Data

Data has to come from somewhere.

Terms:

- **Census**: Information on <u>**all**</u> subjects.
- **Subset** or **sample**: Information on some subjects.
- The subset of subjects included in a sample can depend on many factors:
  - **Self-selected sample**: Subset is whoever chooses to answer.
  - **Convenience sample:** Subset is whomever is convenient for researcher.
  - **Judgment sample:** Subset is whomever researcher deliberately selects.
  - **Random sample**: Subset involves some probabilistic selection.
- **Administrative dataset**: A dataset collected as part of administrative work (e.g. social security names, restaurant safety ratings, etc.).

Data

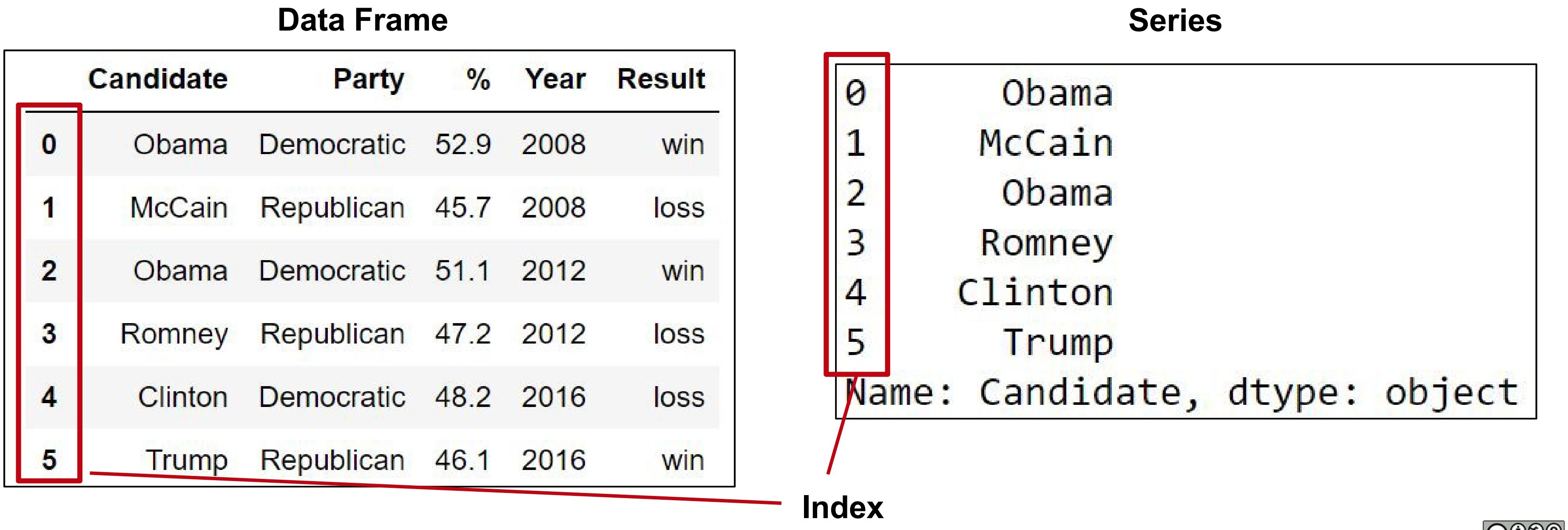
Types of probability samples:

- Simple random sample: “sample drawn from a population at random without replacement.”
- Stratified sample: Divide population into strata, then create one SRS per strata.
    - Kasey from Boston.
    - Gurmeet from New York.
    - Martie from Paris.
    - … (and so on for every single city).
- Cluster sampling: Divide population into clusters. Then take an SRS where each sample is an entire cluster.
    - Everyone from Boston.
    - Everyone from Tokyo.

Pandas Data Structures

There are three fundamental data structures in pandas:

- Data Frame: 2D data tabular data.
- Series: 1D data. I usually think of it as columnar data.
- Index: A sequence of row labels.



Data Frame

	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

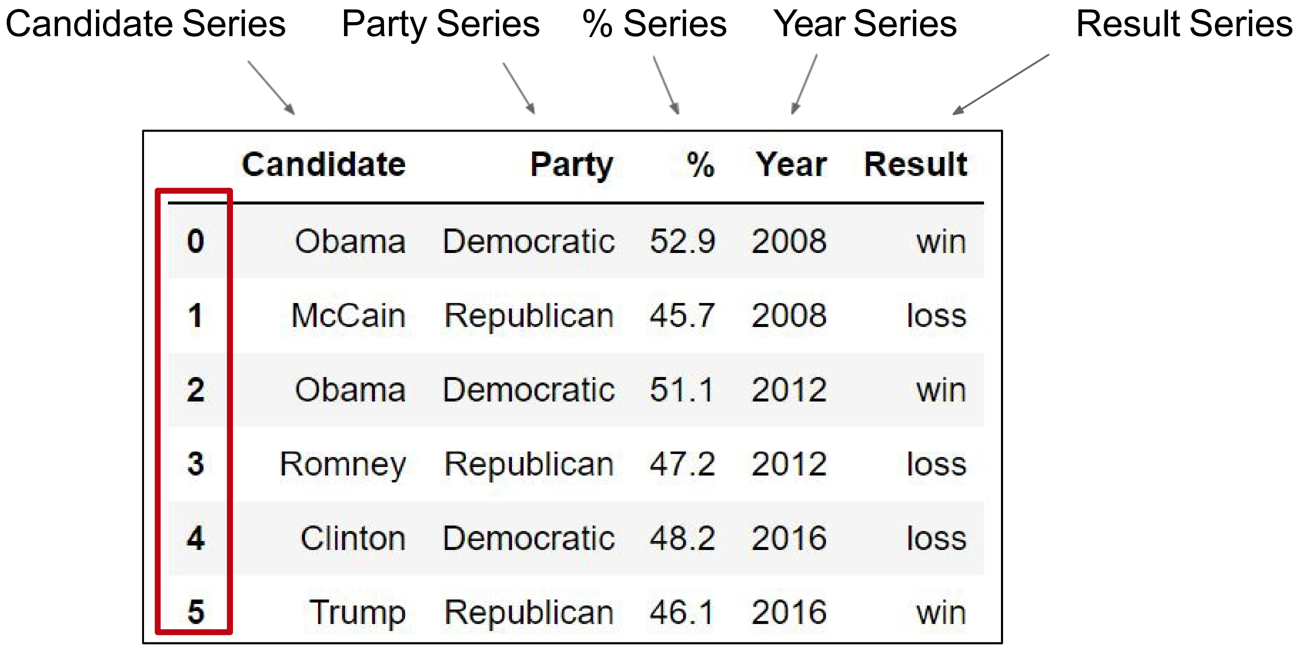
Series

```
0        Obama
1       McCain
2        Obama
3       Romney
4      Clinton
5        Trump
Name: Candidate, dtype: object
```

The Relationship Between Data Frames, Series, and Indices

Can think of a Data Frame as a collection of Series that all share the same Index.

- Candidate, Party, %, Year, and Result Series all share an index from 0 to 5.



Candidate Series

Party Series

% Series

Year Series

Result Series

	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

The Relationship Between Data Frames, Series, and Indices

Most common tools for accessing elements of a Data Frame:

- loc: select rows and columns by labels
- iloc: select rows and columns by integer index
- bracket notation: select columns by labels, df[“name”]

Loc and bracket notation also support boolean arrays.

groupby Key Concepts

If we call `groupby` on a `DataFrame`:

- The resulting output is a `DataFrameGroupBy` object.

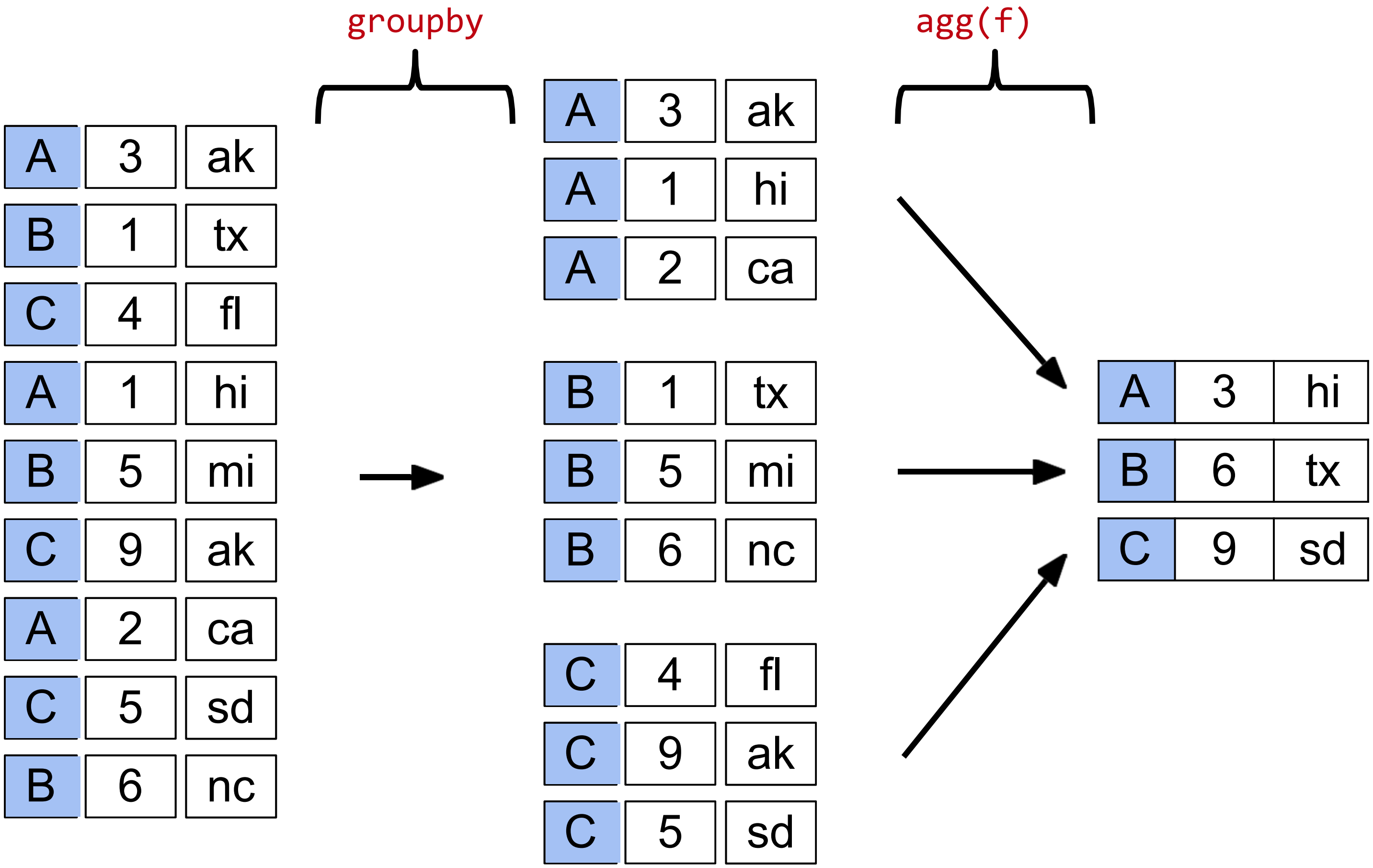
Common uses for `DataFrameGroupBy` objects:

- Aggregation back into a dataframe using an aggregation method.
- Filtering back into a dataframe.

SeriesGroupByObjects are similar, but for Series.

- Can convert a DataFrameGroupBy into a SeriesGroupBy using bracket notation, e.g. dfgb[‘name’] would yield a SeriesGroupBy with only the ‘name’ column, but still grouped as before.

DataFrame groupby/agg Summary



groupby

agg(f)

A	3	ak
B	1	tx
C	4	fl
A	1	hi
B	5	mi
C	9	ak
A	2	ca
C	5	sd
B	6	nc

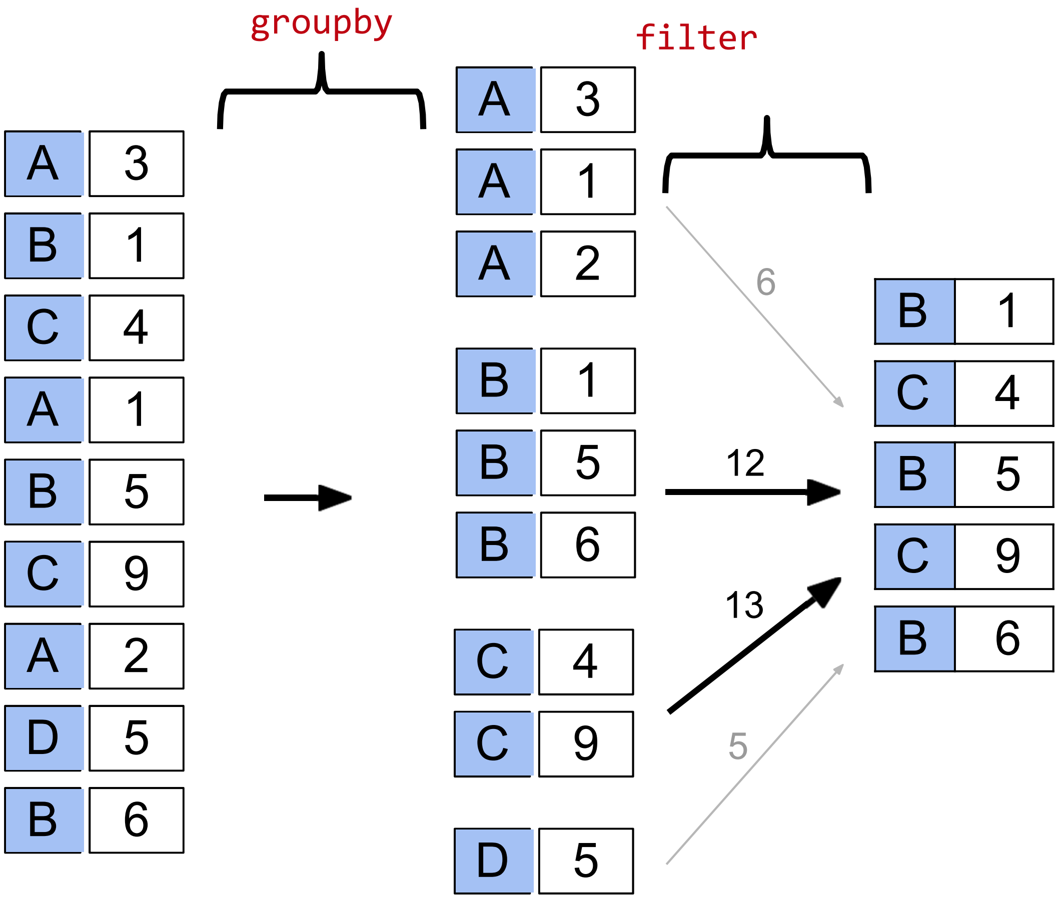
A	3	ak
A	1	hi
A	2	ca

B	1	tx
B	5	mi
B	6	nc

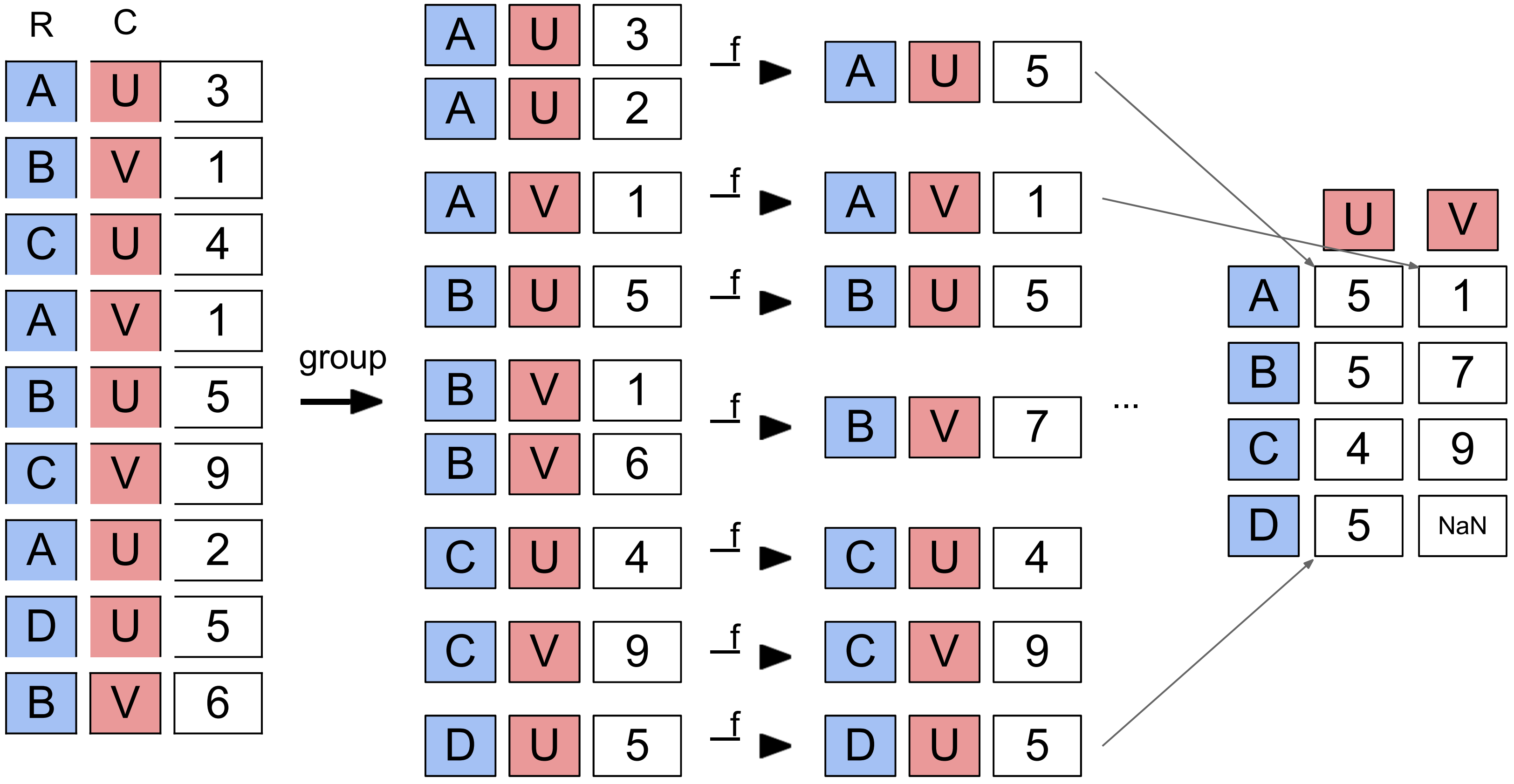
C	4	fl
C	9	ak
C	5	sd

A	3	hi
B	6	tx
C	9	sd

Series groupby/filter Summary



Pivot Tables



R	C	
A	U	3
B	V	1
C	U	4
A	V	1
B	U	5
C	V	9
A	U	2
D	U	5
B	V	6

A	U	3
A	U	2
A	V	1
B	U	5
B	V	1
B	V	6
C	U	4
C	V	9
D	U	5

A	U	5
A	V	1
B	U	5
B	V	7
C	U	4
C	V	9
D	U	5

	U	V
A	5	1
B	5	7
C	4	9
D	5	NaN

Other useful methods and concepts

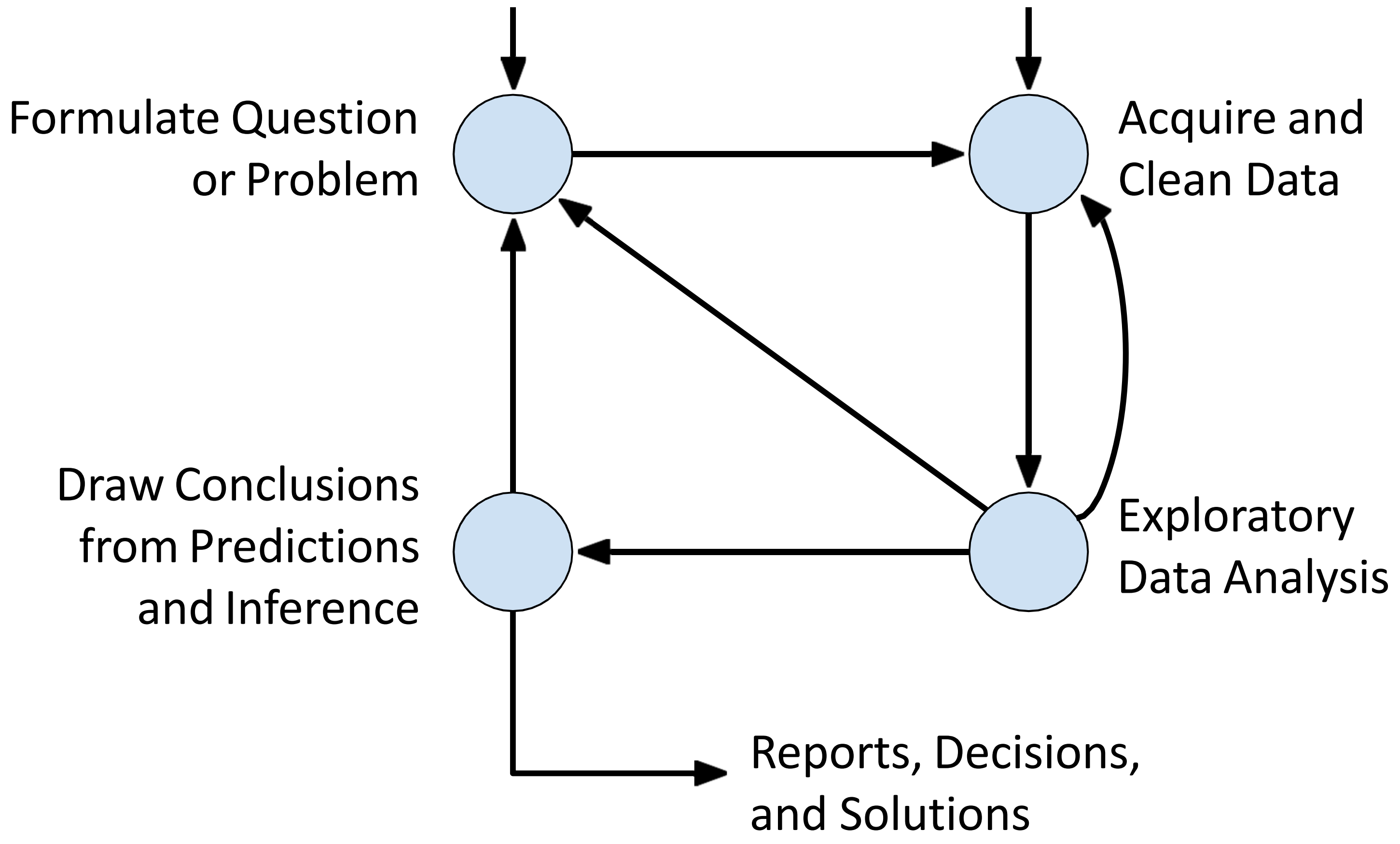
- `.isin(values)`
- `.unique()`
- `.value_counts(...)`
- `.sort_values(...)`
- `.index`
- `.columns`

Joining Tables

Four styles of joins:

- Inner: Returns records that have matching values in both tables.
- Left: Return all records from the left table, and the matched records from the right table.
- Right: Return all records from the right table, and the matched records from the left table.
- Outer: Return all records from both tables.

Data Science Lifecycle (More Standard Terminology)



Formulate Question or Problem

Acquire and Clean Data

Exploratory Data Analysis

Draw Conclusions from Predictions and Inference

Reports, Decisions, and Solutions

Properties of Data and Key Terms

Key Properties:

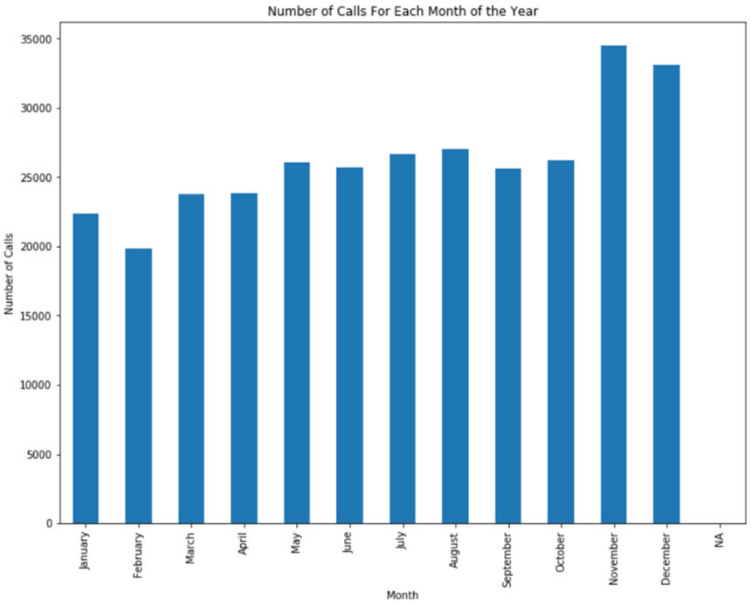
- Quantitative
    - Continuous vs. Discrete data
- Qualitative
    - Ordinal vs. Nominal data.
- Granularity: how fine/coarse is each datum
- Scope: How (in)complete is the data.
- Temporality: How is data situated in time?
- Faithfulness: How well does data represent reality?

Exploratory Data Analysis

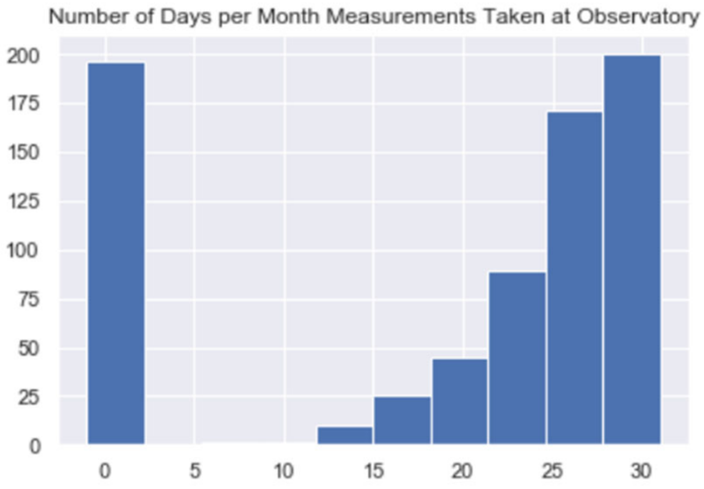
Things one might do:

- Look at data and descriptions of the columns.
- Examine columns individually.
- Examine groups of related records (e.g. grades by major).
- Visualize and summarize data.
- Validate assumptions about data.
- Identify and address inaccuracies.
- Apply transformations.
- Record everything you do.

Visualization: Plot Types

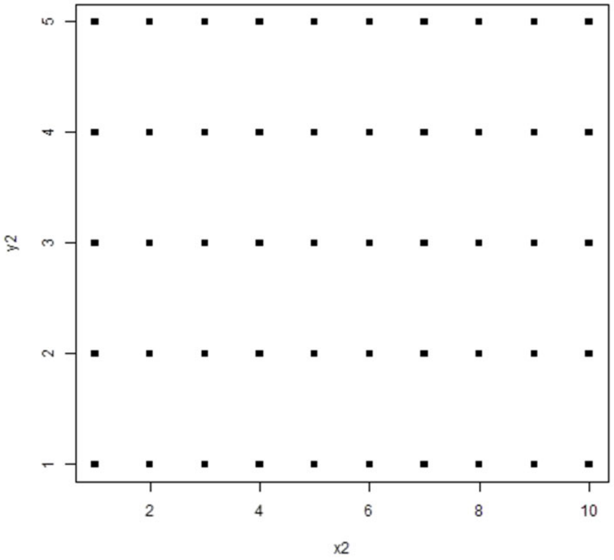


Bar chart
(sns.barplot)



Histogram
(sns.distplot)

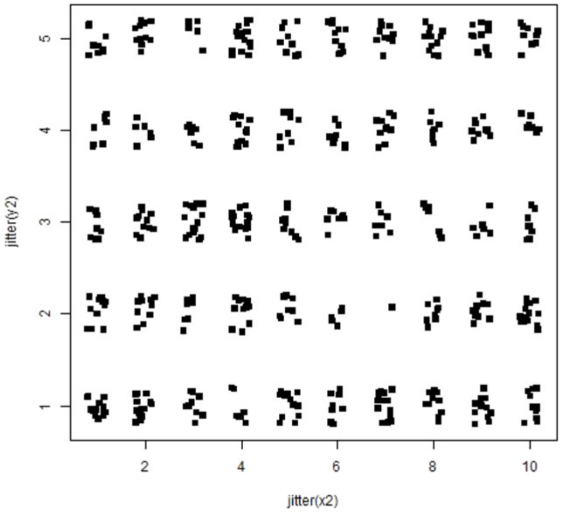
Visualization: Plot Types



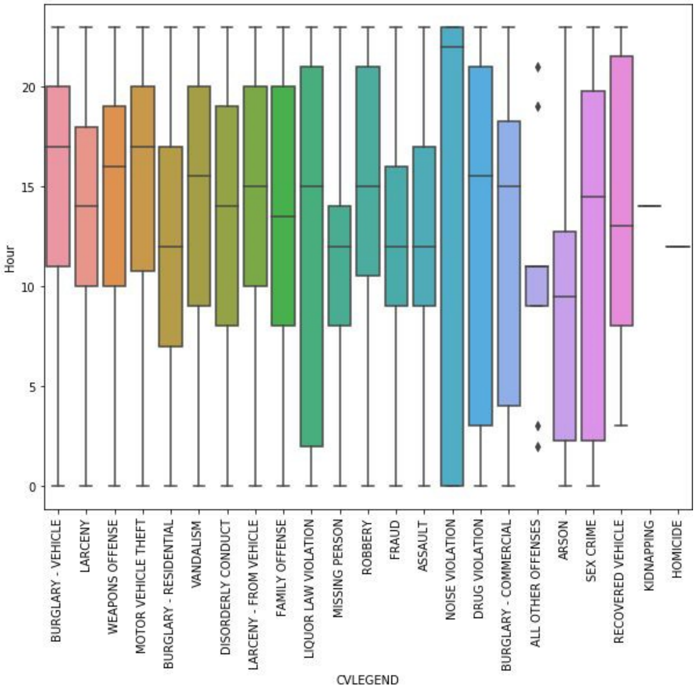
Scatter Plot
sns.scatterplot

Scatterplots often suffer from overplotting.

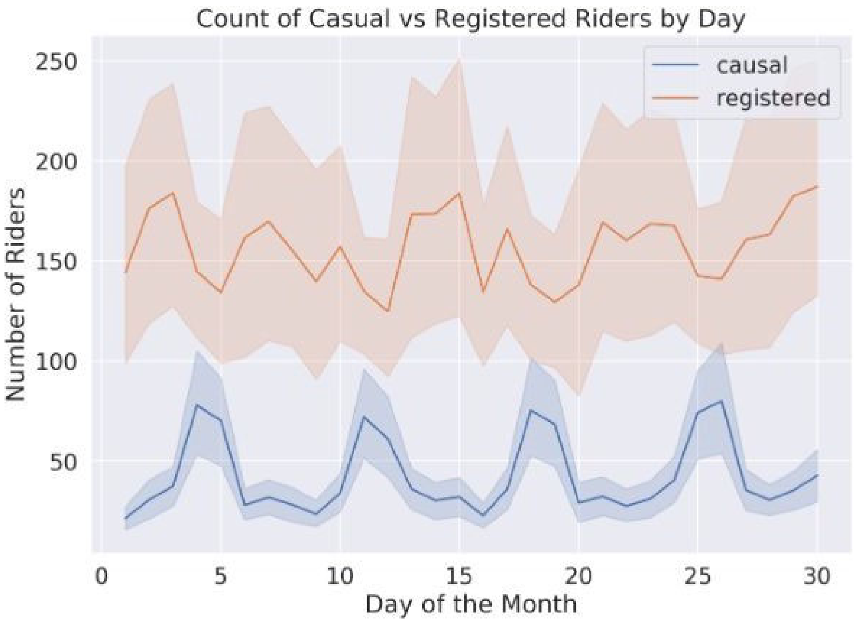
- Use jittering to fix!



Visualization: Plot Types



Box Plot
sns.boxplot

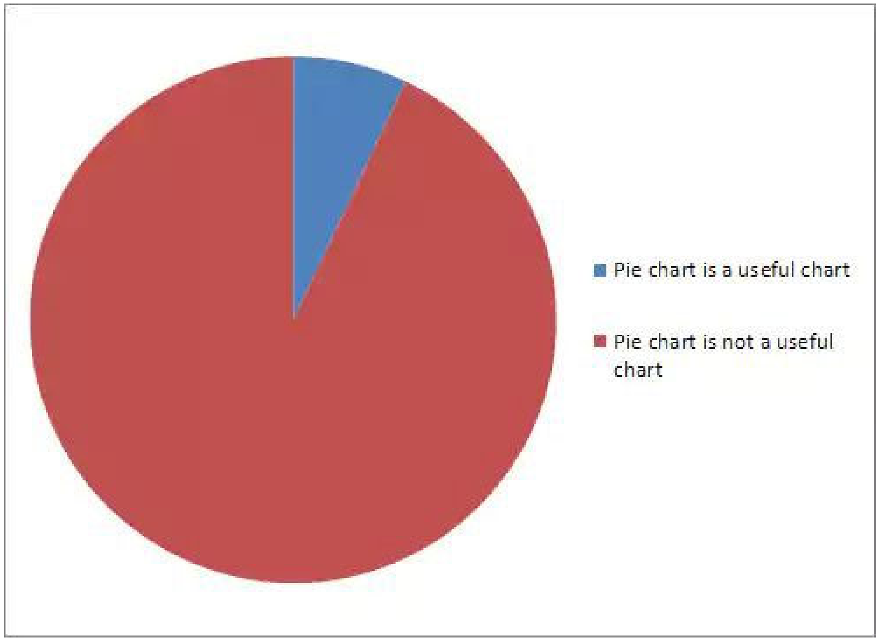


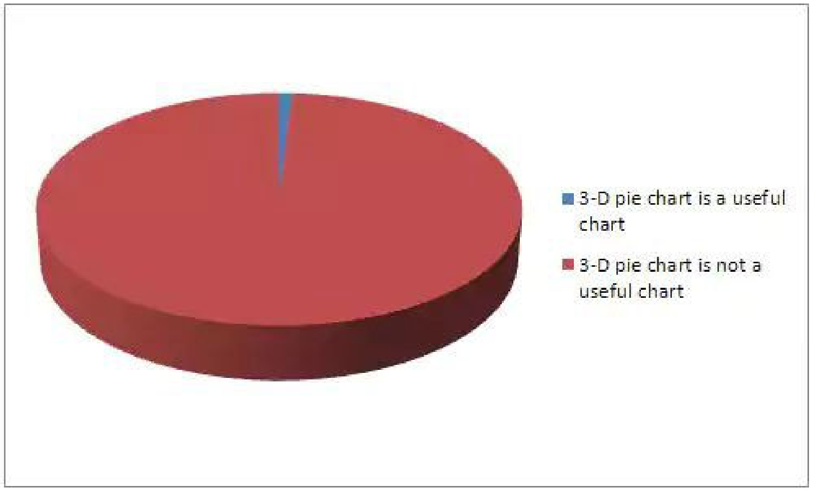
Line Plot
sns.lineplot

Visualization: Plot Types

Pie Charts were a bad idea, don't use them.

- Question for you: Why are they bad?



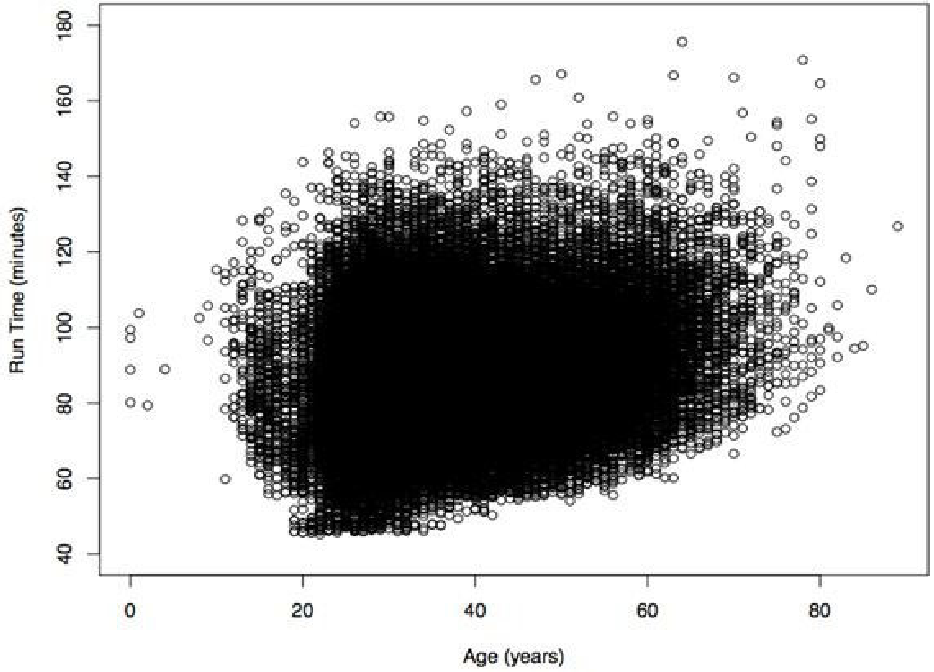


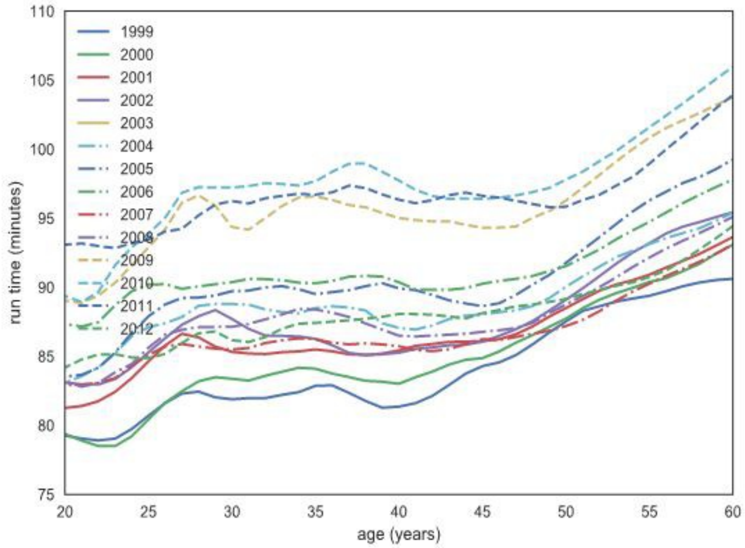
In general: Good to keep human perception in mind!

Stratification

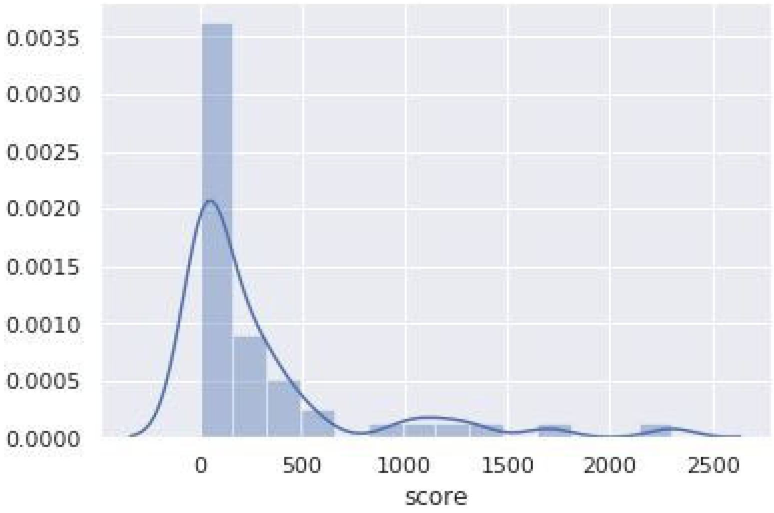
Stratification can be a very powerful tool for data visualization.

- Same data, but right plot stratifies results by the year of the race.

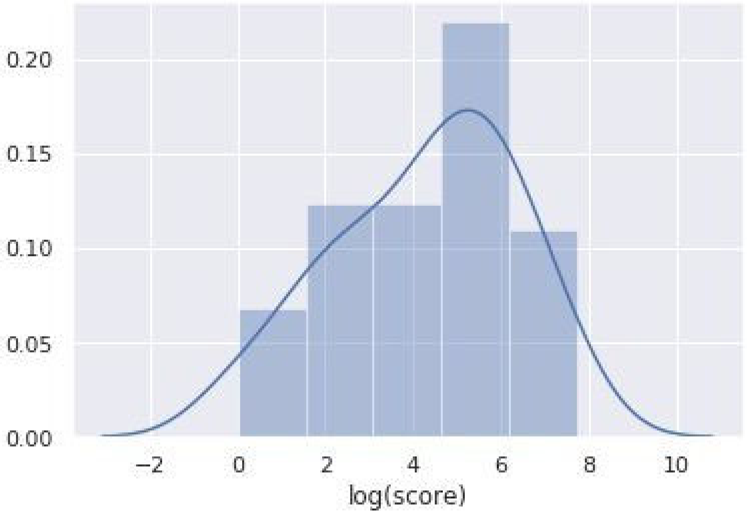




Log Transformation

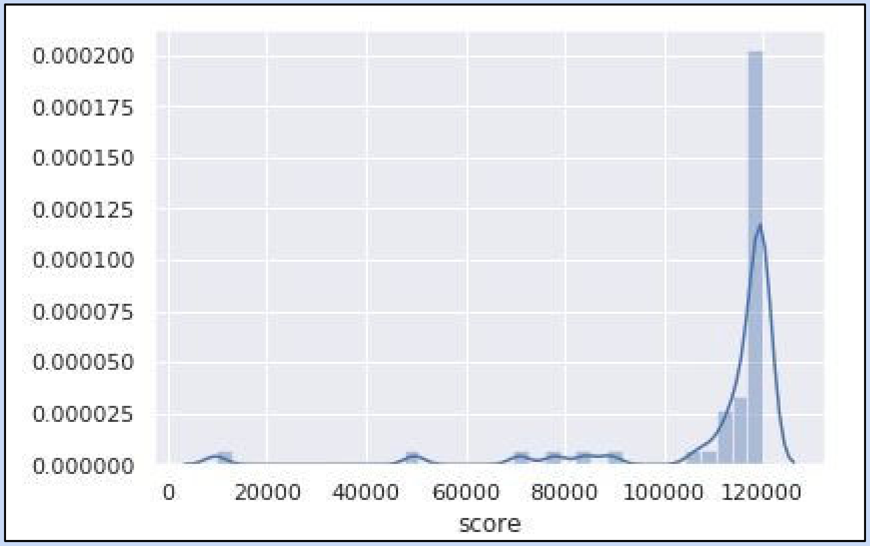


Data in this sns.distplot is right skewed.



After log transformation, it is more symmetrical.

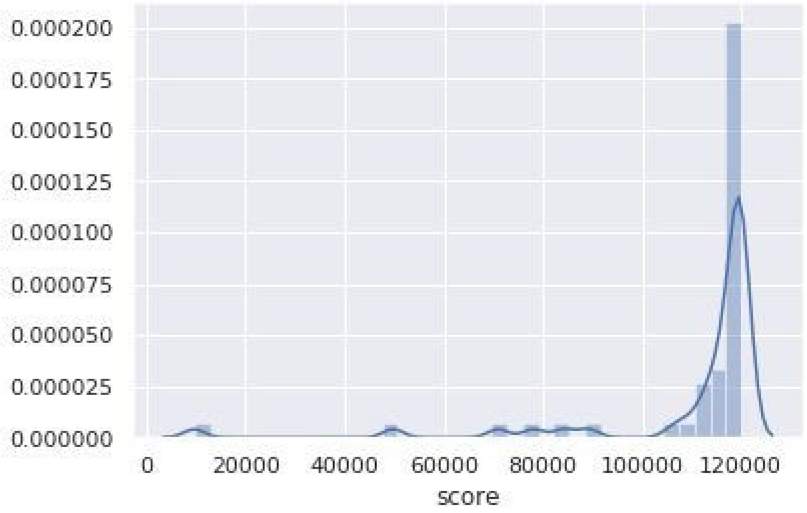
Log Transformation

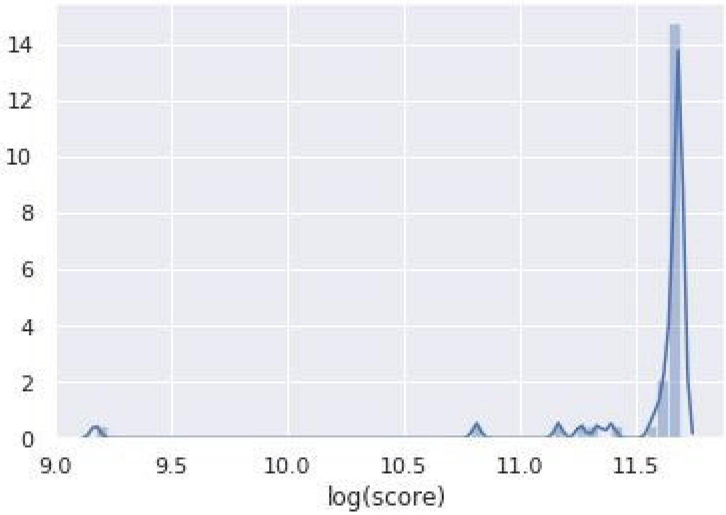


Data in this sns.distplot is left skewed.

If we plot the log of score, will this help?

Log Transformation





Data in this sns.distplot is left skewed.

If we plot the log of score, will this help?

- No! It exacerbates the problem.
- Instead want to use a transformation that separates apart the stuff close to the dense part of the dataset.

Log Transformation

$$y = a^x \rightarrow \log(y) = x \log(a)$$

$$y = ax^k \rightarrow \log(y) = \log(a) + k \log(x)$$

Regular Expression Syntax

AB*: A then zero or more copies of B: A, AB, ABB, ABBB

(AB)*: Zero or more copies of AB: ABABABAB, ABAB, AB,

operation	order	example	matches	does not match
concatenation	3	AABAAB	AABAAB	every other string
or	4	AA|BAAB	AA BAAB	every other string
closure (zero or more)	2	AB*A	AA ABBBBBBA	AB ABABA
parenthesis	1	A(A|B)AAB	AAAAB ABAAB	every other string
		(AB)*A	A ABABABABA	AA ABBA

Expanded Regex Syntax

operation	example	matches	does not match
wildcard	`.U.U.U.`	CUMULUS JUGULUM	SUCCUBUS TUMULTUOUS
character class	`[A-Za-z][a-z]*`	word Capitalized	camelCase 4illegal
at least 1	`m(oo)+n`	moon moooon	mn mon
between a and b occurrences	`m[aeiou]{1,2}m`	mem maam miem	mm mooom meme

These additional operations confer no additional power to regexes.

- For every regex in this expanded syntax, there is a regex in the basic syntax.
- Ex. `[A-E]+` is just shorthand for `(A|B|C|D|E)(A|B|C|D|E)*`

Even More Regular Expression Syntax

operation	example	matches	does not match
built-in character classes	\w+ \d+	fawef 231231	this person 423 people
character class negation	[^a-z]+	PEPPERS3982 17211!↑å	porch CLAmS
escape character	cow\.com	cow.com	cowscom

Suppose you want to match one of our special characters like . or [or]

- In these cases, you must “escape” the character using the backslash.
- You can think of the backslash as meaning “take this next character literally”. We did not discuss the idea of r’’ vs. ‘’ strings in Python.

Even More Regular Expression Features

operation	example	matches	does not match

beginning of line	`^ark`	`ark two`
		`ark o ark`
			`dark`

end of line	`ark$`	`dark`
		`ark o ark`
			`ark two`

non-greedy qualifier	`5.*?5`	`5005`
		`55`
			`5005005`

5*5 would match this!

A few additional common regex features are listed above.

- Won’t discuss these in class, but might come up in discussion or hw.
- There are even more out there!

For the best guide you’ll ever read on regex in Python:
https://docs.python.org/2/howto/regex.html