0.1	0.1			
7	American Samoa								
8	Andorra								
9	Angola	1.9	1.9	1.9	1.9	2	2.1	2.1	2.1
10	Anguilla								
11	Antigua and Barbuda								
12	Argentina	0.4	0.4	0.4	0.4	0.5	0.4	0.4	0.4
13	Armenia	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2
14	Aruba								
15	Australia	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2
16	Austria	0.2	0.2	0.2	0.3	0.3	0.3	0.4	0.4
17	Azerbaijan	0.06	0.06	0.06	0.1	0.1	0.1	0.1	0.1
18	Bahamas	3	3	3	3.1	3.1	2.9	2.8	2.8

Source: Gapminder, Estimated HIV prevalence among 15-49 year olds

What makes this data not tidy?

Subject	United States			
	Estimate	Margin of Error	Percent	Percent Margin of Error
EMPLOYMENT STATUS				
Population 16 years and over	255,797,692	+/-17,051	255,797,692	(X)
In labor force	162,184,325	+/-135,158	63.4%	+/-0.1
Civilian labor force	161,159,470	+/-127,501	63.0%	+/-0.1
Employed	150,599,165	+/-138,066	58.9%	+/-0.1
Unemployed	10,560,305	+/-27,385	4.1%	+/-0.1
Armed Forces	1,024,855	+/-10,363	0.4%	+/-0.1
Not in labor force	93,613,367	+/-126,007	36.6%	+/-0.1
Civilian labor force	161,159,470	+/-127,501	161,159,470	(X)
Unemployment Rate	(X)	(X)	6.6%	+/-0.1
Females 16 years and over	131,092,196	+/-11,187	131,092,196	(X)
In labor force	76,493,327	+/-75,824	58.4%	+/-0.1
Civilian labor force	76,350,498	+/-75,238	58.2%	+/-0.1
Employed	71,451,559	+/-79,007	54.5%	+/-0.1
Own children of the householder under 6 years	22,939,897	+/-14,240	22,939,897	(X)
All parents in family in labor force	14,957,537	+/-36,506	65.2%	+/-0.1
Own children of the householder 6 to 17 years	47,007,147	+/-19,644	47,007,147	(X)
All parents in family in labor force	33,238,793	+/-49,036	70.7%	+/-0.1

Source: US Census Fact Finder, General Economic Characteristics, ACS 2017

Displaying vs. summarising data

Output Code

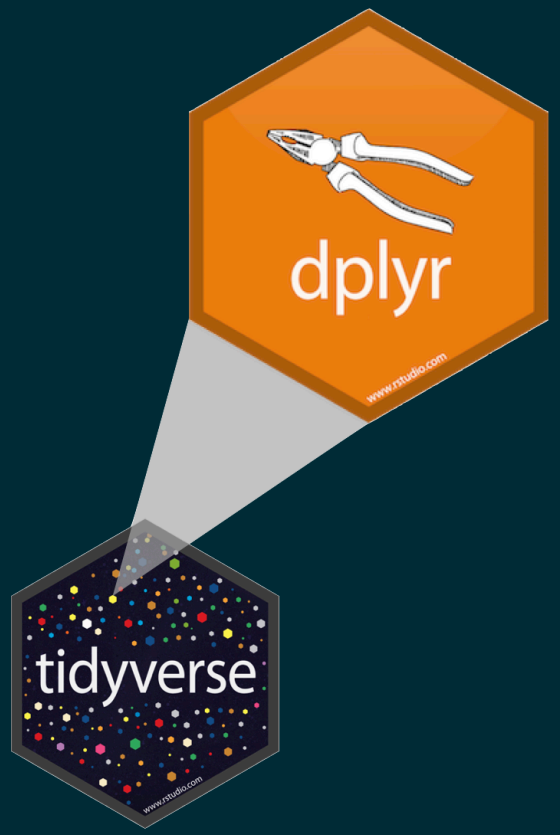
```
## # A tibble: 87 × 3
##   name           height  mass
##   <chr>           <int> <dbl>
## 1 Luke Skywalker    172    77
## 2 C-3PO             167    75
## 3 R2-D2              96    32
## 4 Darth Vader       202   136
## 5 Leia Organa       150    49
## 6 Owen Lars         178   120
## # ℹ 81 more rows
```

```
## # A tibble: 3 × 2
##   gender    avg_ht
##   <chr>      <dbl>
## 1 feminine    167.
## 2 masculine   177.
## 3 <NA>        175
```

Grammar of data wrangling

A grammar of data wrangling...

... based on the concepts of functions as verbs that manipulate data frames



- `rename`: rename column names
- `select`: pick columns by name
- `arrange`: reorder rows
- `slice`: pick rows using index(es)
- `filter`: pick rows matching criteria
- `distinct`: filter for unique rows
- `mutate`: add new variables
- `summarise`: reduce variables to values
- `group_by`: for grouped operations
- ... (many more)

Rules of dplyr functions

- First argument is *always* a data frame
- Subsequent arguments say what to do with that data frame
- Always return a data frame
- Don't modify in place

Data: Hotel bookings

- Data from two hotels: one resort and one city hotel
- Observations: Each row represents a hotel booking
- Goal for original data collection: Development of prediction models to classify a hotel booking's likelihood to be cancelled (Antonia et al., 2019)

```
hotels <- read_csv("data/hotels.csv")
```

Source: TidyTuesday

First look: Variables

```
names(hotels)
```

```
##  [1] "hotel"
##  [2] "is_canceled"
##  [3] "lead_time"
##  [4] "arrival_date_year"
##  [5] "arrival_date_month"
##  [6] "arrival_date_week_number"
##  [7] "arrival_date_day_of_month"
##  [8] "stays_in_weekend_nights"
##  [9] "stays_in_week_nights"
## [10] "adults"
## [11] "children"
## [12] "babies"
## [13] "meal"
## [14] "country"
## [15] "market_segment"
## [16] "distribution_channel"
## [17] "is_repeated_guest"
## [18] "previous_cancellations"
...
```

Second look: Overview

```
glimpse(hotels)
```

```
## Rows: 119,390
## Columns: 32
## $ hotel                         <chr> "Resort Hotel", "Resort …
## $ is_canceled                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, …
## $ lead_time                     <dbl> 342, 737, 7, 13, 14, 14,…
## $ arrival_date_year             <dbl> 2015, 2015, 2015, 2015, …
## $ arrival_date_month            <chr> "July", "July", "July", …
## $ arrival_date_week_number      <dbl> 27, 27, 27, 27, 27, 27, …
## $ arrival_date_day_of_month     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, …
## $ stays_in_weekend_nights       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, …
## $ stays_in_week_nights          <dbl> 0, 0, 1, 1, 2, 2, 2, 2, …
## $ adults                        <dbl> 2, 2, 1, 1, 2, 2, 2, 2, …
## $ children                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, …
## $ babies                        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, …
## $ meal                          <chr> "BB", "BB", "BB", "BB", …
## $ country                       <chr> "PRT", "PRT", "GBR", "GB…
## $ market_segment                <chr> "Direct", "Direct", "Dir…
## $ distribution_channel          <chr> "Direct", "Direct", "Dir…
...
```

Select a single column

View only `lead_time` (number of days between booking and arrival date):

```
select(hotels, lead_time)
```

```
## # A tibble: 119,390 × 1
##   lead_time
##       <dbl>
## 1       342
## 2       737
## 3         7
## 4        13
## 5        14
## 6        14
## # ℹ 119,384 more rows
```

Select a single column

```
select(
  hotels,
  lead_time
  )
```

- Start with the function (a verb): `select()`

Select a single column

```
select(
  hotels,
  lead_time
  )
```

- Start with the function (a verb): `select()`
- First argument: data frame we're working with , `hotels`

Select a single column

```
select(
  hotels,
  lead_time
  )
```

- Start with the function (a verb): `select()`
- First argument: data frame we're working with , `hotels`
- Second argument: variable we want to select, `lead_time`

Select a single column

```
select(
  hotels,
  lead_time
  )
```

```
## # A tibble: 119,390 × 1
##   lead_time
##       <dbl>
## 1       342
## 2       737
## 3         7
## 4        13
## 5        14
## 6        14
## # ℹ 119,384 more rows
```

- Start with the function (a verb): `select()`
- First argument: data frame we're working with , `hotels`
- Second argument: variable we want to select, `lead_time`
- Result: data frame with 119390 rows and 1 column

> dplyr functions always expect a data frame and always yield a data frame.

```
select(hotels, lead_time)
```

```
## # A tibble: 119,390 × 1
##   lead_time
##       <dbl>
## 1       342
## 2       737
## 3         7
## 4        13
## 5        14
## 6        14
## # ℹ 119,384 more rows
```

Select multiple columns

View only the `hotel` type and `lead_time`:

```
select(hotels, hotel, lead_time)
```

```
## # A tibble: 119,390 × 2
##   hotel        lead_time
##   <chr>            <dbl>
## 1 Resort Hotel       342
## 2 Resort Hotel       737
## 3 Resort Hotel         7
## 4 Resort Hotel        13
## 5 Resort Hotel        14
## 6 Resort Hotel        14
## # ℹ 119,384 more rows
```

> What if we wanted to select these columns, and then arrange the data in descending order of lead time?

Data wrangling, step-by-step

Select:

```
hotels %>%
  select(hotel, lead_time)
```

```
## # A tibble: 119,390 × 2
##   hotel        lead_time
##   <chr>            <dbl>
## 1 Resort Hotel       342
## 2 Resort Hotel       737
## 3 Resort Hotel         7
## 4 Resort Hotel        13
## 5 Resort Hotel        14
## 6 Resort Hotel        14
## # ℹ 119,384 more rows
```

Select, then arrange:

```
hotels %>%
  select(hotel, lead_time) %>%
  arrange(desc(lead_time))
```

```
## # A tibble: 119,390 × 2
##   hotel        lead_time
##   <chr>            <dbl>
## 1 Resort Hotel       737
## 2 Resort Hotel       709
## 3 City Hotel         629
## 4 City Hotel         629
## 5 City Hotel         629
## 6 City Hotel         629
## # ℹ 119,384 more rows
```

Pipes

What is a pipe?

In programming, a pipe is a technique for passing information from one process to another.

- Start with the data frame `hotels`, and pass it to the `select()` function,

```
hotels %>%
  select(hotel, lead_time) %>%
  arrange(desc(lead_time))
```

```
## # A tibble: 119,390 × 2
##   hotel        lead_time
##   <chr>            <dbl>
## 1 Resort Hotel       737
## 2 Resort Hotel       709
## 3 City Hotel         629
## 4 City Hotel         629
## 5 City Hotel         629
## 6 City Hotel         629
## # ℹ 119,384 more rows
```

What is a pipe?

In programming, a pipe is a technique for passing information from one process to another.

- Start with the data frame `hotels`, and pass it to the `select()` function,
- then we select the variables `hotel` and `lead_time`,

```
hotels %>%
  select(hotel, lead_time) %>%
  arrange(desc(lead_time))
```

```
## # A tibble: 119,390 × 2
##   hotel        lead_time
##   <chr>            <dbl>
## 1 Resort Hotel       737
## 2 Resort Hotel       709
## 3 City Hotel         629
## 4 City Hotel         629
## 5 City Hotel         629
## 6 City Hotel         629
## # ℹ 119,384 more rows
```

What is a pipe?

In programming, a pipe is a technique for passing information from one process to another.

- Start with the data frame `hotels`, and pass it to the `select()` function,
- then we select the variables `hotel` and `lead_time`,
- and then we arrange the data frame by `lead_time` in descending order.

```
hotels %>%
  select(hotel, lead_time) %>%
  arrange(desc(lead_time))
```

```
## # A tibble: 119,390 × 2
##   hotel        lead_time
##   <chr>            <dbl>
## 1 Resort Hotel       737
## 2 Resort Hotel       709
## 3 City Hotel         629
## 4 City Hotel         629
## 5 City Hotel         629
## 6 City Hotel         629
## # ℹ 119,384 more rows
```

Aside

The pipe operator is implemented in the package **magrittr**, though we don't need to load this package explicitly since **tidyverse** does this for us.

> Any guesses as to why the package is called magrittr?

How does a pipe work?

- You can think about the following sequence of actions - find keys, unlock car, start car, drive to work, park.
- Expressed as a set of nested functions in R pseudocode this would look like:

```
park(drive(start_car(find("keys")), to = "work"))
```

- Writing it out using pipes give it a more natural (and easier to read) structure:

```
find("keys") %>%
  start_car() %>%
  drive(to = "work") %>%
  park()
```

A note on piping and layering

- %>% used mainly in **dplyr** pipelines, *we pipe the output of the previous line of code as the first input of the next line of code*
- + used in **ggplot2** plots is used for "layering", *we create the plot in layers, separated by +*

dplyr

❌

```
hotels +
  select(hotel, lead_time)
```

```
## Error: oggetto 'hotel' non trovato
```

✅

```
hotels %>%
  select(hotel, lead_time)
```

```
## # A tibble: 119,390 × 2
##   hotel        lead_time
##   <chr>            <dbl>
## 1 Resort Hotel       342
## 2 Resort Hotel       737
## 3 Resort Hotel         7
...
```

ggplot2

❌

```
ggplot(hotels, aes(x = hotel, fill = deposit_type)) %>%
  geom_bar()
```

```
## Error in `geom_bar()`:
## ! `mapping` must be created by `aes()`.
## ℹ Did you use `%>%` or `|>` instead of `+`?
```

✅

```
ggplot(hotels, aes(x = hotel, fill = deposit_type)) +
  geom_bar()
```

Code styling

Many of the styling principles are consistent across `%>%` and `+`:

- always a space before
- always a line break after (for pipelines with more than 2 lines)

❌

```
ggplot(hotels,aes(x=hotel,y=deposit_type))+geom_bar()
```

✅

```
ggplot(hotels, aes(x = hotel, y = deposit_type)) +
  geom_bar()
```