Econ 106
Data Analysis in Economics

Lecture 3
Fall 2024

Based on: http://www.datacarpentry.org/R-ecology-lesson/

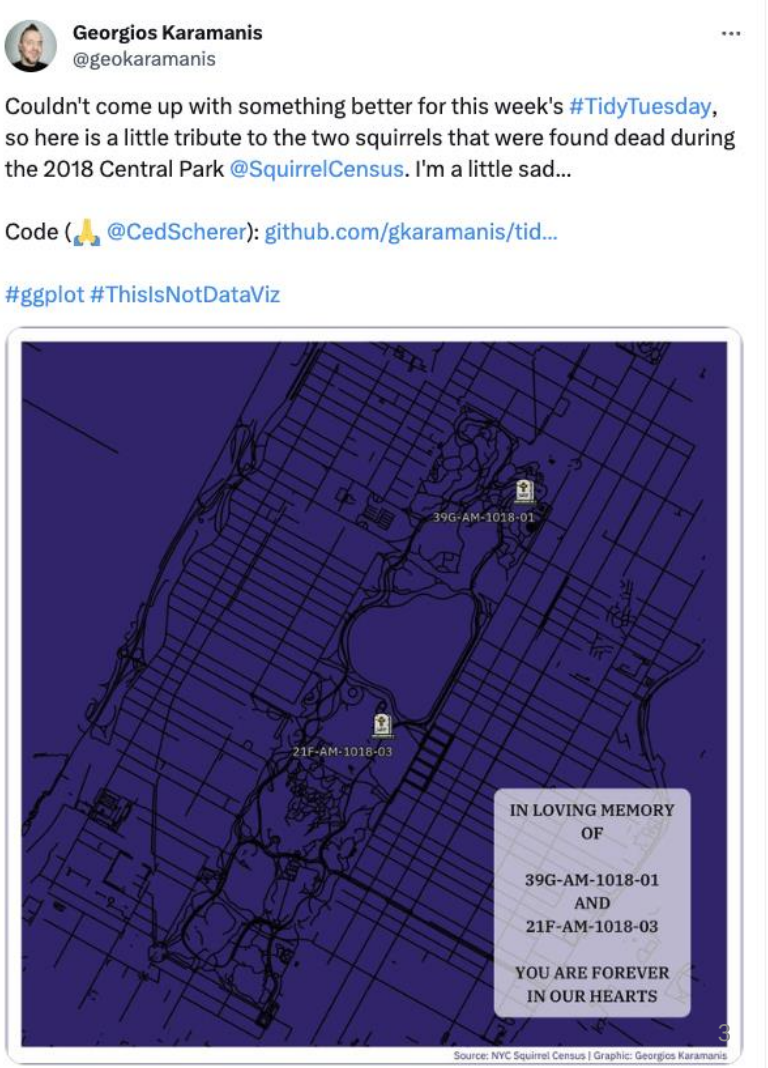
Reminders

- Poll Everywhere will count towards your grade starting this week
  - grade will be based on the percentage of polls you answer correctly

https://pollev.com/vsovero

#tidytuesday

- https://www.bloomberg.com/news/articles/2019-06-24/how-many-squirrels-are-in-nyc-s-central-park
- data: https://github.com/rfordatascience/tidytuesday/tree/master/data/2019/2019-10-29



Outline

- Tidyverse basics
- Data wrangling with dplyr

The tidyverse Package

- The most powerful, intuitive, and popular approach to data cleaning, wrangling, and visualization in R

Advantages:

- Consistent philosophy and syntax (package is maintained!!!!)
- "Verb" based approach makes it more familiar to users of Stata/SAS/SPSS

Packages inside the tidyverse package

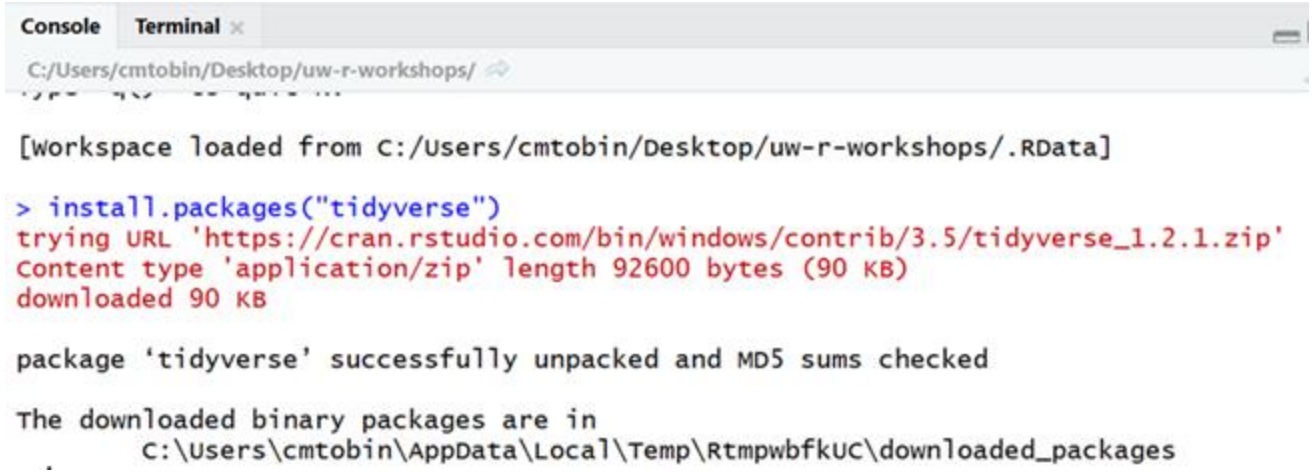
- lots of packages inside the tidyverse (important ones highlighted)
- you won't need to load these separately in order to use them

"broom"	"conflicted"	"cli"	"dbplyr"
"dplyr"	"dtplyr"	"forcats"	"ggplot2"
"googledrive"	"googlesheets4"	"haven"	"hms"
"httr"	"jsonlite"	"lubridate"	"magrittr"
"modelr"	"pillar"	"purrr"	"ragg"
"readr"	"readxl"	"reprex"	"rlang"
"rstudioapi"	"rvest"	"stringr"	"tibble"
"tidyr"	"xml2"	"tidyverse"	

Installing tidyverse package

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

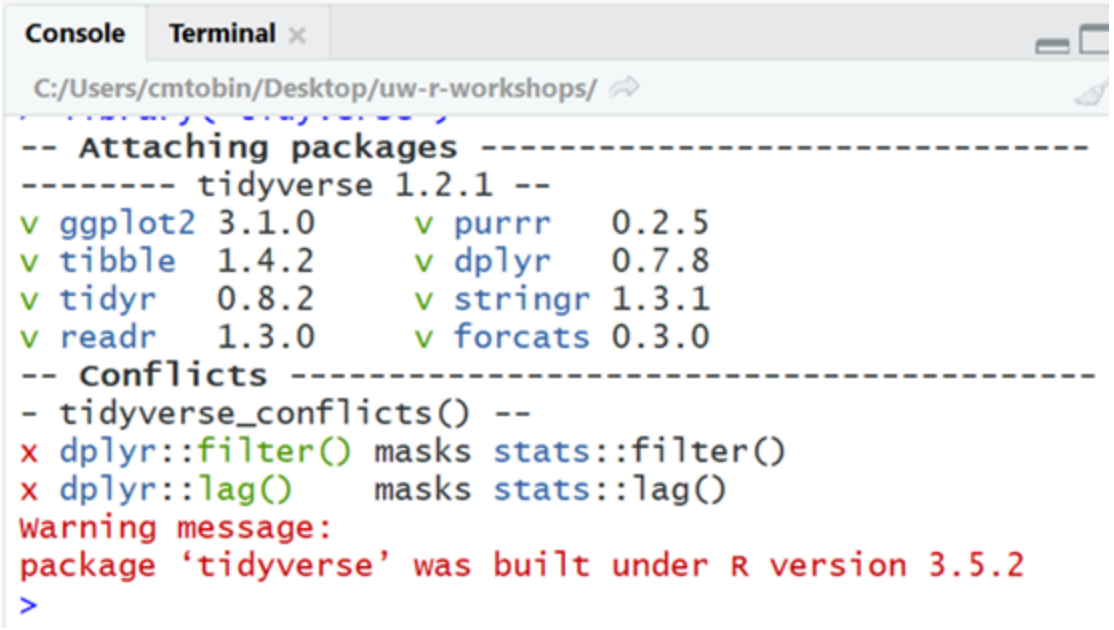
Output:



Loading tidyverse package

```
library("tidyverse")
```

Output:



Let’s load some new data

- We will load the dataset starwars from the tidyverse package

```
library(tidyverse)

data(starwars)
```

Dplyr Example

```
short_characters<- filter(starwars, height<180)
```

- filter() is a dplyr function:
  - First argument is always a dataframe (no vectors)
  - second argument describes which variables to operate on
- Output: a data frame

Pipe operator %>%

- Best practice is to put each function on its own line and indent
- Use %>% to connect your code to the next line
- Say it out loud as “then”

```
short_characters<- starwars %>%
        filter(height<180)
```

Same as

```
short_characters<- filter(starwars, height<180)
```

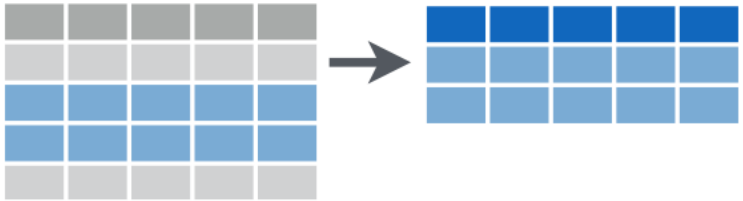
Key dplyr verbs

There are five key dplyr verbs

1. filter: Filter (i.e. subset) rows based on their values.
2. arrange: Arrange (i.e. reorder) rows based on their values.
3. select: Select (i.e. subset) columns by their names:
4. mutate: Create new columns.
5. summarize: Collapse multiple rows into a single summary value

filter()

Subset Observations (Rows)



filter()

- Choose rows based on values of a variable
- **Arguments**: data and a logical expression (returns true/false) **>**, **<**, **>=**, **<=**, **==**, **!=**
- **Output**: data with rows that match the expression

Logical Operators

- Some commonly used logical operators and their descriptions

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to

Combining logical conditions (“and”)

- we use the & logical operator when a case has to meet multiple conditions for the same variable

```
tallish <- starwars %>%
        filter(height >= 190 & height <= 250 )
```

Combining logical conditions (“or”)

- we use the | logical operator when our filter has multiple conditions for the same variable

```
extremes <- starwars %>%
        filter(height <= 160 | height >= 190 )
```

https://pollev.com/vsovero

Exercise

- Find the first and third quartile of the mass variable (try using summarize()
- Filter for characters who fall within those two values

Filtering with non-numeric variables

- If we want to filter based on a non-numeric value, we have to put it in quotes (single or double is fine)

```
humans<- starwars %>%
         filter(species=="Human")
```

Combining logical conditions

- We can also combine logical conditions, same as with numeric variables

```
blue_black<- starwars %>%
        filter(eye_color=="blue" | eye_color=="black")
```

A Useful trick: **%in%** logical operator

- Helpful when you have a longer list of logical conditions for a non-numeric variable

```
droid_humans_hutt <- starwars %>%
        filter(species %in% c( "Human" , "Droid" , "Hutt" ))
```

Exercise

- What’s wrong with these commands?

```
mistake1<- starwars %>%
        filter(species=="human")
```

```
mistake2<- starwars %>%
        filter(species=="Human" | "Droid" )
```

https://pollev.com/vsovero

“not” logical operator

- sometimes it’s easier to filter for what you don’t want
- we use the **!** logical operator

```
not_human<- starwars %>%

            filter(species!="Human")
```

filter() and missing data

- A very common filter use case is identifying (or removing) missing data cases.
- These examples use **is.na** (“is missing”) and **!** (“not”)

```
missing_heights <- starwars %>%
  filter(is.na(height))
```

```
non_missing_heights <- starwars %>%
  filter(!is.na(height))
```

Exercise

- create a data frame of the characters whose sex is listed as none
- create a data frame of the characters whose sex is unknown (missing)

filter()

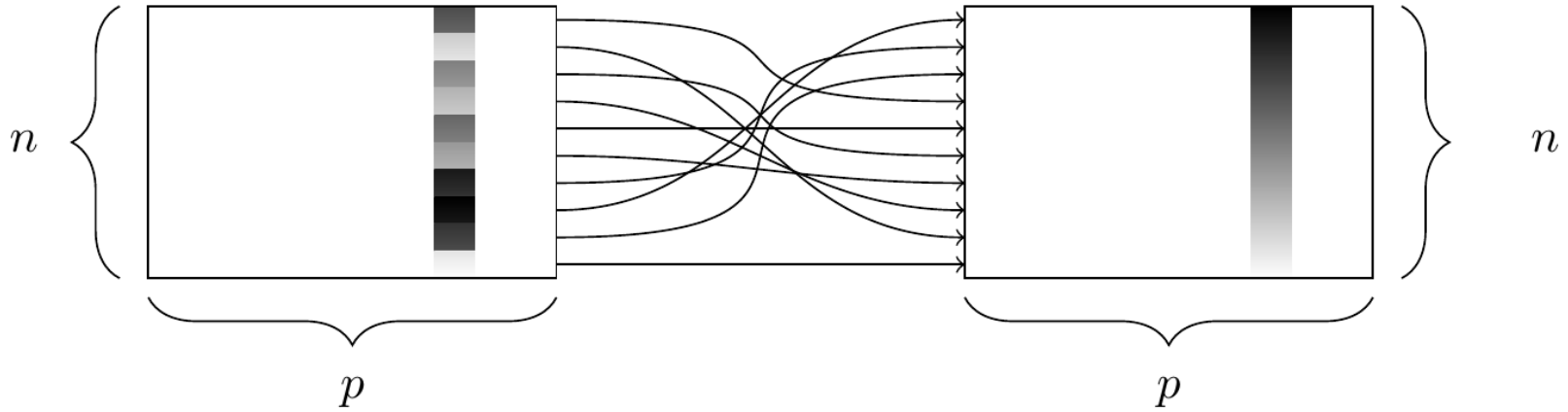
- We can chain multiple filter commands with the pipe %>%
- we can also separate them within a single filter command using commas.

```
tall_humans <- starwars %>%
        filter(species=="Human") %>%
        filter(height >= 190)
```

```
tall_humans <- starwars %>%
        filter(species=="Human", height >= 190)
```

https://pollev.com/vsovero

arrange()



arrange()

- **arrange**() sorts your data frame by a particular variable in ascending order :
  - numeric: 1,2,3, etc.
  - character: a, b, c, etc.

- **Arguments**:
  - A variable
  - Use **desc**() to put them in descending order

```
ascending_birth_year<- starwars %>%
  arrange(birth_year)

descending_birth_year<- starwars %>%
  arrange( desc(birth_year))
```

Exercise

- Who is the tallest character in Naboo?

https://pollev.com/vsovero

select()

Subset Variables (Columns)

select()

- **select**() selects variables from a data frame

```
fewer_vars1<- starwars %>%
              select(mass, species, height)
```

- **Arguments**:
  - variables to be kept (separated by commas)

- **Output**: data with only the specified variables

https://pollev.com/vsovero

select()

- You can also use "first:last" to select consecutive columns.
- Deselect a column with "-".

```
fewer_vars<- starwars %>%
            select(height:eye_color, -skin_color)
```

select()

```
fewer_vars<- starwars %>%
        select(height:eye_color, -skin_color)
```

height	mass	hair_color	skin_color	eye_color	birth_year	sex
172	77.0	blond	fair	blue	19.0	male
167	75.0	*NA*	gold	yellow	112.0	none
96	32.0	*NA*	white, blue	red	33.0	none
202	136.0	none	white	yellow	41.9	male
150	49.0	brown	light	brown	19.0	female
178	120.0	brown, grey	light	blue	52.0	male
165	75.0	brown	light	blue	47.0	female

select() variable names that match a pattern

```
color_vars<- starwars %>%
        select(contains("color"))
```

height	mass	hair_color	skin_color	eye_color	birth_year	sex
172	77.0	blond	fair	blue	19.0	male
167	75.0	*NA*	gold	yellow	112.0	none
96	32.0	*NA*	white, blue	red	33.0	none
202	136.0	none	white	yellow	41.9	male
150	49.0	brown	light	brown	19.0	female
178	120.0	brown, grey	light	blue	52.0	male
165	75.0	brown	light	blue	47.0	female

mutate()

mutate()

- Creates a new variable
- **Arguments:** data frame and the definition of a new variable
- **Output**: data frame with a new variable

mutate()

- You can create multiple new variables at a time with pipes (first example)
- You can also skip the pipes and put a comma instead (second example)

```
add_new_vars<- starwars %>%
        mutate(dog_years=birth_year*7) %>%
          mutate(height_m=height/100)
```

```
add_new_vars<- starwars %>%
        mutate(dog_years=birth_year*7,
                  height_m=height/100)
```

mutate()

- We can also create non-numeric vectors with mutate

```
add_height_vars<- starwars %>%
                  select(name, height) %>%
                  mutate(tall1 = height > 180,
                         tall2 = ifelse(height > 180, "Tall", "Short"))
```

	name	height	tall1	tall2
1	Luke Skywalker	172	FALSE	Short
2	C-3PO	167	FALSE	Short
3	R2-D2	96	FALSE	Short
4	Darth Vader	202	TRUE	Tall
5	Leia Organa	150	FALSE	Short
6	Owen Lars	178	FALSE	Short
7	Beru Whitesun lars	165	FALSE	Short
8	R5-D4	97	FALSE	Short

Be careful with the order of chained commands

- Why doesn't this work?

```
hmm<- starwars %>%
        select(name, mass) %>%
        arrange( height)
```

Exercise

- Calculate the BMI of the star wars characters (height divided by mass squared)
- filter for non-human characters
- sort the data from highest to lowest BMI