Econ 106
Data Analysis in Economics

Fall 2024
Lecture 2

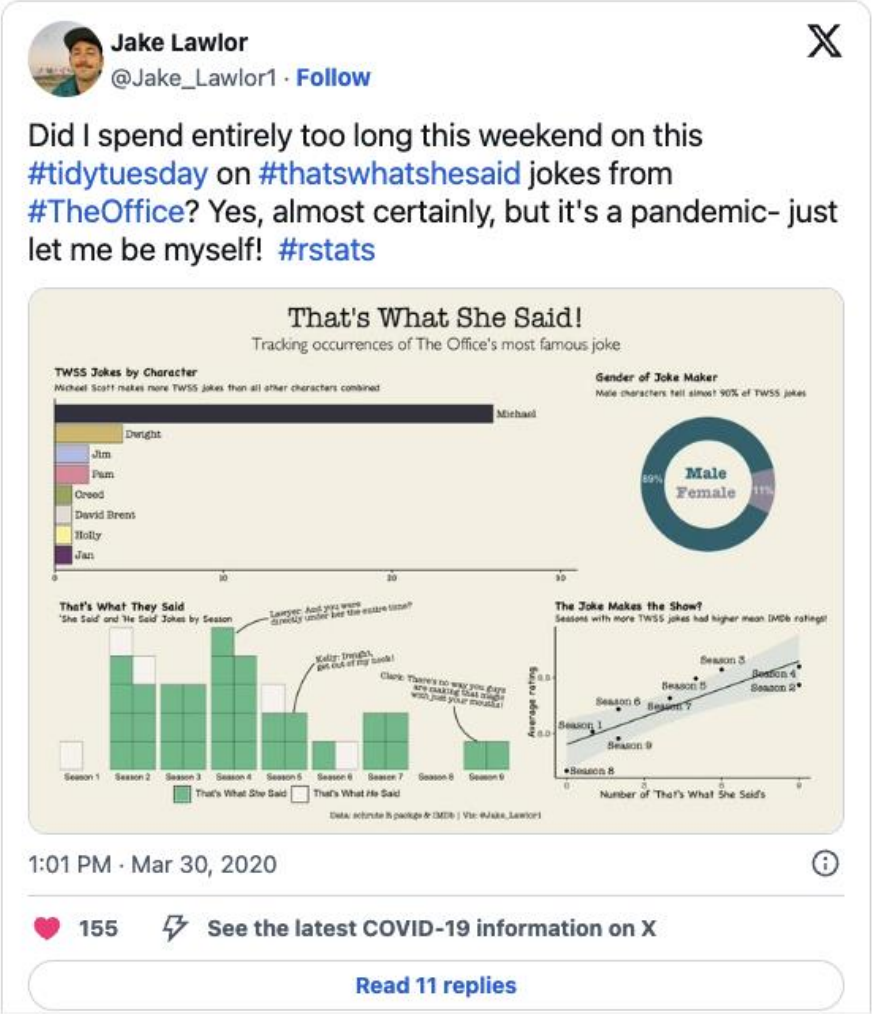
Based on: http://www.datacarpentry.org/R-ecology-lesson/

Reminders

- Poll Everywhere is practice this week, next week counts towards your grade
- Please remember to log in at the beginning of each lecture:
- PollEv.com/vsovero

#tidytuesday

- weekly data challenge using R
- every Tuesday a new challenge is posted here
- The data used in this example can be found here
- you can browse the results on social media (#tidytuesday)
- Good source of inspiration for your projects



Outline

- Vectors
- Packages
- Data Frames

Objects

- **Objects** store data(information)
- We use the assignment operator **<-** to save values into objects

```
weight_kg <- 55
```

Objects: Not Just for Storing Numbers

- Basic Data types:
  - Character
  - Numeric

	type
x <- 32	numeric
y <- “hi”	character

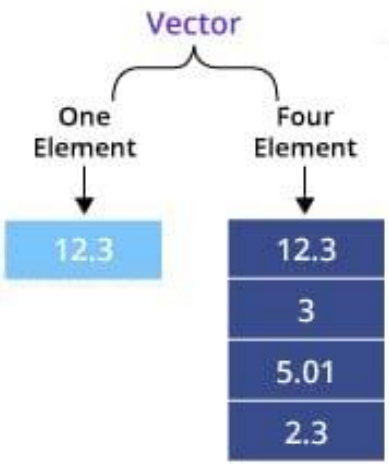
Data structures

- So far, we've been dealing with rather simple data:
  - one value stored in one object
- R has more complex data structures that make working with more complex data easier:
  - **Vectors**: multiple values of the same type.
  - **Data frame**: multiple **vectors** of the same length grouped as columns.

Vectors: Objects with Multiple Values

- R can store multiple values of the same type in a data type called a vector.
- An object with a single element is actually a vector (length 1):

```
x <- 12.3
```



Creating Objects With Multiple Values

- Just like other objects, vectors are assigned values using the assignment operator
- To assign multiple values to the object, we use the combine function **c**()
- Each value is separated by a comma

```
#Example: a vector of ages
age<-c(30, 40, 25, 22)
```

› PollEv.com/vsovero

Creating Objects With Multiple Values

- Combine function **c**() :
  - **Input**: values separated by commas
  - **Output**: a vector object
  - All the same data type

```
#Example: a vector of ages
age<-c(30, 40, 25, 22)
```

Creating Objects With Multiple Values

- for character vectors, use straight quotes around the values
- Single or double is fine

```
#Example: a character vector of favorite animals
animals<-c("cat", "dog", "dog", "dog")

#This also works
animals<-c('cat', 'dog', 'dog', 'dog')
```

Functions for inspecting vector type

- R automatically records what type of data is stored in an object
- **Helpful functions:**
  - **class**()
  - **typeof**()

	class( )	**typeof**( )
x <- 32	numeric	double
y <- "hi"	character	character

Class Coercion

- A vector can't mix and match types, so R will choose a type that applies to most values (class coercion)
- Sometimes this choice isn’t obvious (usually character)
- Use **class**() to see what R picked

```
#What types are these vectors?

first_vector<- c(1, 2, 3, "a" )
second_vector <- c(1, 2, 3, "4" )
```

Summarizing Numeric Vectors

- **Numeric vector:** you can compute the mean

```
#A vector of ages
age<-c(30, 40, 25, 22)

#What happens when you try to compute the mean?
mean(age)
```

Helpful functions for summarizing numeric vectors

- mean()
- median()
- max()
- min()
- sd()
- summary()

Summarizing character vectors?

- **character vector:** you will get warning message if you try to compute the mean, median, min, max ()
- It will return a value of NA (how R records missing values)

```
#a character vector of favorite animals
animals<-c("cat", "dog", "dog", "dog")

#What happens when you try to compute the mean?
mean(animals)
```

› PollEv.com/vsovero

Factors

- We need to convert the animal vector into a **factor,** which is R's way of storing categorical data (variables with a finite set of categories)
- we use the **factor**() function to create a factor variable from the animals vector

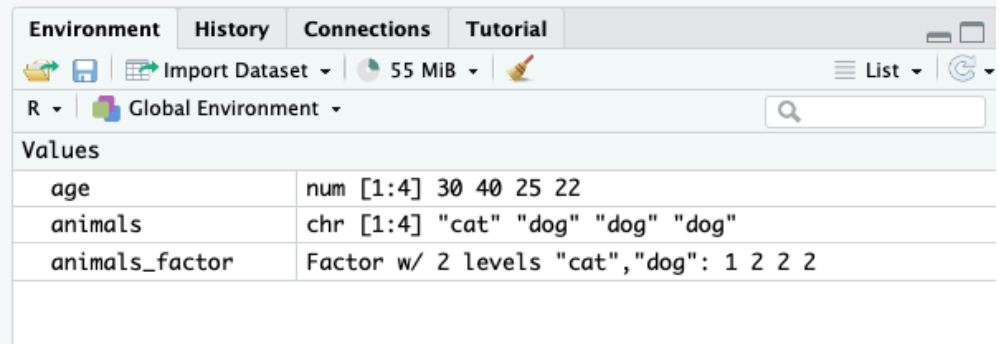
```
#a character vector of favorite animals
animals<-c("cat", "dog", "dog", "dog")

#let's convert animals into a factor variable

animals_factor<-factor(animals)
```

Factors

- You can already see that there's more useful information when animals is stored as a factor instead of a character vector



Summarizing Factors

- We can now calculate summary statistics more appropriate for categorical data

Input	Output
levels(animals_factor)	"cat" "dog"
nlevels(animals_factor)	2
summary(animals_factor)	cat dog 1 3

Data structures: Vectors to Data Frames

- When we have multiple pieces of information from the same people, it’s more useful to have the information stored as a single table vs individual vectors

vector

30
40
25
22

vector

cat
dog
dog
dog

data frame

30	cat
40	dog
25	dog
22	dog

Data structures: Vectors to Data Frames

- This will allow us to do thing such as:
  - calculate average age for cat people
  - calculate average age for dog people

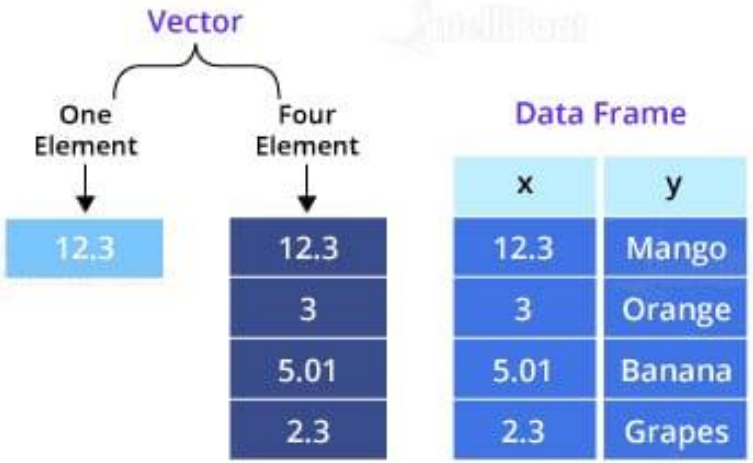
- We'll learn how to do these things next week

data frame

30	cat
40	dog
25	dog
22	dog

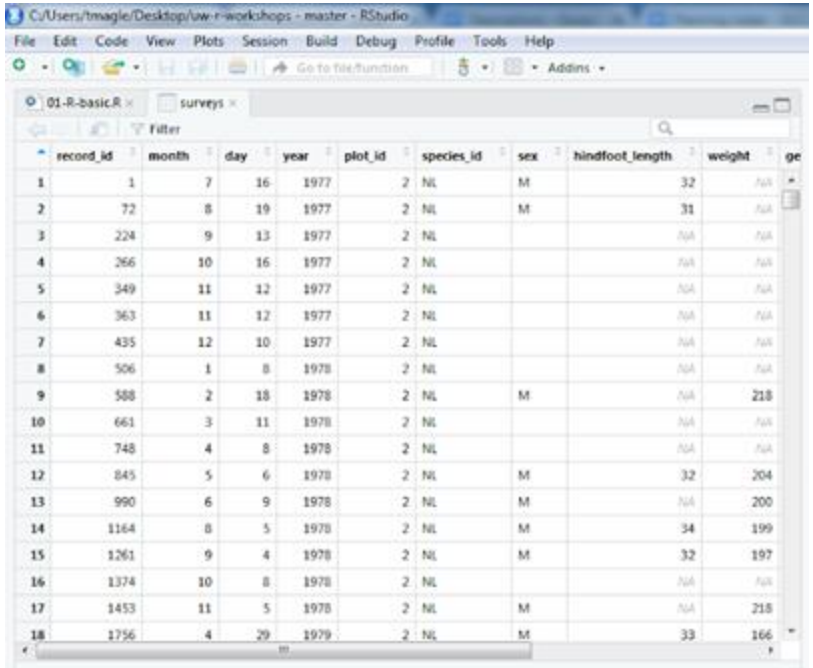
Data structures: Vectors to Data Frames

- Vectors are the building blocks of Data Frames



Data frames

- **Rows** = observations
- **Columns** = variables
- All values in a column must be the same data type.
- All columns must have same number of rows.



	record_id	month	day	year	plot_id	species_id	sex	hindfoot_length	weight	ge
1	1	7	16	1977	2	NL	M	32	NA	
2	72	8	19	1977	2	NL	M	31	NA	
3	224	9	13	1977	2	NL		NA	NA	
4	266	10	16	1977	2	NL		NA	NA	
5	349	11	12	1977	2	NL		NA	NA	
6	363	11	12	1977	2	NL		NA	NA	
7	435	12	10	1977	2	NL		NA	NA	
8	506	1	8	1978	2	NL		NA	NA	
9	588	2	18	1978	2	NL	M	NA	218	
10	661	3	11	1978	2	NL		NA	NA	
11	748	4	8	1978	2	NL		NA	NA	
12	845	5	6	1978	2	NL	M	32	204	
13	990	6	9	1978	2	NL	M	NA	200	
14	1164	8	5	1978	2	NL	M	34	199	
15	1261	9	4	1978	2	NL	M	32	197	
16	1374	10	8	1978	2	NL		NA	NA	
17	1453	11	5	1978	2	NL	M	NA	218	
18	1756	4	29	1979	2	NL	M	33	166	

Let’s check out a data frame!

- we will use “practice data” for now
- we will learn to import “real data” later on in the quarter

Packages

- Many of the most useful functions of R come from add-on **packages.**
- Packages can include:
  - reusable functions
  - the documentation that describes how to use them
  - sample data
- Some are well maintained, and others are not (user beware)

Packages Example

- Think of packages as apps for your phone
- Install it once, and “open” it when you want to use a package in your script
- Important: you need to load the package each time you open RStudio

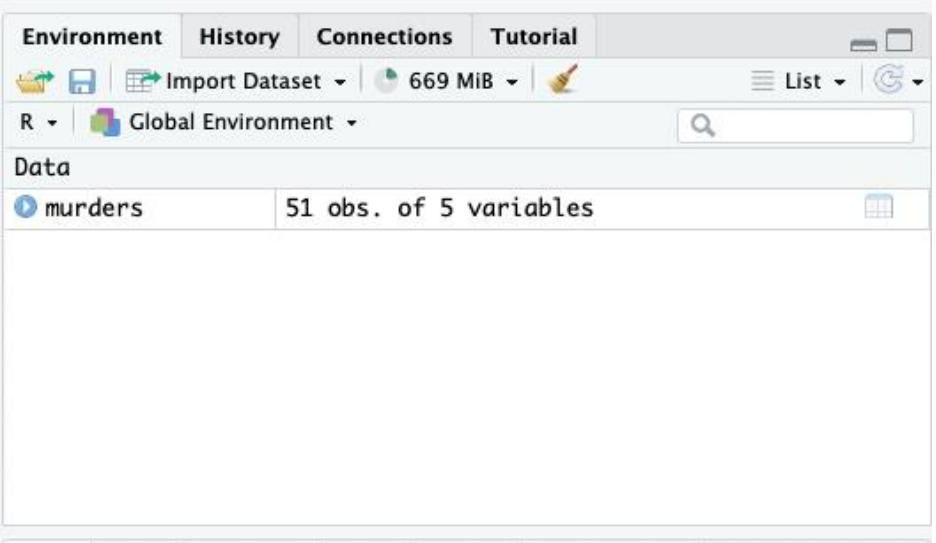
```
#run this once
install.packages("dslabs")

#run this every time you open RStudio
library(dslabs)
```

Let’s check out a data frame!

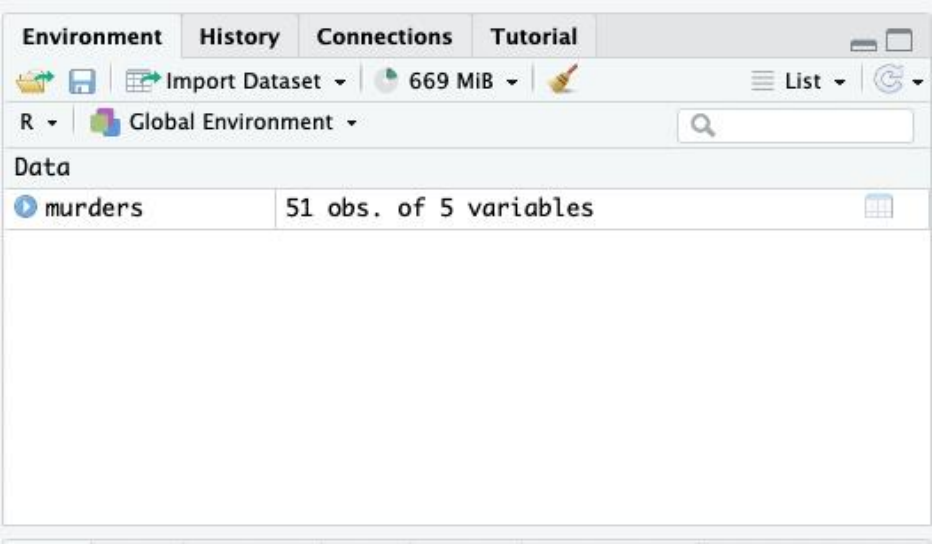
- Load the dataset murders from the dslabs package:

```
library(dslabs)
data(murders)
```



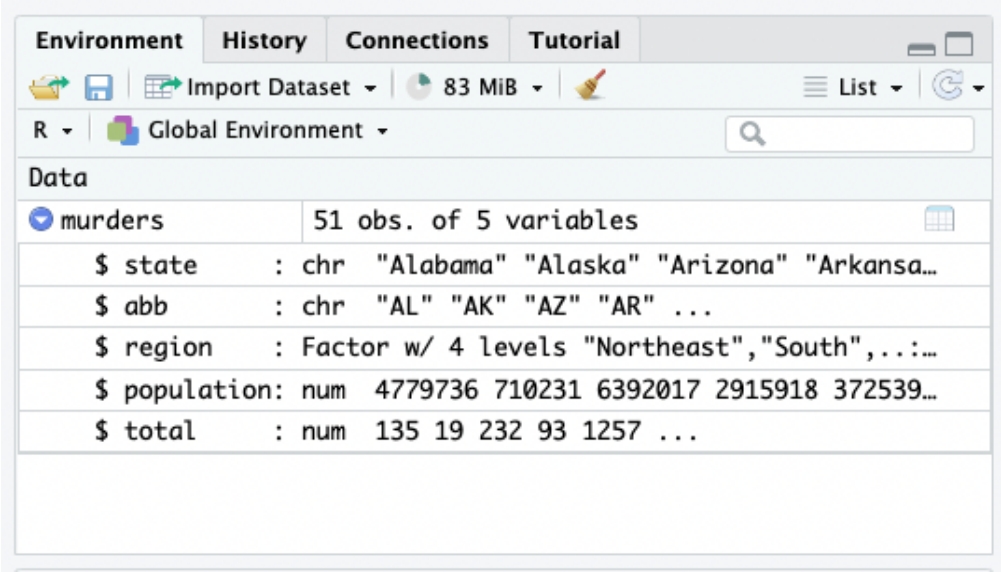
Let’s check out a data frame!

- You can already see some information about the data frame:
  - 51 observations
  - 5 variables



Let’s check out a data frame!

- If you press the blue “play button” to the left of the name, you will get more detailed information on the variables



Let’s check out a data frame!

- Same information gets sent to the console if you examine its **str**ucture with the str() function:

  str(murders)

Console Terminal Background Jobs

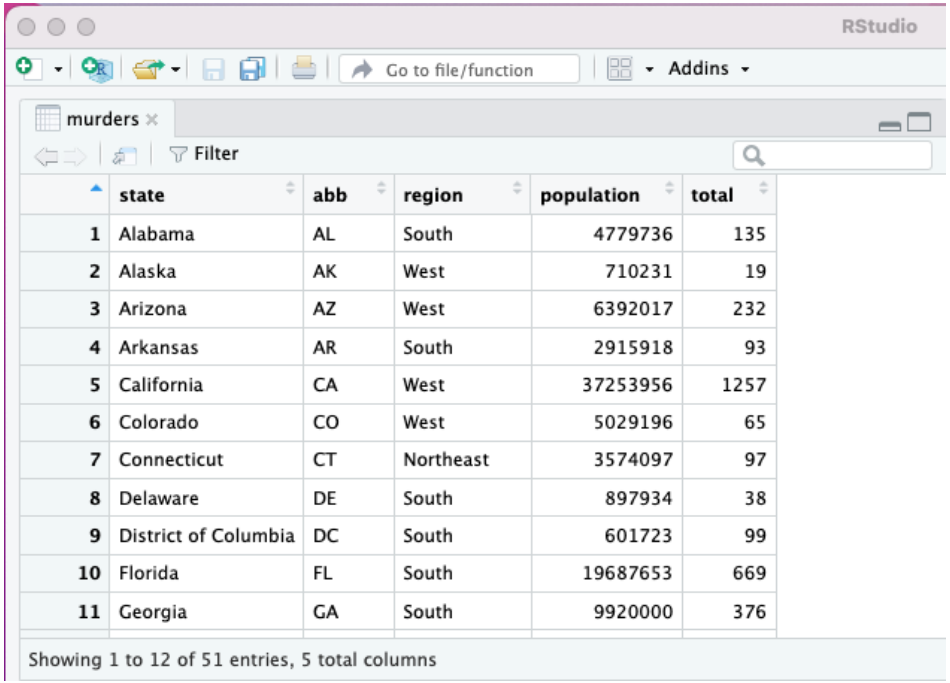
R 4.2.2 · ~/Dropbox/Teaching/Data analytics/spatial data 2/

```
> class(murders)
[1] "data.frame"
> str(murders)
'data.frame':   51 obs. of  5 variables:
 $ state     : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
 $ abb       : chr  "AL" "AK" "AZ" "AR" ...
 $ region    : Factor w/ 4 levels "Northeast","South",..: 2 4 4 2 4 4 1 2 2 2
...
 $ population: num  4779736 710231 6392017 2915918 37253956 ...
 $ total     : num  135 19 232 93 1257 ...
>
```

Let’s check out a data frame!

- Click on the dataframe in your Environment pane to view the dataset in it’s entirety
- You can also type:

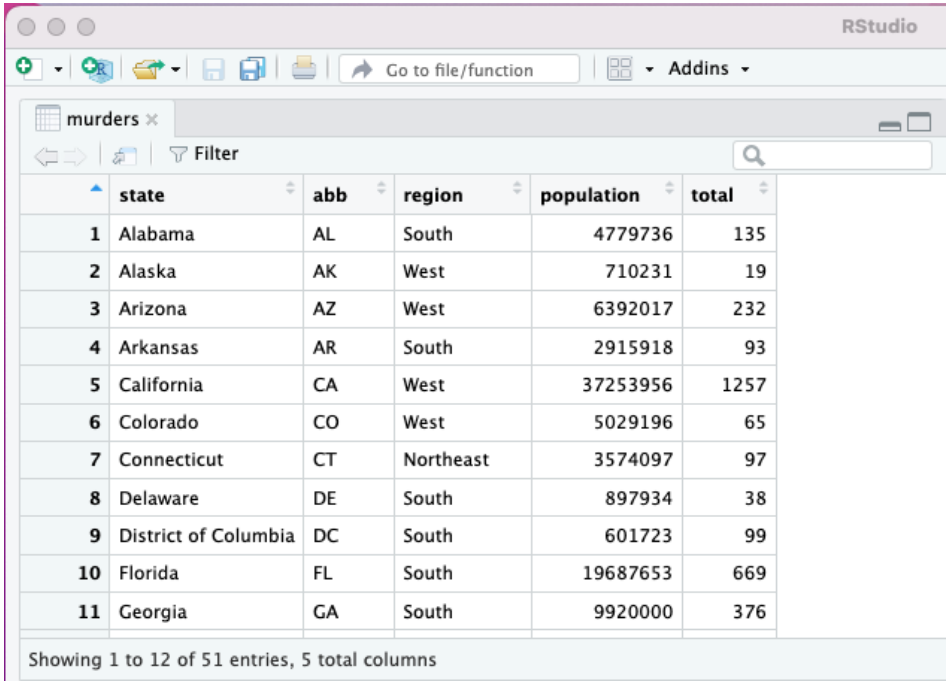
  view(murders)



	state	abb	region	population	total
1	Alabama	AL	South	4779736	135
2	Alaska	AK	West	710231	19
3	Arizona	AZ	West	6392017	232
4	Arkansas	AR	South	2915918	93
5	California	CA	West	37253956	1257
6	Colorado	CO	West	5029196	65
7	Connecticut	CT	Northeast	3574097	97
8	Delaware	DE	South	897934	38
9	District of Columbia	DC	South	601723	99
10	Florida	FL	South	19687653	669
11	Georgia	GA	South	9920000	376

Let’s check out a data frame!

- The data viewer provides a helpful first look at your data (you should always visually inspect your data)
- We can also apply some easy point and click tools to sort/filter in the viewer



	state	abb	region	population	total
1	Alabama	AL	South	4779736	135
2	Alaska	AK	West	710231	19
3	Arizona	AZ	West	6392017	232
4	Arkansas	AR	South	2915918	93
5	California	CA	West	37253956	1257
6	Colorado	CO	West	5029196	65
7	Connecticut	CT	Northeast	3574097	97
8	Delaware	DE	South	897934	38
9	District of Columbia	DC	South	601723	99
10	Florida	FL	South	19687653	669
11	Georgia	GA	South	9920000	376

Showing 1 to 12 of 51 entries, 5 total columns

Sorting in the data viewer

- you can sort by any column by just by clicking on the column
- Click on a column that's already sorted to reverse the sort direction

Lecture 1.R × | Lecture 2.R* × | murders ×

Filter

	state	abb	region	population	total	murder_rates
51	Wyoming	WY	West	563626	5	0.8871131
9	District of Columbia	DC	South	601723	99	16.4527532
46	Vermont	VT	Northeast	625741	2	0.3196211
35	North Dakota	ND	North Central	672591	4	0.5947151
2	Alaska	AK	West	710231	19	2.6751860
42	South Dakota	SD	North Central	814180	8	0.9825837
8	Delaware	DE	South	897934	38	4.2319369
27	Montana	MT	West	989415	12	1.2128379
40	Rhode Island	RI	Northeast	1052567	16	1.5200933
30	New Hampshire	NH	Northeast	1316470	5	0.3798036
20	Maine	ME	Northeast	1328361	11	0.8280881
12	Hawaii	HI	West	1360301	7	0.5145920
13	Idaho	ID	West	1567582	12	0.7655102
28	Nebraska	NE	North Central	1826341	32	1.7521372

Showing 1 to 14 of 51 entries, 6 total columns

Filtering in the data viewer

- To apply filters, click the Filter icon in the toolbar
- Any field that can be filtered will have a white box labeled *All*

Lecture 1.R × | Lecture 2.R* × | murders ×

Filter

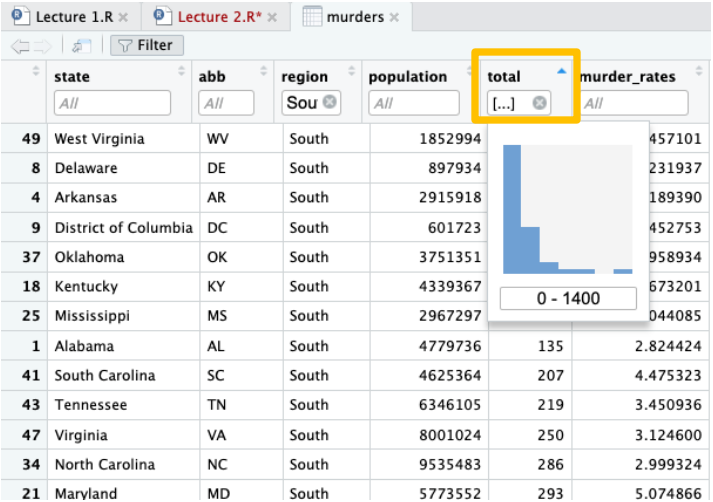
	state	abb	region	population	total	murder_rates
	All	All	Sou	All	All	All
44	Texas	TX	South	25145561	805	3.201360
10	Florida	FL	South	19687653	669	3.398069
11	Georgia	GA	South	9920000	376	3.790323
19	Louisiana	LA	South	4533372	351	7.742581
21	Maryland	MD	South	5773552	293	5.074866
34	North Carolina	NC	South	9535483	286	2.999324
47	Virginia	VA	South	8001024	250	3.124600
43	Tennessee	TN	South	6346105	219	3.450936
41	South Carolina	SC	South	4625364	207	4.475323
1	Alabama	AL	South	4779736	135	2.824424
25	Mississippi	MS	South	2967297	120	4.044085
18	Kentucky	KY	South	4339367	116	2.673201
37	Oklahoma	OK	South	3751351	111	2.958934

Showing 1 to 13 of 17 entries, 6 total columns (filtered from 51 total entries)

› PollEv.com/vsovero

Filtering in the data viewer

- If you click on a numeric variable, a histogram will pop up where you can select the values you want to filter for



	state	abb	region	population	total	murder_rates
49	West Virginia	WV	South	1852994		457101
8	Delaware	DE	South	897934		231937
4	Arkansas	AR	South	2915918		189390
9	District of Columbia	DC	South	601723		452753
37	Oklahoma	OK	South	3751351		958934
18	Kentucky	KY	South	4339367		673201
25	Mississippi	MS	South	2967297		044085
1	Alabama	AL	South	4779736	135	2.824424
41	South Carolina	SC	South	4625364	207	4.475323
43	Tennessee	TN	South	6346105	219	3.450936
47	Virginia	VA	South	8001024	250	3.124600
34	North Carolina	NC	South	9535483	286	2.999324
21	Maryland	MD	South	5773552	293	5.074866

Filtering in the data viewer

- Filtering/sorting in the data viewer does not make any permanent changes to the murders data frame
- For example, if you reopen the viewer after filtering for the South, all of the regions will still be there

Lecture 1.R | Lecture 2.R* | murders

Filter

	state	abb	region	population	total	murder_rates
	All	All	Sou	All	All	All
44	Texas	TX	South	25145561	805	3.201360
10	Florida	FL	South	19687653	669	3.398069
11	Georgia	GA	South	9920000	376	3.790323
19	Louisiana	LA	South	4533372	351	7.742581
21	Maryland	MD	South	5773552	293	5.074866
34	North Carolina	NC	South	9535483	286	2.999324
47	Virginia	VA	South	8001024	250	3.124600
43	Tennessee	TN	South	6346105	219	3.450936
41	South Carolina	SC	South	4625364	207	4.475323
1	Alabama	AL	South	4779736	135	2.824424
25	Mississippi	MS	South	2967297	120	4.044085
18	Kentucky	KY	South	4339367	116	2.673201
37	Oklahoma	OK	South	3751351	111	2.958934

Showing 1 to 13 of 17 entries, 6 total columns (filtered from 51 total entries)

Functions for Inspecting data frames

Function	Output
class*	Class of the object
str*	structure: # rows, cols, data types
head/tail*	look at first/last 6 rows (all columns)
nrow/ncol	number of rows/columns
names	column names
summary*	summary stats for each column (min, max, etc.)

* Also inspects vectors

Summary Statistics

- Display some summary statistics with summary() function:

summary(murders)

Console | Terminal | Background Jobs

R 4.2.2 · ~/Dropbox/Teaching/Data analytics/spatial data 2/

```
> summary(murders)
    state               abb                     region
 Length:51          Length:51          Northeast     : 9
 Class :character   Class :character   South         :17
 Mode  :character   Mode  :character   North Central:12
                                       West          :13


   population           total
 Min.   :  563626   Min.   :   2.0
 1st Qu.: 1696962   1st Qu.:  24.5
 Median : 4339367   Median :  97.0
 Mean   : 6075769   Mean   : 184.4
 3rd Qu.: 6636084   3rd Qu.: 268.0
 Max.   :37253956   Max.   :1257.0
>
```

Summary Statistics (numeric)

- summary stats for numeric vectors:
  - min, max
  - quartiles
  - mean

Console | Terminal | Background Jobs

R 4.2.2 · ~/Dropbox/Teaching/Data analytics/spatial data 2/

```
> summary(murders)
    state               abb                     region
 Length:51          Length:51          Northeast     : 9
 Class :character   Class :character   South         :17
 Mode  :character   Mode  :character   North Central:12
                                       West          :13

  population          total
 Min.   :  563626   Min.   :   2.0
 1st Qu.: 1696962   1st Qu.:  24.5
 Median : 4339367   Median :  97.0
 Mean   : 6075769   Mean   : 184.4
 3rd Qu.: 6636084   3rd Qu.: 268.0
 Max.   :37253956   Max.   :1257.0
>
```

Summary Statistics (factor)

- summary stats for factor variables:
  - levels (categories)
  - frequencies of each level

Console | Terminal | Background Jobs

R 4.2.2 · ~/Dropbox/Teaching/Data analytics/spatial data 2/

```
> summary(murders)
    state               abb                 region
 Length:51          Length:51          Northeast     : 9
 Class :character   Class :character   South         :17
 Mode  :character   Mode  :character   North Central:12
                                       West          :13

   population           total
 Min.   :  563626   Min.   :   2.0
 1st Qu.: 1696962   1st Qu.:  24.5
 Median : 4339367   Median :  97.0
 Mean   : 6075769   Mean   : 184.4
 3rd Qu.: 6636084   3rd Qu.: 268.0
 Max.   :37253956   Max.   :1257.0
>
```

Summary Statistics (character)

- summary stats for character vectors: not much to report
- This is also why it's better to store categorical data as a <u>factor</u>

Console | Terminal | Background Jobs

R 4.2.2 · ~/Dropbox/Teaching/Data analytics/spatial data 2/

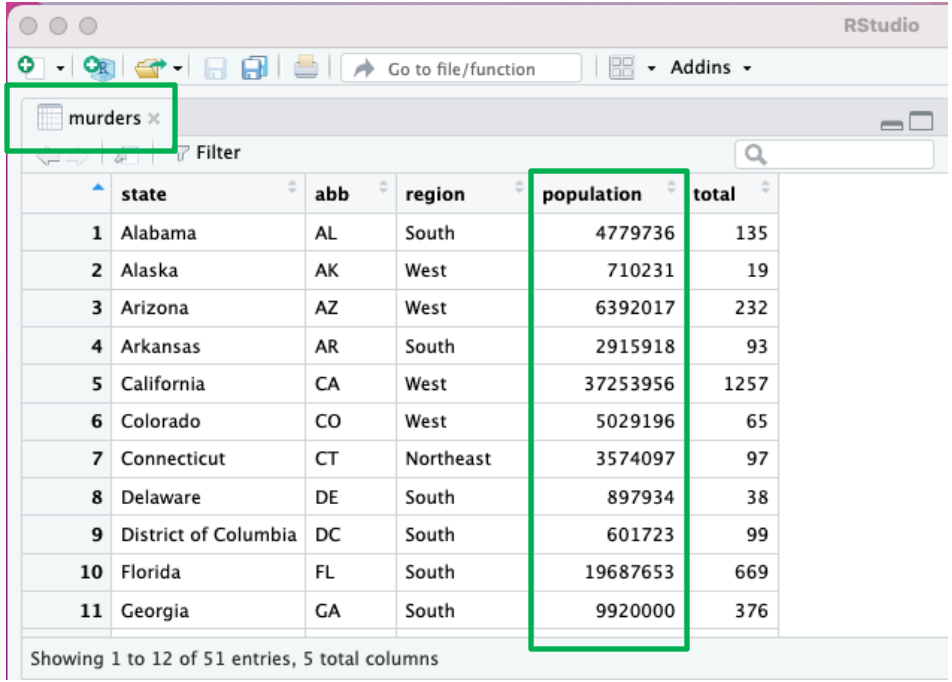
```
> summary(murders)
    state              abb                 region
 Length:51          Length:51          Northeast     : 9
 Class :character   Class :character   South         :17
 Mode  :character   Mode  :character   North Central:12
                                       West          :13


   population           total
 Min.   :  563626   Min.   :   2.0
 1st Qu.: 1696962   1st Qu.:  24.5
 Median : 4339367   Median :  97.0
 Mean   : 6075769   Mean   : 184.4
 3rd Qu.: 6636084   3rd Qu.: 268.0
 Max.   :37253956   Max.   :1257.0
>
```

Referencing Vectors in a Data Frame

- We use the accessor operator $ to reference a vector in a data frame:

  murders$population



	state	abb	region	population	total
1	Alabama	AL	South	4779736	135
2	Alaska	AK	West	710231	19
3	Arizona	AZ	West	6392017	232
4	Arkansas	AR	South	2915918	93
5	California	CA	West	37253956	1257
6	Colorado	CO	West	5029196	65
7	Connecticut	CT	Northeast	3574097	97
8	Delaware	DE	South	897934	38
9	District of Columbia	DC	South	601723	99
10	Florida	FL	South	19687653	669
11	Georgia	GA	South	9920000	376

Showing 1 to 12 of 51 entries, 5 total columns

Example: Check vector type in a data frame

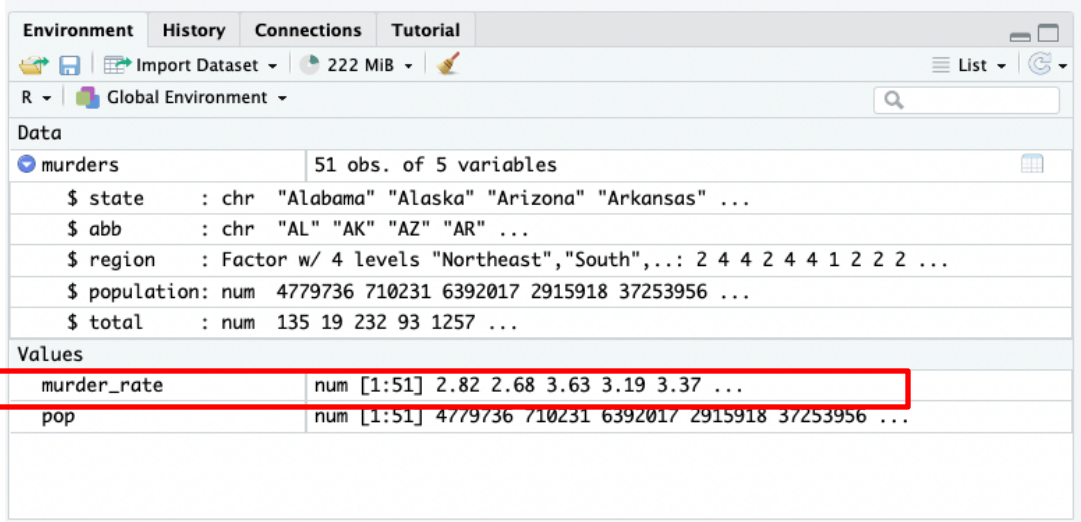
```
class(murders$population)
```

- You can use the accessor operator **$** combined with class() to check the vector type of population

Creating new vectors with $

- we can create a new vector (murder_rate) by referencing vectors in the murders data frame

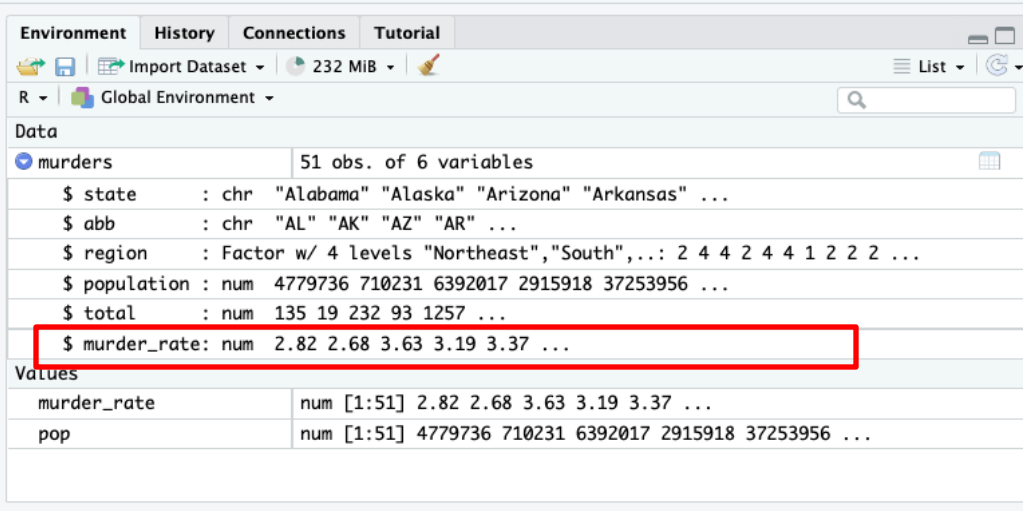
murder_rate <- murders $ total /murders $ population * 100000



Adding a vector to a Data Frame with $

```
murders$murder_rate <- murders$total /murders$population*100000
```

- It's more useful to add the new vector to the murder data frame



› PollEv.com/vsovero