NYU Center for Data Science

DS-UA 112
Introduction to Data Science

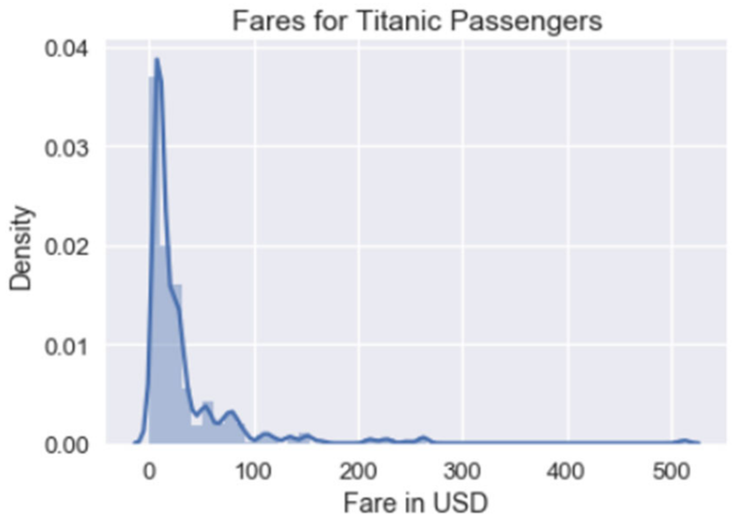
Lecture 12
Text II - Text from Websites

Announcements

- Homework 3
  - Due Friday October 18
- Project 1
  - Extended to Sunday October 27
- Midterm
  - Wednesday October 23 4:55-6:10
  - Pencil and Paper with Cheat-Sheets
  - Section and Office Hours
  - Practice Exam

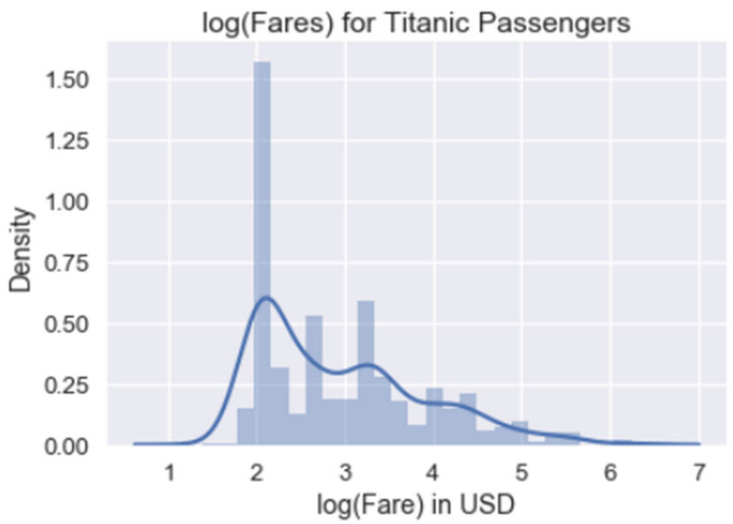
Review

- Transformations
  - Logarithms and Powers



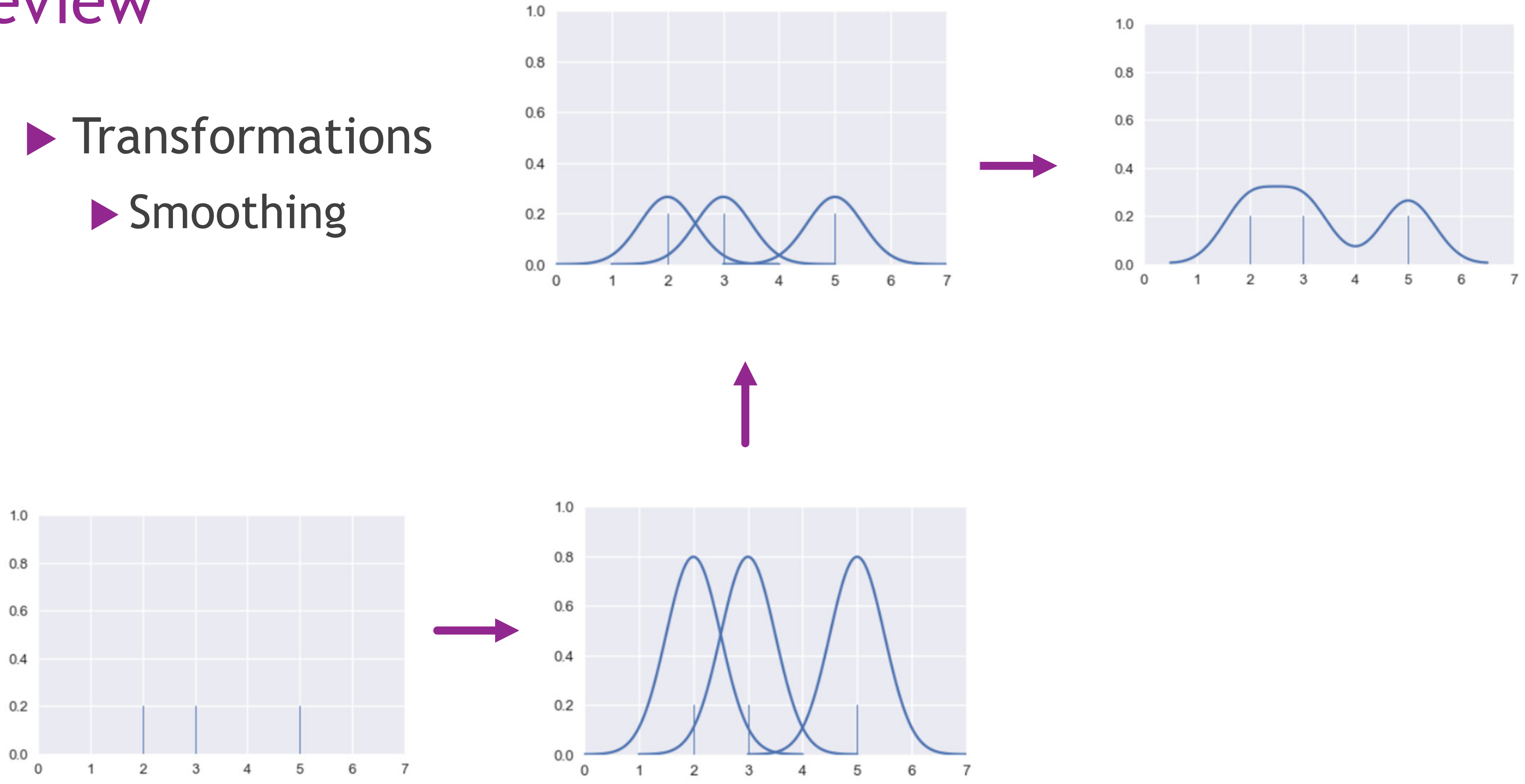
$\longrightarrow$

Logarithm



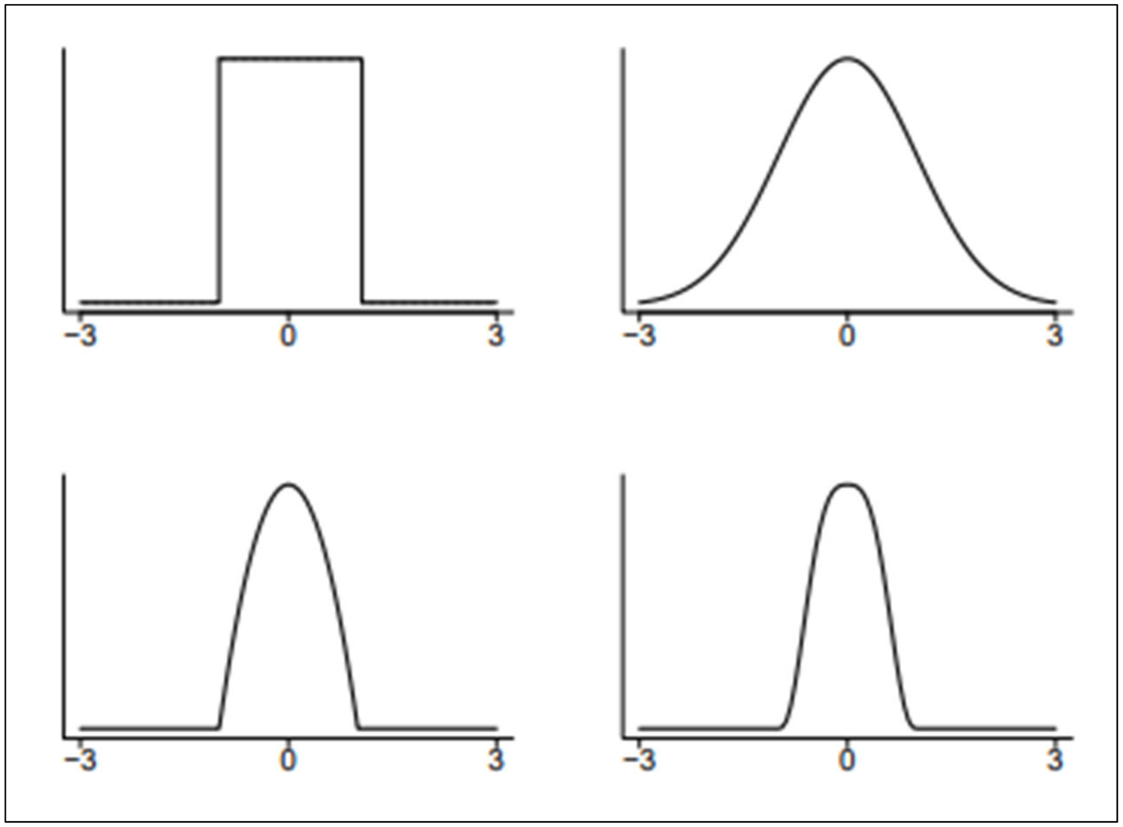
Review

- Transformations
  - Smoothing



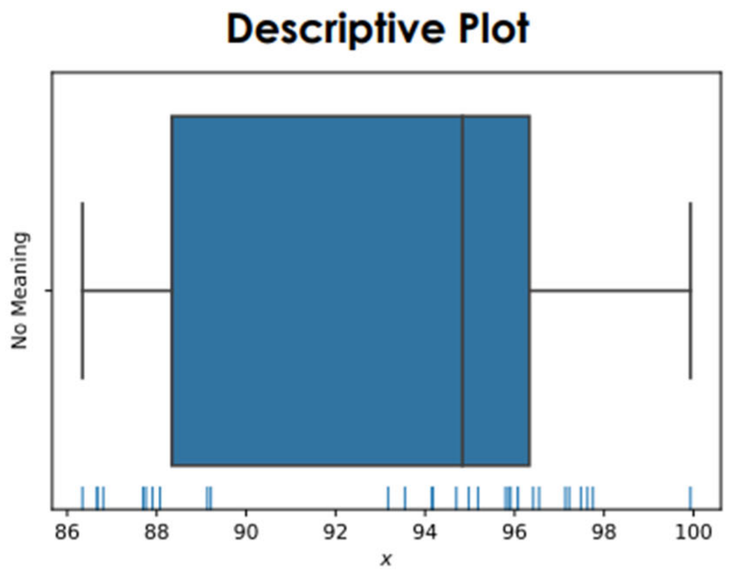
Review

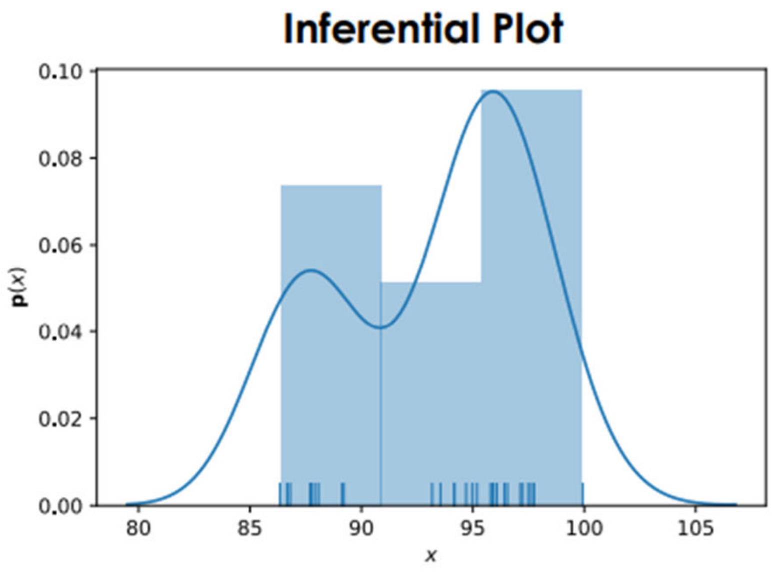
- Transformations
  - Smoothing



Review

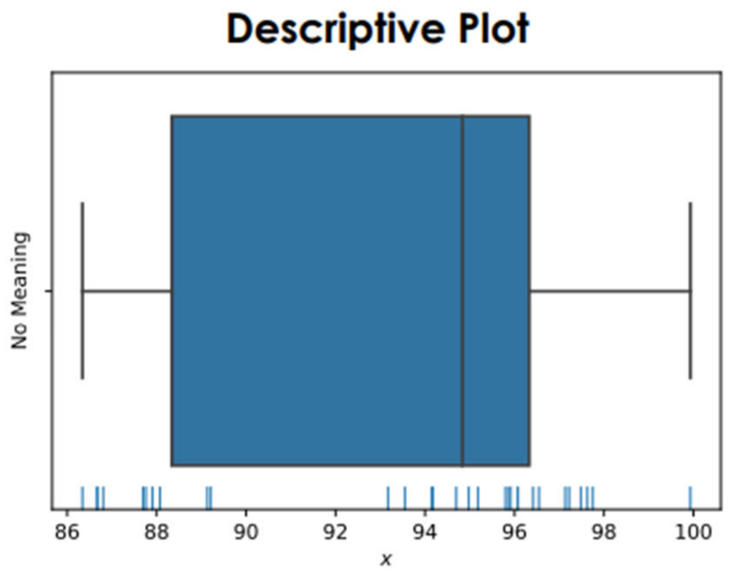
- Transformations
  - Smoothing

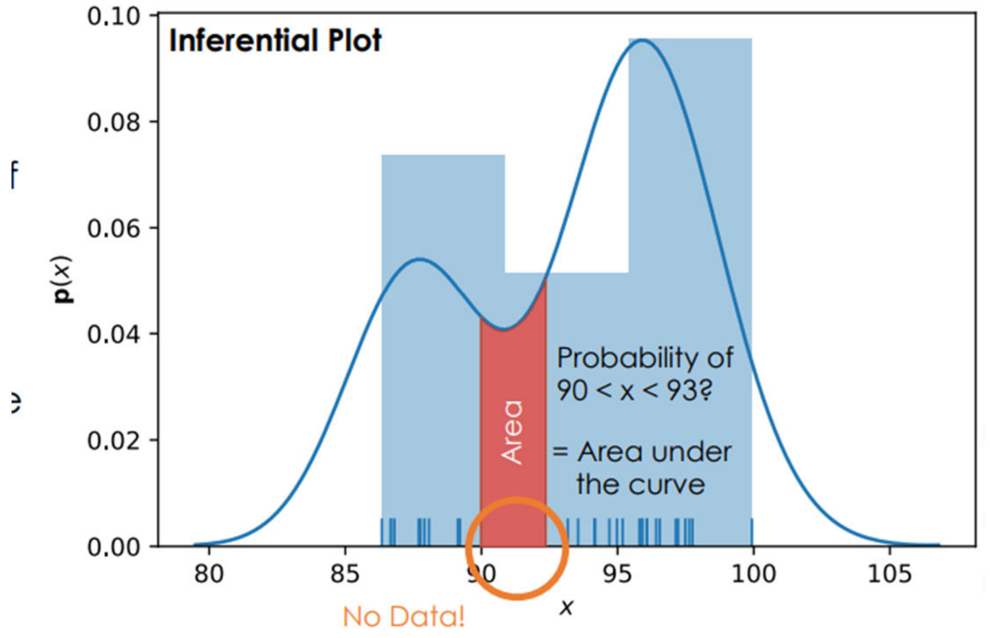




Review

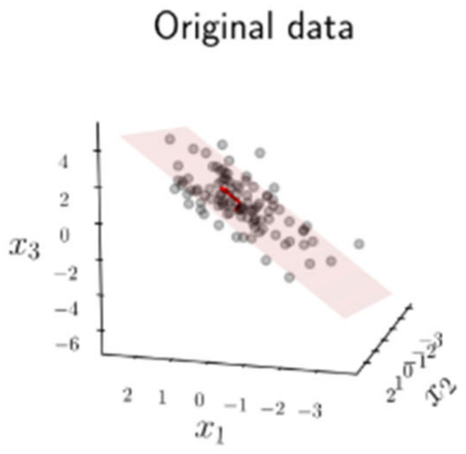
- Transformations
  - Smoothing

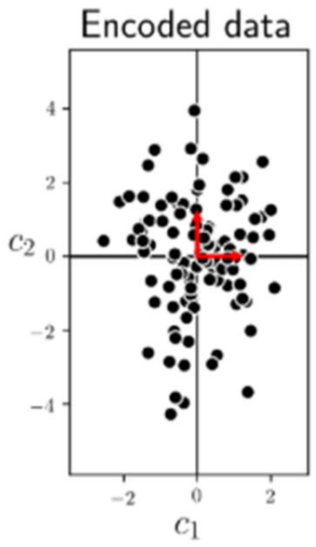




Review

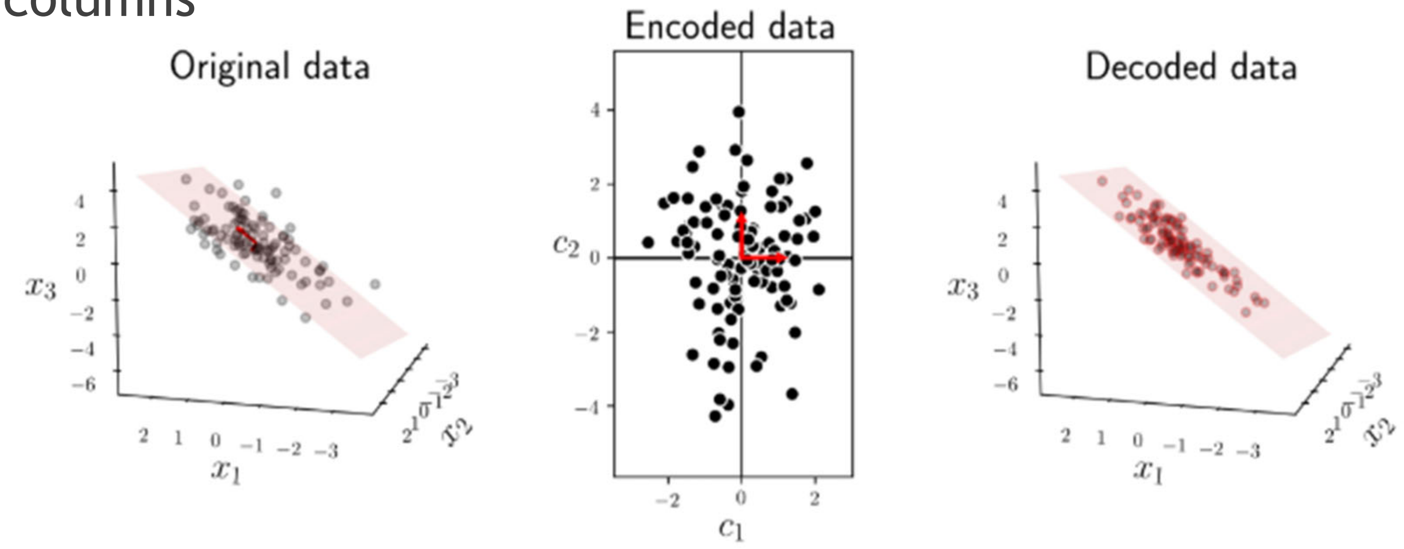
- Transformations
  - Reducing Dimension





Review

- Transformations
  - Reducing Dimension
    - Many rows
    - Many columns



Agenda

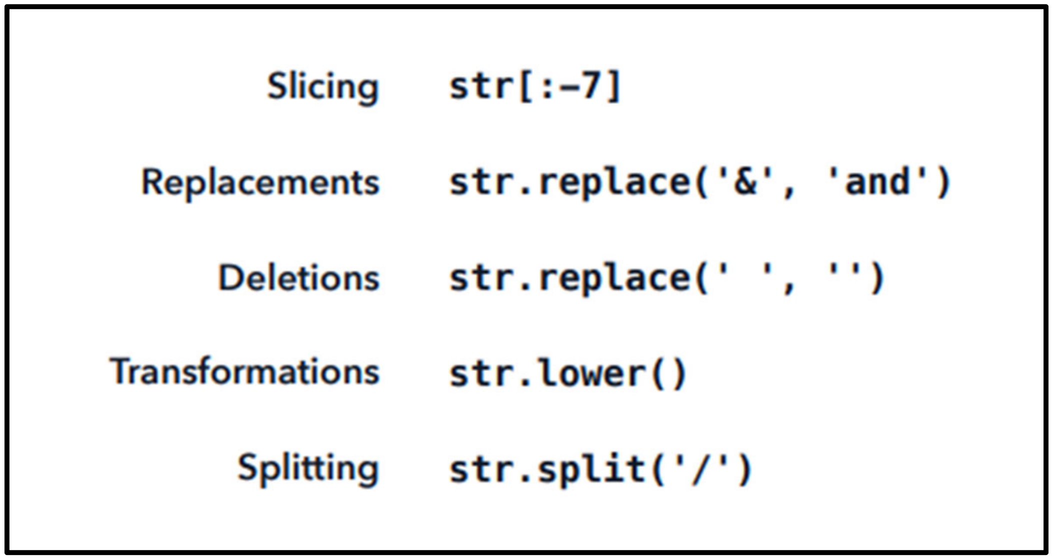
- Lessons
  - Working with dates and times
  - Data from Websites
- Demos
  - Police Reports
  - Restaurant Inspections
- Questions

Objectives

- Properties of Data
  - What are Scope, Temporality, Granularity, Faithfulness?
- Application Programming Interfaces
  - What file formats do we need for Websites?
- Readings:
  - Nolan 5.3-5.7, 7.1, 8
  - Grus 9

String Methods

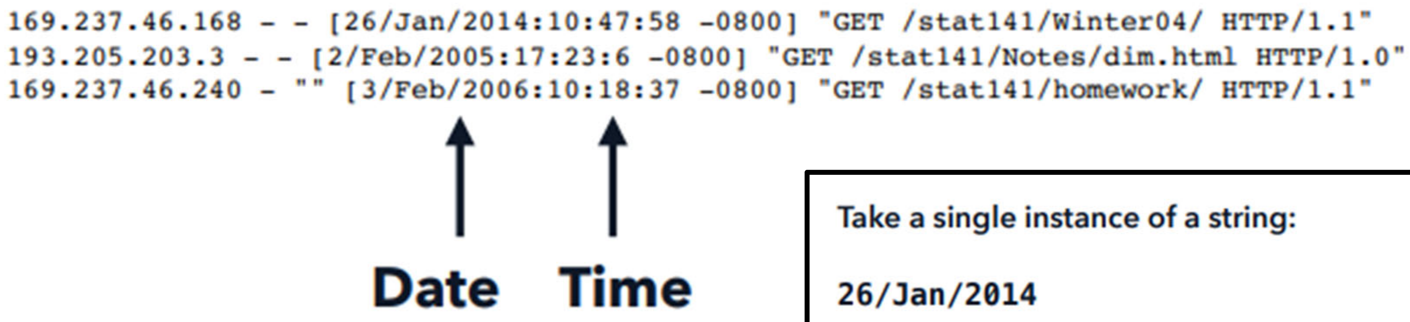
- Sometimes strings contain special characters like ‘\n’ for newline.
- We can escape these special characters with an extra backslash ‘\\n’ or indicate raw string r’\n’



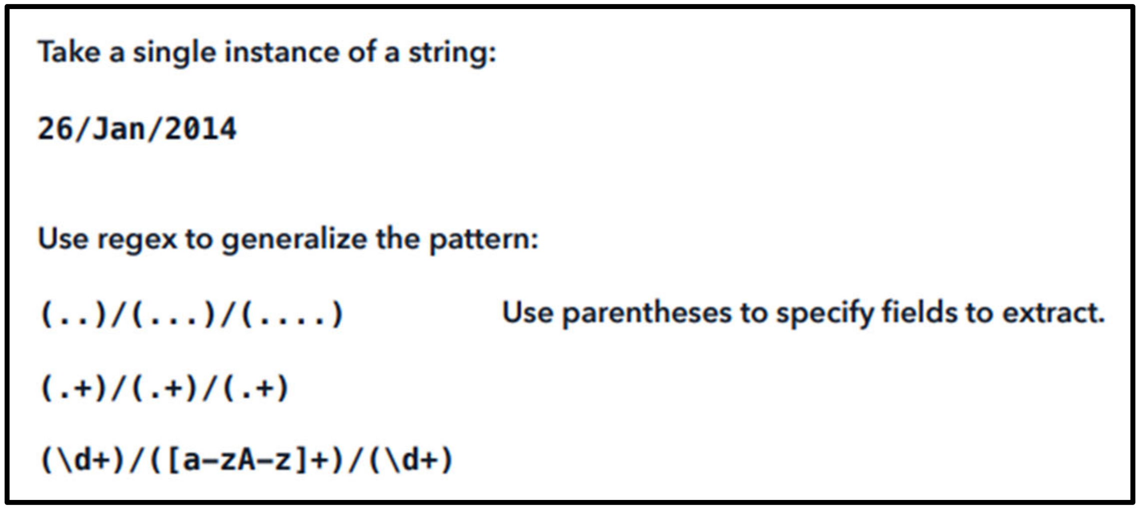
Slicing	`str[:-7]`
Replacements	`str.replace('&', 'and')`
Deletions	`str.replace(' ', '')`
Transformations	`str.lower()`
Splitting	`str.split('/')`

Regular Expressions

- Rules for matching portions of string. Useful for extracting fields like date and time…



```
169.237.46.168 - - [26/Jan/2014:10:47:58 -0800] "GET /stat141/Winter04/ HTTP/1.1"
193.205.203.3 - - [2/Feb/2005:17:23:6 -0800] "GET /stat141/Notes/dim.html HTTP/1.0"
169.237.46.240 - "" [3/Feb/2006:10:18:37 -0800] "GET /stat141/homework/ HTTP/1.1"
```



Take a single instance of a string:

```
26/Jan/2014
```

Use regex to generalize the pattern:

```
(..)/(...)/(....)
```

Use parentheses to specify fields to extract.

```
(.+)/(.+)/(.+)

(\d+)/([a-zA-z]+)/(\d+)
```

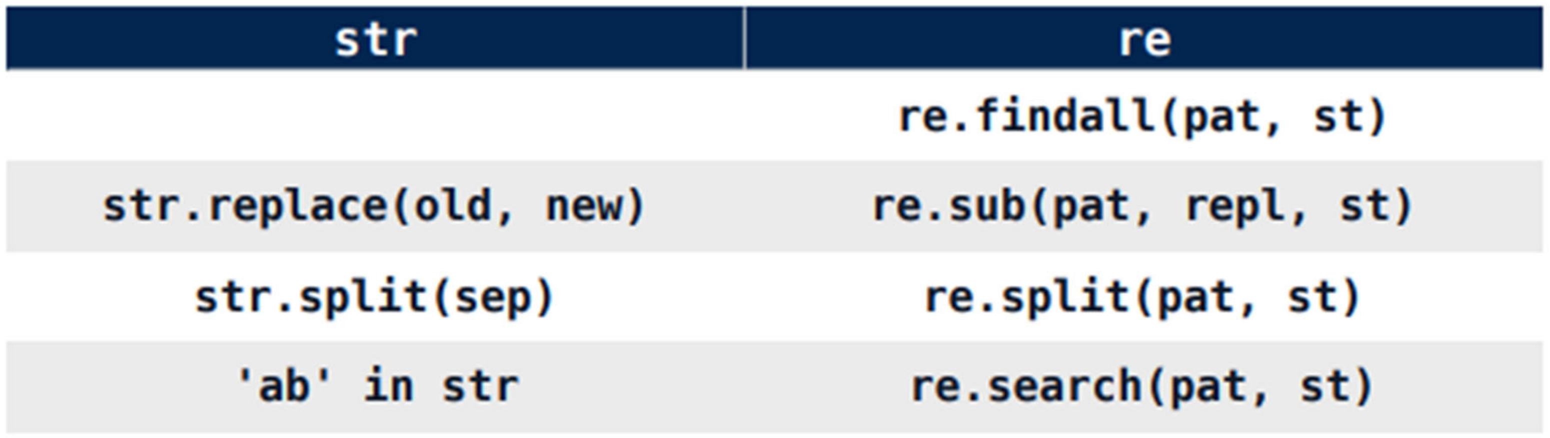
Regular Expressions

- Set of Characters
- Number of Characters
- Position in String
- Short hands for letters, numbers and space

Description	Bracket Form	Shorthand
Alphanumeric character	`[a-zA-Z0-9]`	`\w`
Not an alphanumeric character	`[^a-zA-Z0-9]`	`\W`
Digit	`[0-9]`	`\d`
Not a digit	`[^0-9]`	`\D`
Whitespace	`[\t\n\f\r\p{Z}]`	`\s`
Not whitespace	`[^\t\n\f\r\p{z}]`	`\S`

Char	Description	Example	Matches	Doesn't Match
.	Any character except \n	`...`	abc	ab abcd
[ ]	Any character inside brackets	`[cb.]ar`	car .ar	jar
[^]	Any character *not* inside brackets	`[^b]ar`	car par	bar ar
*	≥ 0 or more of last symbol	`[pb]*ark`	bbark ark	dark
+	≥ 1 or more of last symbol	`[pb]+ark`	bbpark bark	dark ark
?	0 or 1 of last symbol	`s?he`	she he	the
{*n*}	Exactly *n* of last symbol	`hello{3}`	hellooo	hello
|	Pattern before or after bar	`we|[ui]s`	we us is	e s
\	Escapes next character	`\[hi\]`	[hi]	hi
^	Beginning of line	`^ark`	ark two	dark
$	End of line	`ark$`	noahs ark	noahs arks

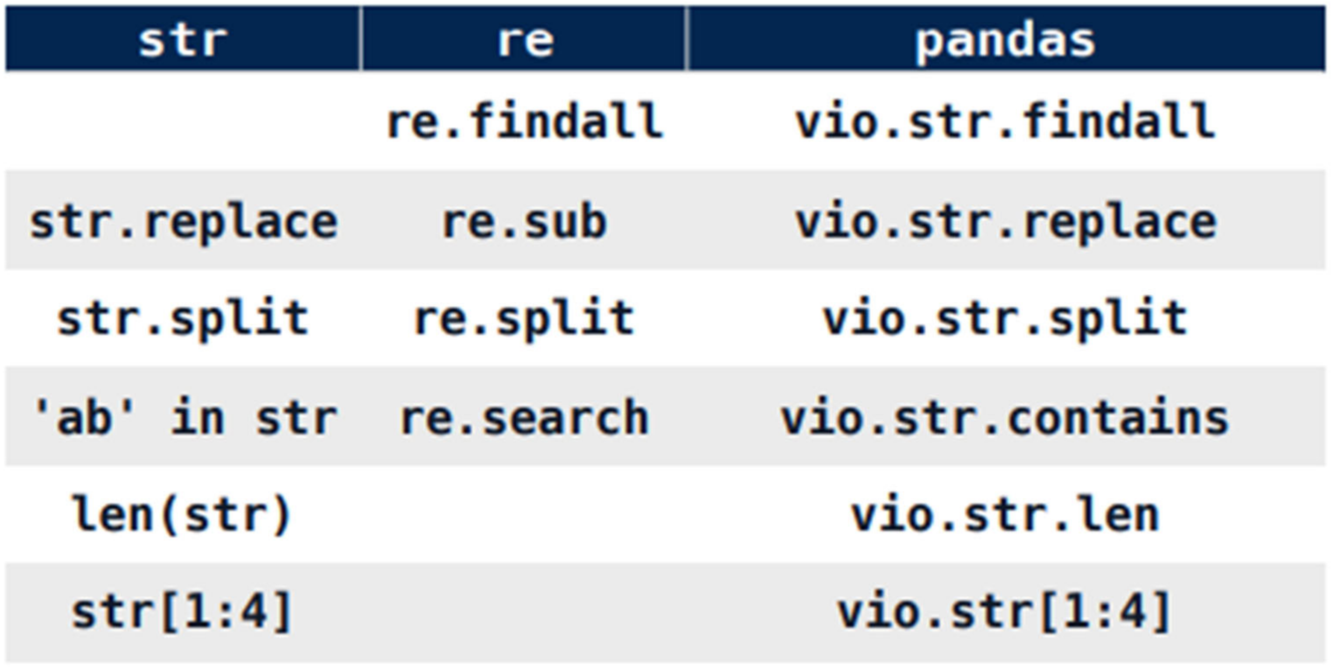
String Methods vs Regex



str	re
	`re.findall(pat, st)`
`str.replace(old, new)`	`re.sub(pat, repl, st)`
`str.split(sep)`	`re.split(pat, st)`
`'ab' in str`	`re.search(pat, st)`

https://docs.python.org/3/library/re.html

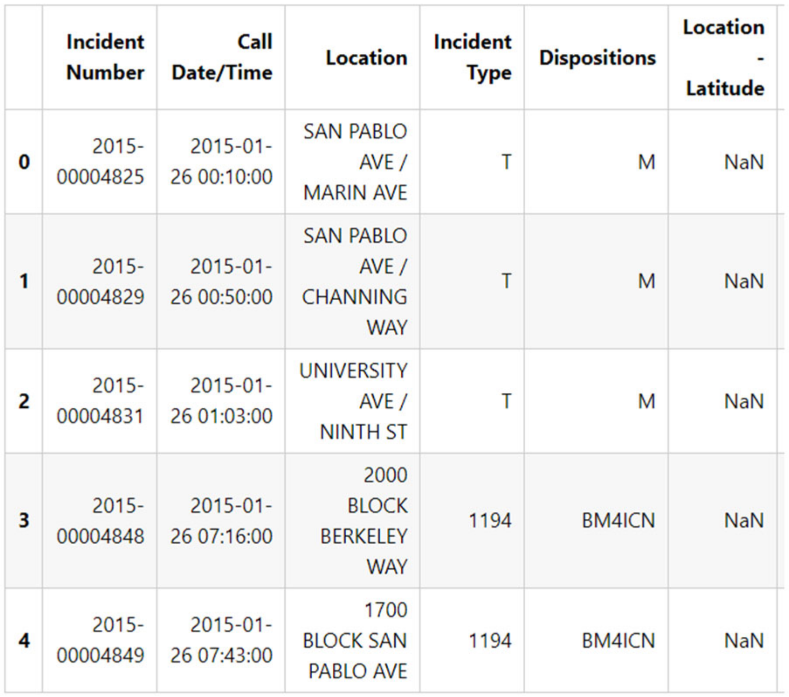
String Methods vs Regex vs pandas (DEMO)



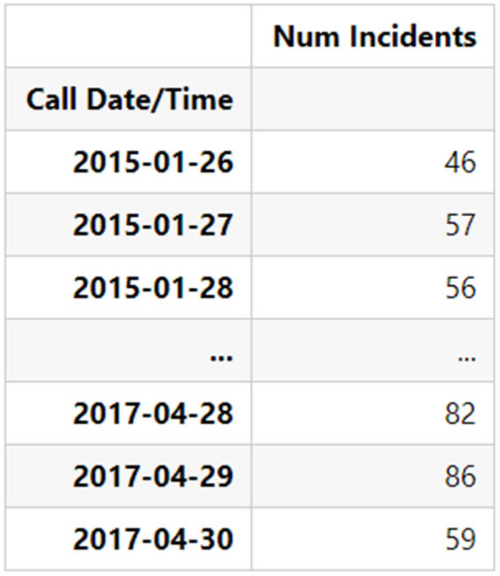
str	re	pandas
	re.findall	vio.str.findall
str.replace	re.sub	vio.str.replace
str.split	re.split	vio.str.split
'ab' in str	re.search	vio.str.contains
len(str)		vio.str.len
str[1:4]		vio.str[1:4]

Properties of Data

- The ***granularity*** of your data is what each record in your data represents. We have coarse and fine granularity



	Incident Number	Call Date/Time	Location	Incident Type	Dispositions	Location - Latitude
0	2015-00004825	2015-01-26 00:10:00	SAN PABLO AVE / MARIN AVE	T	M	NaN
1	2015-00004829	2015-01-26 00:50:00	SAN PABLO AVE / CHANNING WAY	T	M	NaN
2	2015-00004831	2015-01-26 01:03:00	UNIVERSITY AVE / NINTH ST	T	M	NaN
3	2015-00004848	2015-01-26 07:16:00	2000 BLOCK BERKELEY WAY	1194	BM4ICN	NaN
4	2015-00004849	2015-01-26 07:43:00	1700 BLOCK SAN PABLO AVE	1194	BM4ICN	NaN

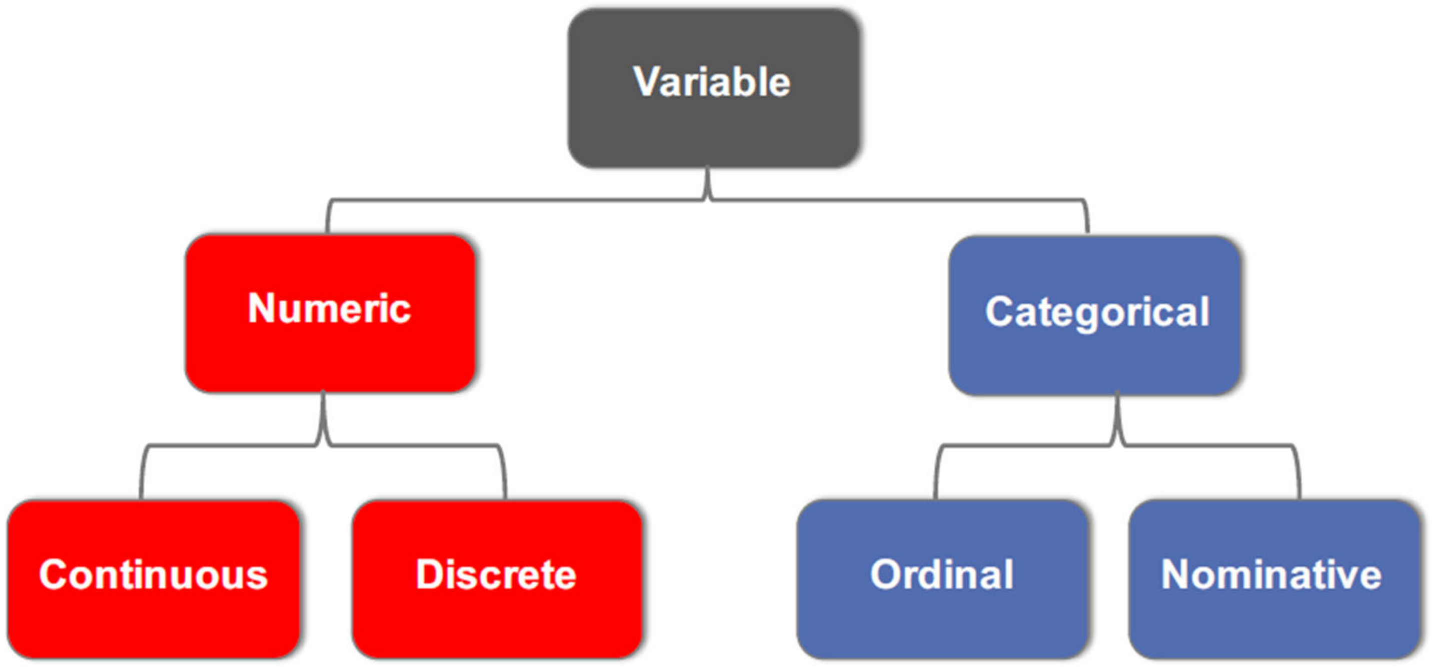


	Num Incidents
Call Date/Time	
2015-01-26	46
2015-01-27	57
2015-01-28	56
...	...
2017-04-28	82
2017-04-29	86
2017-04-30	59

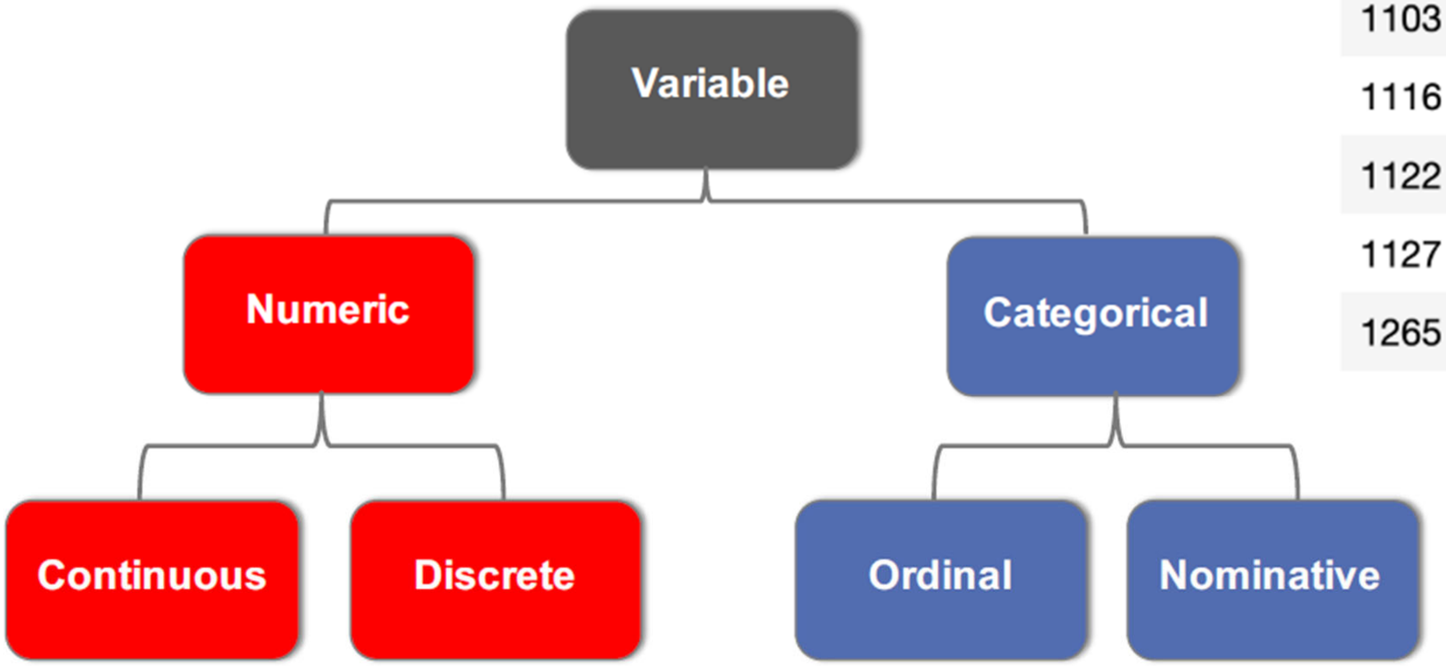
Properties of Data

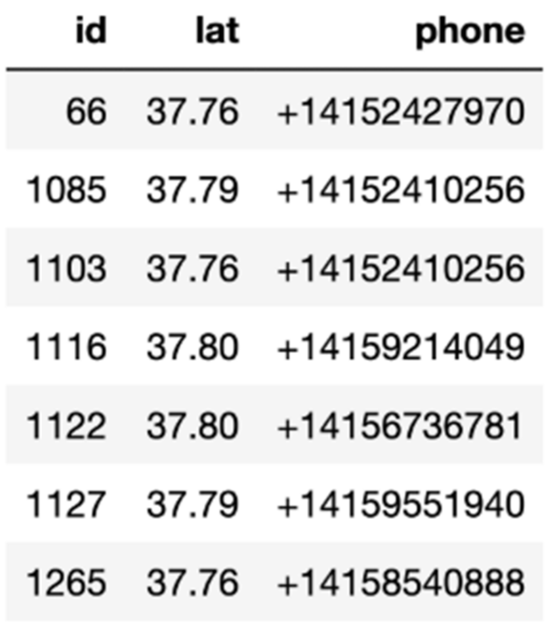
- **Data Types** What kinds of data do we have?
- **Granularity** How fine/coarse is each datum?
- **Scope** How (in)complete are the data?
- **Temporality** How are the data situated in time?
- **Faithfulness** How accurately do the data describe the world?

Data Types: Statistical vs Computational



Data Types: Statistical vs Computational





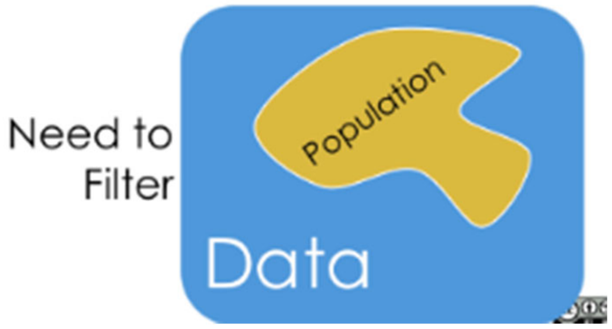
id	lat	phone
66	37.76	+14152427970
1085	37.79	+14152410256
1103	37.76	+14152410256
1116	37.80	+14159214049
1122	37.80	+14156736781
1127	37.79	+14159551940
1265	37.76	+14158540888

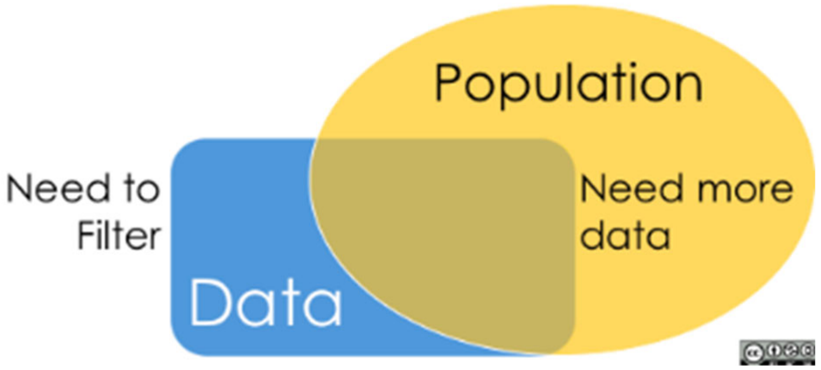
Properties of Data

- The ***granularity*** of your data is what each record in your data represents. We have coarse and fine granularity
  - What does a record represent?
  - Do all records capture granularity at the same level? (Sometimes a table will contain summary rows.)
  - If the data were aggregated, how was the aggregation performed? Sampling and averaging are common aggregations.
  - What kinds of aggregations can we perform on the data?
- In general, how do we change the granularity?

Properties of Data

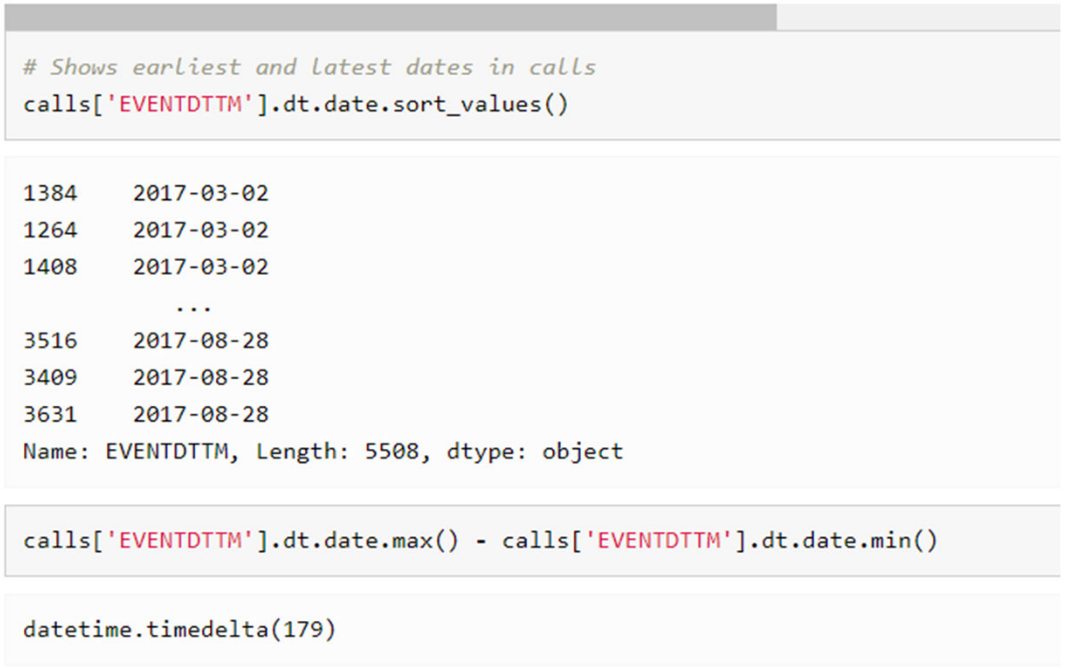
- The **scope** of the dataset refers to the coverage of the dataset in relation to what we are interested in analyzing.
  - Geographic Scope?





Properties of Data

- The ***temporality*** refers to the date and time fields in the dataset.
  - What is the meaning of the date and time fields in the dataset?
  - What representation do the date and time fields have in the data?
  - Are there strange timestamps that might represent null values?



```
# Shows earliest and latest dates in calls
calls['EVENTDTTM'].dt.date.sort_values()
```

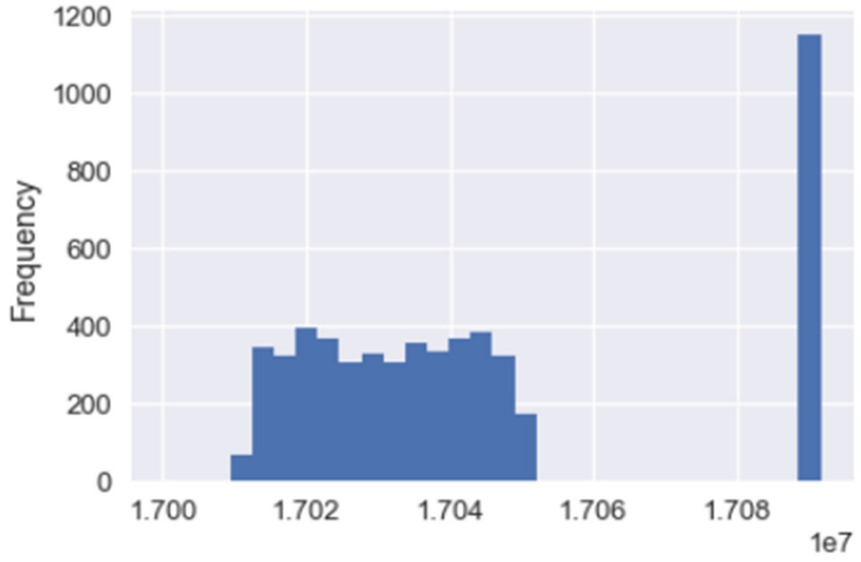
```
1384    2017-03-02
1264    2017-03-02
1408    2017-03-02
           ...
3516    2017-08-28
3409    2017-08-28
3631    2017-08-28
Name: EVENTDTTM, Length: 5508, dtype: object
```

```
calls['EVENTDTTM'].dt.date.max() - calls['EVENTDTTM'].dt.date.min()
```

```
datetime.timedelta(179)
```

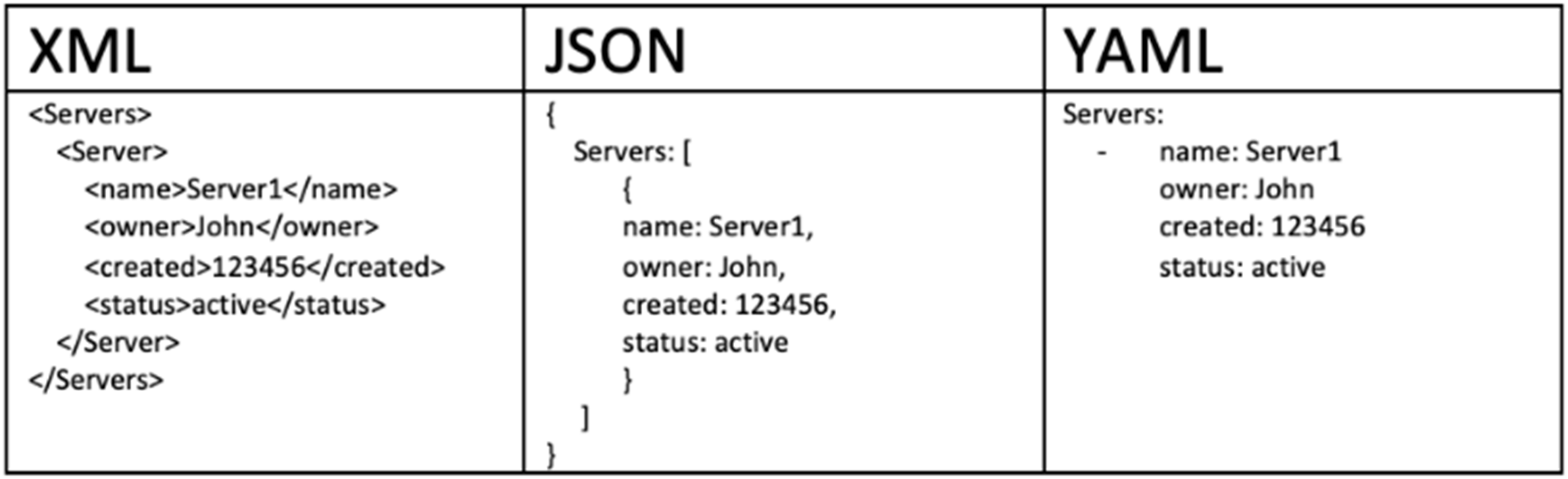
Properties of Data

- We describe a dataset as ***faithful*** if we believe it accurately captures reality.
  - Unrealistic or incorrect values
  - Violations of obvious dependencies
  - Hand-entered data
  - Clear signs of data falsification



Data Formats for Websites

- Descriptive
- Extensible
- Human and Machine Readable



XML

```
<Servers>
   <Server>
      <name>Server1</name>
      <owner>John</owner>
      <created>123456</created>
      <status>active</status>
   </Server>
</Servers>
```

JSON

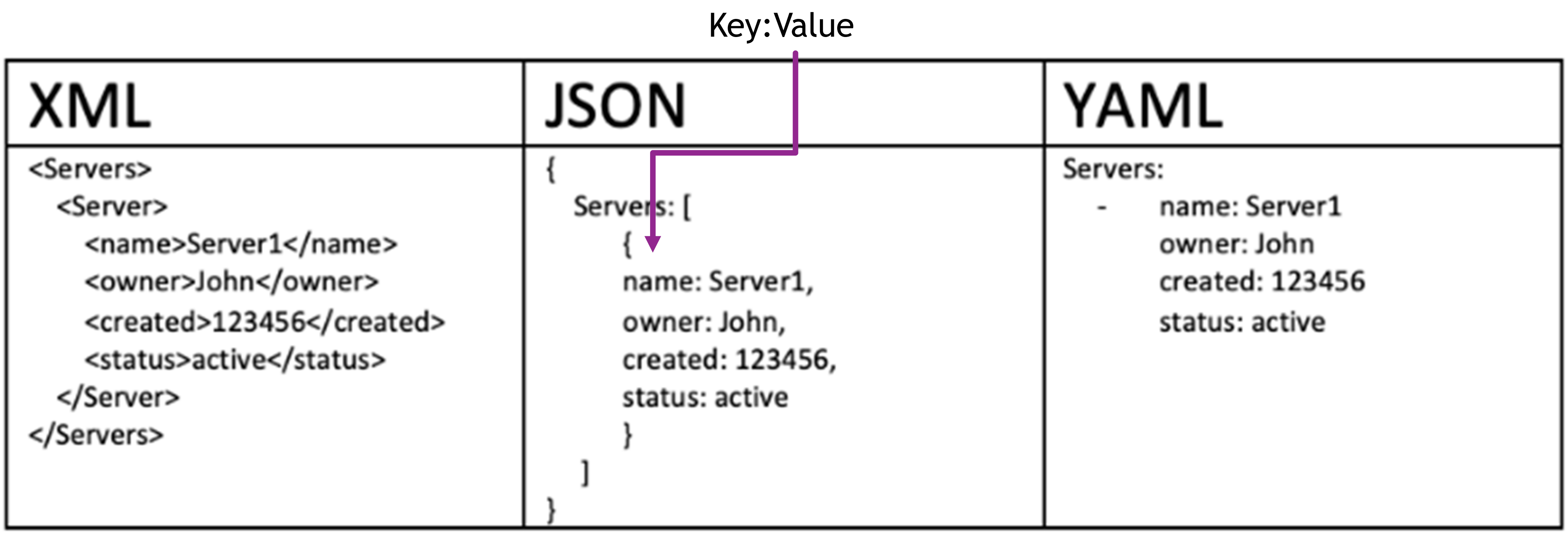
```
{
   Servers: [
        {
        name: Server1,
        owner: John,
        created: 123456,
        status: active
        }
    ]
}
```

YAML

```
Servers:
    -      name: Server1
           owner: John
           created: 123456
           status: active
```

JavaScript Object Notation

- Key: Value
- Value is Array of
  - string, number, Boolean, null



XML

```
<Servers>
   <Server>
      <name>Server1</name>
      <owner>John</owner>
      <created>123456</created>
      <status>active</status>
   </Server>
</Servers>
```

JSON

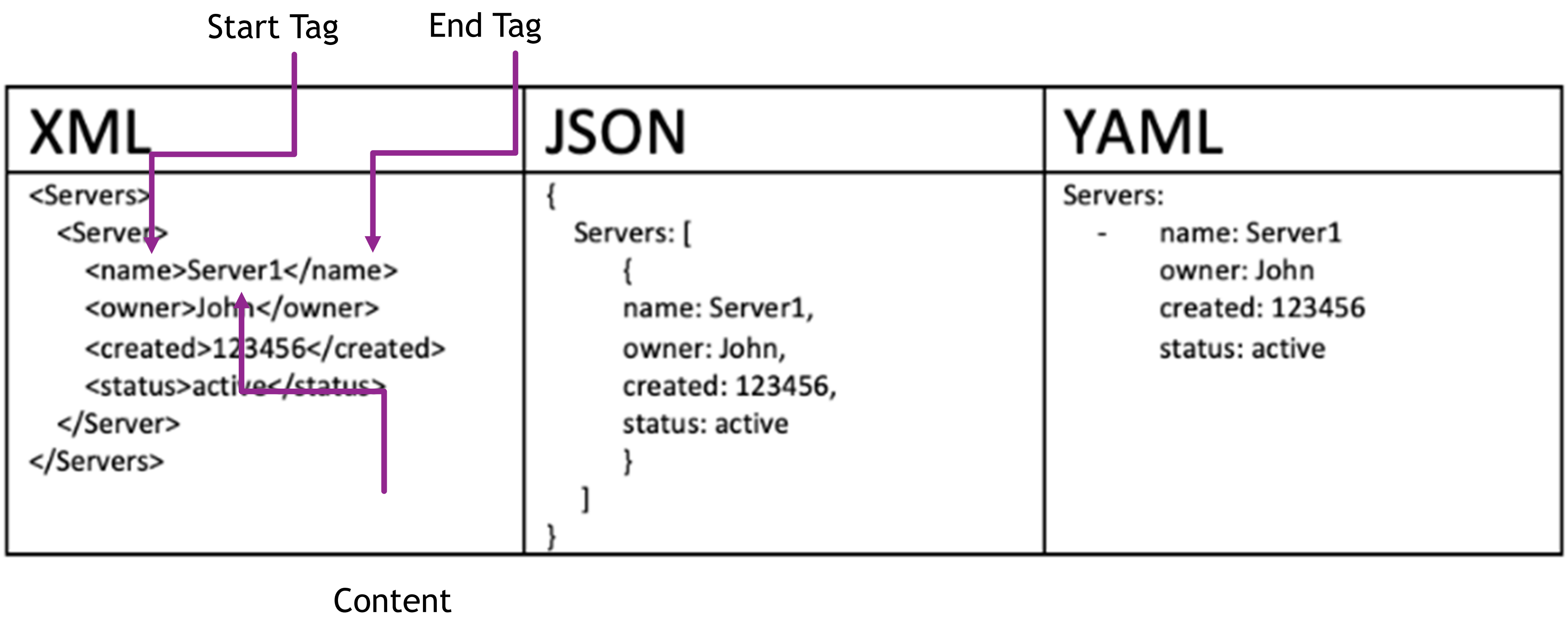
```
{
   Servers: [
        {
        name: Server1,
        owner: John,
        created: 123456,
        status: active
        }
    ]
}
```

YAML

```
Servers:
   -     name: Server1
         owner: John
         created: 123456
         status: active
```

eXtensible Markup Language

- Start Tag
- End Tag
- Content along with other nodes


Start Tag
End Tag
Content

XML	JSON	YAML

`<Servers>`
`  <Server>`
`    <name>Server1</name>`
`    <owner>John</owner>`
`    <created>123456</created>`
`    <status>active</status>`
`  </Server>`
`</Servers>`
	`{`
	`  Servers: [`
	`    {`
	`    name: Server1,`
	`    owner: John,`
	`    created: 123456,`
	`    status: active`
	`    }`
	`  ]`
	`}`
		`Servers:`
		`  -   name: Server1`
		`      owner: John`
		`      created: 123456`
		`      status: active`

Take-Aways

- Regular Expressions
- Properties of Data
    - Data Types
    - Scope, Temporality, Faithfulness, Granularity
- File Formats for Websites
    - JSON, YAML
    - XML, HTML