Econ 106: Data Analyis for Economics

Lecture 7

slides adapted from: https://r4ds.had.co.nz/tidy-data.html

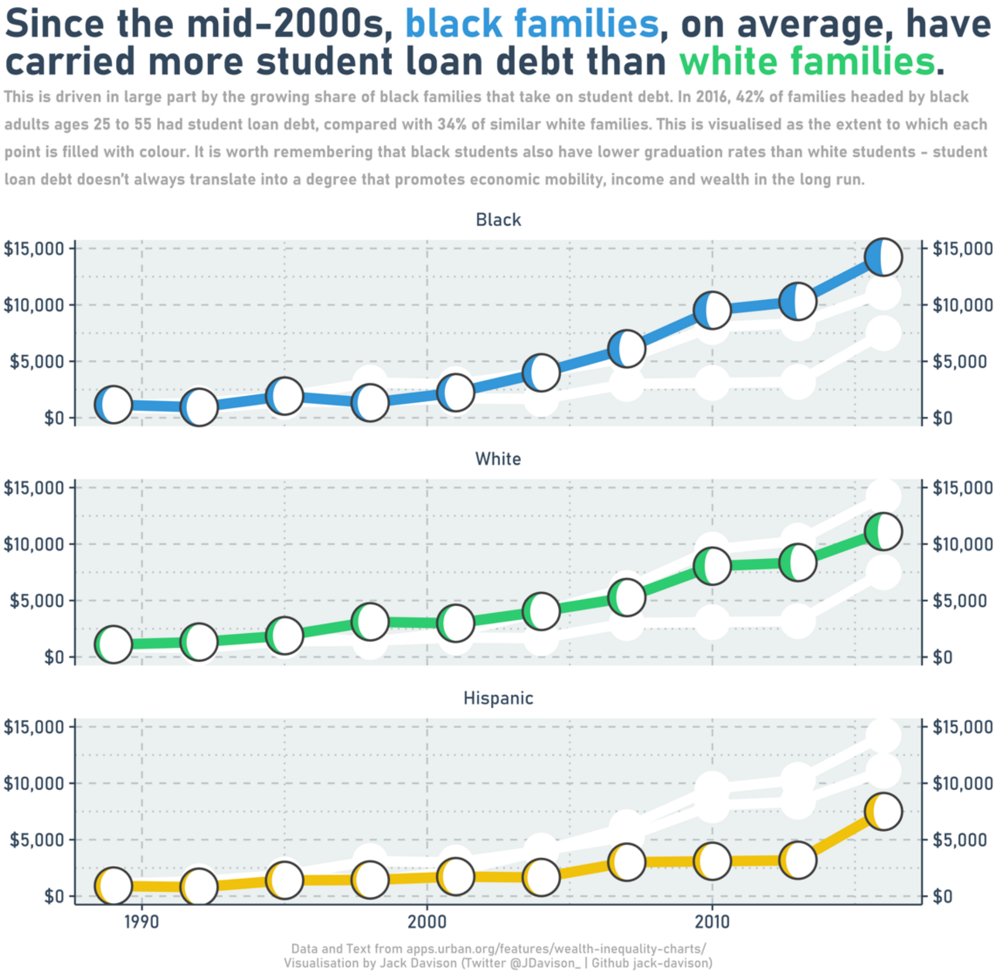
Reminders

- Lab 2 is due Sunday, 11:59pm

https://pollev.com/vsovero

#tidytuesday

- Fun fact: ggplot has a geom_moon (someone please use it for your project!)
- Code is [here]



Outline for Today

- cleaning quantitative variables:
  - removing non-numeric values
  - separate into multiple columns
  - converting to date time objects

Variable Type

Things you should decide about a variable:

- Categorical: values can be non-numeric, have a fixed and known set of possible values
- Quantitative: values are numeric, represent an order and a quantity

This can be different from how the data is stored in R:

- numeric
- character
- factor

Cleaning Quantitative Variables

- Problem: there are character variables that should be numeric (quantitative)

```
coverage                52 obs. of 29 variables
    $ Location          : chr [1:52] "United States" "Alabama" "Alaska" "Arizona" ...
    $ 2013__Employer    : num [1:52] 1.56e+08 2.13e+06 3.65e+05 2.88e+06 1.13e+06 ...
    $ 2013__Non-Group   : num [1:52] 13816000 174200 24000 170800 155600 ...
    $ 2013__Medicaid    : num [1:52] 54919100 869700 95000 1346100 600800 ...
    $ 2013__Medicare    : num [1:52] 40876300 783000 55200 842000 515200 ...
    $ 2013__Other Public: chr [1:52] "6295400" "85600" "60600" "N/A" ...
    $ 2013__Uninsured   : num [1:52] 41795100 724800 102200 1223000 436800 ...
    $ 2013__Total       : num [1:52] 3.13e+08 4.76e+06 7.02e+05 6.60e+06 2.90e+06 ...
```

https://pollev.com/vsovero

Converting Vector Type

- as.numeric(): your values should be numbers
- as.character(): your values can be numbers or characters
- factor(): your values are categorical (finite set of values)

Converting Vector Type (Coercion)

- Sometimes R has to change the values of a vector to accommodate the new type
- Sometimes the value cannot be accommodated, so it gets set to missing (NA)
- Example: "badger" is not numeric, gets erased (NA)

Original Vector	**As character**	**As numeric**
x <- TRUE	"TRUE"	1
y <-35	"35"	35
z <- "badger"	"badger"	NA

Coercion

- We coerce a variable to numeric when it has some non-numeric values
- You will get a warning from R telling you that non-numeric values are being set to missing
- This is fine when the values didn't contain any useable information

Check your data

- Let’s take a look at the data and see which values are non-numeric
- We can see that “N/A” is the culprit
- This is an example where coercion would not removing valid data

	Location	2013__Employer	2013__Non-Group	2013__Medicaid	2013__Medicare	2013__Other Public
1	United States	155696900	13816000	54919100	40876300	6295400
2	Alabama	2126500	174200	869700	783000	85600
3	Alaska	364900	24000	95000	55200	60600
4	Arizona	2883800	170800	1346100	842000	N/A
5	Arkansas	1128800	155600	600800	515200	67600
6	California	17747300	1986400	8344800	3828500	675400
7	Colorado	2852500	426300	697300	549700	118100
8	Connecticut	2030500	126800	532000	475300	48200
9	Delaware	473700	25100	192700	141300	13800
10	District of Columbia	324300	30400	174900	59900	N/A
11	Florida	8023400	968200	3190900	3108800	517800
12	Georgia	4700500	401600	1503000	1280400	334700
13	Hawaii	732600	47900	232600	195000	78600
14	Idaho	802200	114100	233200	194600	21500

Side Note: Variable Names

- In this data set, variables are named in a way that breaks some of our naming rules:
  - shouldn't start with a number
  - shouldn't have spaces
  - shouldn't have special characters

	Location	2013__Employer	2013__Non-Group	2013__Medicaid	2013__Medicare	2013__Other Public
1	United States	155696900	13816000	54919100	40876300	6295400
2	Alabama	2126500	174200	869700	783000	85600
3	Alaska	364900	24000	95000	55200	60600
4	Arizona	2883800	170800	1346100	842000	N/A
5	Arkansas	1128800	155600	600800	515200	67600
6	California	17747300	1986400	8344800	3828500	675400
7	Colorado	2852500	426300	697300	549700	118100
8	Connecticut	2030500	126800	532000	475300	48200
9	Delaware	473700	25100	192700	141300	13800
10	District of Columbia	324300	30400	174900	59900	N/A
11	Florida	8023400	968200	3190900	3108800	517800
12	Georgia	4700500	401600	1503000	1280400	334700
13	Hawaii	732600	47900	232600	195000	78600
14	Idaho	802200	114100	233200	194600	21500

Side Note: Variable Names

- When the variable names “break the rules”, we have to include backticks to reference them in our code

```
large_uninsured_states<- coverage %>%
filter (Location!= "United States") %>%
  filter (`2016__Uninsured`>=1000000)
```

Coverage Example

- We use mutate() to create new variables
- the new variable names are listed in purple
- note that we are recycling the original names and just converting vector type

```
coverage_coerce <- coverage %>%
  mutate(`2016__Other Public`=as.numeric(`2016__Other Public`),
      `2015__Other Public`=as.numeric(`2015__Other Public`),
      `2014__Other Public`=as.numeric(`2014__Other Public`),
      `2013__Other Public`=as.numeric(`2013__Other Public`))
```

Coverage Example

- When there is coercion (values are being converted to NA), you will get a warning message in the console

```
coverage_coerce <- coverage %>%
  mutate(`2016__Other Public`=as.numeric(`2016__Other Public`),
         `2015__Other Public`=as.numeric(`2015__Other Public`),
         `2014__Other Public`=as.numeric(`2014__Other Public`),
         `2013__Other Public`=as.numeric(`2013__Other Public`))
```

```
Warning message:
There were 4 warnings in `mutate()`.
The first warning was:
ℹ In argument: `2016__Other Public = as.numeric(`2016__Other Public`)`.
Caused by warning:
! NAs introduced by coercion
ℹ Run dplyr::last_dplyr_warnings() to see the 3 remaining warnings.
```

Exercise

- examine the end_year variable in the transit_cost data frame. Why is it stored as character?
- convert end_year to numeric

When coercion goes wrong

- There are instances where you convert a variable to numeric and end up wiping out valid data
- the case rate in the table below is a character, but should be numeric (quantitative)

```
table3                      6 obs. of 3 variables
   $ country: chr [1:6] "Afghanistan" "Afghanistan" "Brazil" "Brazil" ...
   $ year   : num [1:6] 1999 2000 1999 2000 1999 ...
   $ rate   : chr [1:6] "745/19987071" "2666/20595360" "37737/172006362" "80488/1745…
```

When coercion goes wrong

```
table3_coercion <- table3 %>%
mutate(rate=as.numeric(rate))
```

	country	year	rate
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
1	Afghanistan	1999	NA
2	Afghanistan	2000	NA
3	Brazil	1999	NA
4	Brazil	2000	NA
5	China	1999	NA
6	China	2000	NA

Splitting values into multiple columns

- In table3, the rate variable needs to be split into two columns

	country	year	rate
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

Separate()

Arguments

1. col: The name of the existing variable whose values you want to split
2. into: The name of new variables where the split values will be moved into
3. sep: The string used to identify where to make the split
4. convert: whether you want the new variables to be converted to numeric

```
table3_separated <- table3%>%
 separate(col=rate,
            into = c("cases", "population"),
            sep = "/”,
           convert=TRUE)
```

Separate()

```
table3_separated <- table3%>%
 separate(col=rate,
                  into = c("cases", "population"),
                  sep = "/",
                 convert=TRUE)
```

	country	year	rate
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	cases	population
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

A Trickier Example

- Population is a quantitative variable, but the commas are making R read it as character
- How do we remove the commas?

	state	population	total	murder_rate
1	Alabama	4,853,875	348	7.2
2	Alaska	737,709	59	8.0
3	Arizona	6,817,565	309	4.5
4	Arkansas	2,977,853	181	6.1
5	California	38,993,940	1,861	4.8
6	Colorado	5,448,819	176	3.2
7	Connecticut	3,584,730	117	3.3
8	Delaware	944,076	63	6.7
9	District of Columbia	670,377	162	24.2
10	Florida	20,244,914	1,041	5.1
11	Georgia	10,199,398	615	6.0
12	Hawaii	1,425,157	19	1.3
13	Idaho	1,652,828	32	1.9
14	Illinois	12,839,047	744	5.8
15	Indiana	6,612,768	373	5.6

String Processing

- The values in a character variable are referred to as *strings*
- One of the most common data wrangling challenges involves extracting numeric data contained in character strings and converting them into numeric

String processing

- String processing tasks can involve:
  - **pattern detection**
  - **extraction based on a pattern**
  - **replacement based on a pattern**

Murders Data

- What's the pattern?
- There are commas in the population values

	state	population	total	murder_rate
1	Alabama	4,853,875	348	7.2
2	Alaska	737,709	59	8.0
3	Arizona	6,817,565	309	4.5
4	Arkansas	2,977,853	181	6.1
5	California	38,993,940	1,861	4.8
6	Colorado	5,448,819	176	3.2
7	Connecticut	3,584,730	117	3.3
8	Delaware	944,076	63	6.7
9	District of Columbia	670,377	162	24.2
10	Florida	20,244,914	1,041	5.1
11	Georgia	10,199,398	615	6.0
12	Hawaii	1,425,157	19	1.3
13	Idaho	1,652,828	32	1.9
14	Illinois	12,839,047	744	5.8
15	Indiana	6,612,768	373	5.6

Murders Data

- What do I want to do based on the pattern?
- Remove the commas in the population values, leave everything else

	state	population	total	murder_rate
1	Alabama	4,853,875	348	7.2
2	Alaska	737,709	59	8.0
3	Arizona	6,817,565	309	4.5
4	Arkansas	2,977,853	181	6.1
5	California	38,993,940	1,861	4.8
6	Colorado	5,448,819	176	3.2
7	Connecticut	3,584,730	117	3.3
8	Delaware	944,076	63	6.7
9	District of Columbia	670,377	162	24.2
10	Florida	20,244,914	1,041	5.1
11	Georgia	10,199,398	615	6.0
12	Hawaii	1,425,157	19	1.3
13	Idaho	1,652,828	32	1.9
14	Illinois	12,839,047	744	5.8
15	Indiana	6,612,768	373	5.6

str_replace_all()

```
murders_clean<-murders_raw%>%
   mutate(new_population=str_replace_all(population,",", ""))
```

Arguments for str_replace_all():

1. the variable that contains strings: population
2. the pattern to look for: a comma
3. the replacement value: nothing

str_replace_all()

```
murders_clean<-murders_raw%>%
  mutate(new_population=str_replace_all(population,",", ""))
```

	state	population	total	murder_rate
1	Alabama	4,853,875	348	7.2
2	Alaska	737,709	59	8.0
3	Arizona	6,817,565	309	4.5
4	Arkansas	2,977,853	181	6.1
5	California	38,993,940	1,861	4.8
6	Colorado	5,448,819	176	3.2
7	Connecticut	3,584,730	117	3.3
8	Delaware	944,076	63	6.7
9	District of Columbia	670,377	162	24.2
10	Florida	20,244,914	1,041	5.1
11	Georgia	10,199,398	615	6.0
12	Hawaii	1,425,157	19	1.3
13	Idaho	1,652,828	32	1.9
14	Illinois	12,839,047	744	5.8
15	Indiana	6,612,768	373	5.6

	state	population	total	murder_rate	new_population
1	Alabama	4,853,875	348	7.2	4853875
2	Alaska	737,709	59	8.0	737709
3	Arizona	6,817,565	309	4.5	6817565
4	Arkansas	2,977,853	181	6.1	2977853
5	California	38,993,940	1,861	4.8	38993940
6	Colorado	5,448,819	176	3.2	5448819
7	Connecticut	3,584,730	117	3.3	3584730
8	Delaware	944,076	63	6.7	944076
9	District of Columbia	670,377	162	24.2	670377
10	Florida	20,244,914	1,041	5.1	20244914
11	Georgia	10,199,398	615	6.0	10199398
12	Hawaii	1,425,157	19	1.3	1425157
13	Idaho	1,652,828	32	1.9	1652828
14	Illinois	12,839,047	744	5.8	12839047
15	Indiana	6,612,768	373	5.6	6612768

But wait, it’s still not numeric

- We use as.numeric() to convert it

```
murders_clean<-murders_raw%>%
mutate(new_population=str_replace_all(population,",", ""),
       numeric_population=as.numeric(new_population))
```

```
murders_clean          51 obs. of 6 variables
  $ state             : chr [1:51] "Alabama" "Alaska" "Arizona" "Arkansas" ...
  $ population        : chr [1:51] "4,853,875" "737,709" "6,817,565" "2,977,853"
  $ total             : chr [1:51] "348" "59" "309" "181" ...
  $ murder_rate       : num [1:51] 7.2 8 4.5 6.1 4.8 3.2 3.3 6.7 24.2 5.1 ...
  $ new_population    : chr [1:51] "4853875" "737709" "6817565" "2977853" ...
  $ numeric_population: num [1:51] 4853875 737709 6817565 2977853 38993940 ...
```

Exercise

- Examine the tunnel_per variable in the transit_cost data frame. Why is it stored as character?
- Convert it to numeric.

https://pollev.com/vsovero

Working with Dates and Times

- Dates and times are a type of quantitative variable
- This date variable is currently stored as numeric, but it is better to convert it to a date object

hectare	shift	date
37F	PM	10142018
37E	PM	10062018
02E	AM	10102018
05D	PM	10182018
39B	AM	10182018
33H	AM	10192018
06G	PM	10202018

Working with Dates and Times

- Things you can do with date objects:
  - extract year, month, day
  - calculate time spans

hectare	shift	date
37F	PM	10142018
37E	PM	10062018
02E	AM	10102018
05D	PM	10182018
39B	AM	10182018
33H	AM	10192018
06G	PM	10202018

Functions to create a Date Object

- Depending on how your date is ordered, select the appropriate function to create a date object:
  - ymd("2017-01-31")
  - mdy("January 31st, 2017")
  - dmy("31-Jan-2017")
- You can also create a date-time object:
  - ymd_hms("2017-01-31 20:11:59")
  - mdy_hm("01/31/2017 08:01")

Converting to Dates

- We will use the mdy() function, which comes from the lubridate package inside the tidyverse
  - Arguments: the variable you want to convert to a date object
  - Output: a variable that is a date type

```
squirrels_date<-nyc_squirrels%>%
  mutate(date_converted=mdy(date))
```

Converting to Dates

```
squirrels_date<-
nyc_squirrels%>%

mutate(date_converted=mdy(date))
```

```
$ city_council_districts      : num [1:3023] 19 19 19 19 19 19 19 19 19 19 ...
$ police_precincts            : num [1:3023] 13 13 13 13 13 13 13 13 13 13 ...
$ date_converted              : Date[1:3023], format: "2018-10-14" "2018-10-06" "…
```

Pulling out components from a Date

- Once you have a date or date object, you can pull out individual parts of the date, such as:
  - year()
  - month()
  - mday() : day of the month
  - yday(): day of the year
  - wday(): day of the week
- For date-times, you can additionally pull out:
  - hour()
  - minute()
  - second()

Pulling out components from a Date

- Arguments:
  - the date variable you want to pull information from(date_converted)
  - label: whether you want the new variable to be numeric or factor

```
squirrel_month<-squirrel_date%>%
  mutate(month_factor=month(date_converted, label=TRUE),
         month_num=month(date_converted, label=FALSE))
```

Pulling out components from a Date

```
squirrel_month<-squirrel_date%>%
   mutate(month_factor=month(date_converted, label=TRUE),
           month_num=month(date_converted, label=FALSE))
```

date_converted	month_cat	month_num
2018-10-14	Oct	10
2018-10-06	Oct	10
2018-10-10	Oct	10
2018-10-18	Oct	10
2018-10-18	Oct	10
2018-10-19	Oct	10
2018-10-20	Oct	10
2018-10-13	Oct	10
2018-10-08	Oct	10
2018-10-17	Oct	10

https://pollev.com/vsovero

Exercise

- Pull out day of the week as a factor variable
- count the number of squirrel sightings by day of the week, filled by time of day (shift)