# DATA 301: Data Cleaning

#### Outline

- Why
- Missing Values
- Duplicates
- Strings
- Categorical data
- Numerical Data
- Dates

### Why

#### Data is usually messy.

You can minimize some problems

 For surveys, prefer comboboxes populated with a curated list rather than free form text field

#### Some you cannot

- external datasets (like your first project)
- free form text (like a collection of movie reviews)
- Missing and duplicate values
- Sensor data (outliers, missing values)

#### Either way it has to be cleaned

#### General steps

Remove duplicates

Handle missing data

Process strings

**Process Categorical data** 

**Process Numerical data** 

Normalize Data (essential if data will be used by a

machine learning algorithm)

Process dates (maybe later)

#### General steps

Remove duplicates

Handle missing data

Process strings

Did much of this when introducing project 1

**Process Categorical data** 

**Process Numerical data** 

Normalize Data (essential if data will be used by a

machine learning algorithm)

Process dates (maybe later)

#### General steps

Remove duplicates
Handle missing data

Todays topics

Process strings

**Process Categorical data** 

**Process Numerical data** 

Normalize Data (essential if data will be used by a machine learning algorithm)

Process dates (maybe later)

First see if there are any

1 df.duplicated().sum()

First see if there are any

```
1 df.duplicated().sum()
```

If so then verify them visually

```
1 df[df.duplicated()].sort_values(by='name')
```

First see if there are any

```
1 df.duplicated().sum()
```

If so then verify them visually

```
1 df[df.duplicated()].sort_values(by='name')
```

If everything looks fine, get rid of them

```
1 df.drop_duplicates(inplace=True)
```

First see if there are any

```
1 df.duplicated().sum()
```

If so then verify them visually

```
1 df[df.duplicated()].sort_values(by='name')
```

If everything looks fine, get rid of them

```
1 df.drop_duplicates(inplace=True)
```

But there could be extenuating circumstances; What if duplicate is missing data?

First see if there are any

```
1 df.duplicated().sum()
```

If so then verify them visually

```
1 df[df.duplicated()].sort_values(by='name')
```

If everything looks fine, get rid of them

```
1 df.drop_duplicates(inplace=True)
```

But there could be extenuating circumstances; What if duplicate is missing data?

Go to 31\_cleaning\_missing\_and\_duplicate\_data.ipynb

First see if there are any

1 df.duplicated().sum()

### Handle missing data (np.Nan)

199	138.423257	large	Shemeka Tweed	large
201	179.943743	large	Curtis Perry	large
202	192.245354	large	Jean Vanblarcom	large
99	110.433988	med	Marion Murphy	med
100	172.863897	med	Ronald Edwards	med
103	143.853752	med	Kathleen Ringrose	med
0	104.820189	small	Deborah Bradshaw	small
1	78.662745	small	Betty Shannon	small
2	76.240932	small	Mai Audet	small
5	112.973731	NaN	Pearl Miller	small
19	92.639737	NaN	Yvonne Arroyo	small
25	98.201594	NaN	James Dana	small

# Handle missing data (np.Nan)

	weight	t_snirt_size	name	t_snirt_size_orig
199	138.423257	large	Shemeka Tweed	large
201	179.943743	large	Curtis Perry	large
202	192.245354	large	Jean Vanblarcom	large
99	110.433988	med	Marion Murphy	med
100	172.863897	med	Ronald Edwards	med
103	143.853752	med	Kathleen Ringrose	med
0	104.820189	small	Deborah Bradshaw	small
1	78.662745	small	Betty Shannon	small
2	76.240932	small	Mai Audet	small
5	112.973731	NaN	Pearl Miller	small
19	92.639737	NaN	Yvonne Arroyo	small
25	98.201594	NaN	James Dana	small
		<b>A</b>	).	

Missing values here

### Handle missing data (np.Nan)

First the easy solution; Use sklearns SimpleImputer

					_	_	
199	138.423257	la	arge	Shemeka Tweed			large
201	179.943743	la	arge	Curtis Perry			large
202	192.245354	la	arge	Jean Vanblarcom			large
99	110.433988		med	Marion Murphy			med
100	172.863897	1	med	Ronald Edwards			med
103	143.853752	1	med	Kathleen Ringrose			med
0	104.820189	s	mall	Deborah Bradshaw			small
1	78.662745	s	mall	Betty Shannon			small
2	76.240932	s	mall	Mai Audet			small
5	112.973731		NaN	Pearl Miller			small
19	92.639737	1	NaN	Yvonne Arroyo			small
25	98.201594	1	NaN	James Dana			small
			<b>A</b>	į			

#### Handle missing data (np.Nan)

First the easy solution; Use sklearns SimpleImputer

Installed with Anaconda

from sklearn.impute import SimpleImputer

large	Shemeka Tweed	large	138.423257	199
large	Curtis Perry	large	179.943743	201
large	Jean Vanblarcom	large	192.245354	202
med	Marion Murphy	med	110.433988	99
med	Ronald Edwards	med	172.863897	100
med	Kathleen Ringrose	med	143.853752	103
small	Deborah Bradshaw	small	104.820189	0
small	Betty Shannon	small	78.662745	1
small	Mai Audet	small	76.240932	2
small	Pearl Miller	NaN	112.973731	5
small	Yvonne Arroyo	NaN	92.639737	19
small	James Dana	NaN	98.201594	25

## Handle missing data (np.Nan)

First the easy solution; Use sklearns SimpleImputer

Installed with Anaconda

1 from sklearn.impute import SimpleImputer

imp = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')

199	138.423257	large	Shemeka Tweed	large
201	179.943743	large	Curtis Perry	large
202	192.245354	large	Jean Vanblarcom	large
99	110.433988	med	Marion Murphy	med
100	172.863897	med	Ronald Edwards	med
103	143.853752	med	Kathleen Ringrose	med
0	104.820189	small	Deborah Bradshaw	small
1	78.662745	small	Betty Shannon	small
2	76.240932	small	Mai Audet	small
5	112.973731	NaN	Pearl Miller	small
19	92.639737	NaN	Yvonne Arroyo	small
25	98.201594	NaN	James Dana	small
			)	

Imputation strategy, can be mean, median (numeric only), most\_frequent or constant (numeric and strings)

Shemeka Tweed

Jean Vanblarcom

Marion Murphy

Ronald Edwards

Kathleen Ringrose

Deborah Bradshaw

Betty Shannon

Mai Audet

Pearl Miller

Curtis Perry

name t\_shirt\_size\_orig

large

large

large

med

small

small

small

small

## Handle missing data (np.Nan)

#### First the easy solution; Use sklearns SimpleImputer



5 imp = imp.fit(df\_med[['t\_shirt\_size']])

y='most\_frequent')

Imputation strategy, can be mean, median (numeric only),

most\_frequent or constant (numeric and strings)

weight t\_shirt\_size

large

large

large

small

small

NaN

138.423257

179.943743

192.245354

110.433988

172.863897

143.853752

104.820189

78.662745

76.240932

5 112.973731

Fit the imputer to the data, in this case calculate the most Frequent value seen

Shemeka Tweed

Jean Vanblarcom

Marion Murphy

Ronald Edwards

Kathleen Ringrose

Deborah Bradshaw

Betty Shannon

Mai Audet

Pearl Miller

Yvonne Arrovo

James Dana

Curtis Perry

large

large

large

small

small

NaN

NaN

name t\_shirt\_size\_orig

large

large

large

med

small

small

small

## Handle missing data (np.Nan)

#### First the easy solution; Use sklearns SimpleImputer

Installed with Anaconda

1 from sklearn.impute import SimpleImputer

imp = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')

```
5 imp = imp.fit(df_med[['t_shirt_size']])
```

Imputation strategy, can be mean, median (numeric only), most\_frequent or constant (numeric and strings)

138.423257

179 943743

192.245354

110.433988

172.863897

143.853752

104.820189

78.662745

76.240932

92 639737

98.201594

5 112.973731

Fit the imputer to the data, in this case calculate the most Frequent value seen

```
7 df_med['impute_t_shirt_size']=imp.transform(df_med[['t_shirt_size']])
```

Transform the data using the imputer, in this case calculate the most Frequent value seen and place df\_med['it in impute\_t\_shirt\_size']

Shemeka Tweed

Jean Vanblarcom

Marion Murphy

Ronald Edwards

Kathleen Ringrose

Deborah Bradshaw

Betty Shannon

Mai Audet

Pearl Miller

Yvonne Arroyo

James Dana

Curtis Perry

large

large

small

small

NaN

NaN

name t\_shirt\_size\_orig

large

large

large

med

small

small

### Handle missing data (np.Nan)

#### First the easy solution; Use sklearns SimpleImputer

imp = imp.fit(df med[['t shirt size']])

Installed with Anaconda

from sklearn.impute import SimpleImputer

imp = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')

Imputation strategy, can be mean, median (numeric only), most\_frequent or constant (numeric and strings)

138.423257

179 943743

192.245354

110.433988

172.863897

104.820189

78.662745

76.240932

92 639737

98.201594

5 112.973731

Fit the imputer to the data, in this case calculate the most Frequent value seen

```
7 df_med['impute_t_shirt_size']=imp.transform(df_med[['t_shirt_size']])
```

Transform the data using the imputer, in this case calculate the most Frequent value seen and place df\_med['it in impute\_t\_shirt\_size']

But you can usually do better than this ...

## Handle missing data (np.Nan)

What if you calculate missing values Based on weight.

199	138.423257	large	Shemeka Tweed	large
201	179.943743	large	Curtis Perry	large
202	192.245354	large	Jean Vanblarcom	large
99	110.433988	med	Marion Murphy	med
100	172.863897	med	Ronald Edwards	med
103	143.853752	med	Kathleen Ringrose	med
0	104.820189	small	Deborah Bradshaw	small
1	78.662745	small	Betty Shannon	small
2	76.240932	small	Mai Audet	small
5	112.973731	NaN	Pearl Miller	small
19	92.639737	NaN	Yvonne Arroyo	small
25	98.201594	NaN	James Dana	small
			1	

### Handle missing data (np.Nan)

What if you calculate missing values Based on weight.

#### Calculate average weight for each t-shirt size

199	138.423257	large	Shemeka Tweed	large
201	179.943743	large	Curtis Perry	large
202	192.245354	large	Jean Vanblarcom	large
99	110.433988	med	Marion Murphy	med
100	172.863897	med	Ronald Edwards	med
103	143.853752	med	Kathleen Ringrose	med
0	104.820189	small	Deborah Bradshaw	small
1	78.662745	small	Betty Shannon	small
2	76.240932	small	Mai Audet	small
5	112.973731	NaN	Pearl Miller	small
19	92.639737	NaN	Yvonne Arroyo	small
25	98.201594	NaN	James Dana	small
			J	

### Handle missing data (np.Nan)

What if you calculate missing values Based on weight.

#### Calculate average weight for each t-shirt size

		weight	t_shirt_size	name	t_shirt_size_orig
1	99	138.423257	large	Shemeka Tweed	large
2	01	179.943743	large	Curtis Perry	large
2	02	192.245354	large	Jean Vanblarcom	large
	99	110.433988	med	Marion Murphy	med
1	00	172.863897	med	Ronald Edwards	med
1	03	143.853752	med	Kathleen Ringrose	med
	0	104.820189	small	Deborah Bradshaw	small
	1	78.662745	small	Betty Shannon	small
	2	76.240932	small	Mai Audet	small
	5	112.973731	NaN	Pearl Miller	small
	19	92.639737	NaN	Yvonne Arroyo	small
	25	98.201594	NaN	James Dana	small
				J.	

#### Use that info to impute missing values based on user weight

```
#map works on a column apply works on a row, which means we have access to the entire row
3 def func(row):
       if row.t shirt size is np.NaN:
           #get a list of differences between this weight and average weights
           lst vals = [abs(row.weight-val) for val in avgs.weight]
           #get the index of the minimum value
8
9
           min val = min(lst vals)
10
           min_index=lst_vals.index(min_val)
           #return t shirt size corresponding to this index
13
           return avgs.index[min_index]
14
       #its not missing, return what's there
       return row.t shirt size
16 df better['impute t shirt size'] = df.apply(func, axis=1)
```

### Handle missing data (np.Nan)

What if you calculate missing values Based on weight.

#### Calculate average weight for each t-shirt size

	weight	t_snirt_size	name	t_snirt_size_orig
199	138.423257	large	Shemeka Tweed	large
201	179.943743	large	Curtis Perry	large
202	192.245354	large	Jean Vanblarcom	large
99	110.433988	med	Marion Murphy	med
100	172.863897	med	Ronald Edwards	med
103	143.853752	med	Kathleen Ringrose	med
0	104.820189	small	Deborah Bradshaw	small
1	78.662745	small	Betty Shannon	small
2	76.240932	small	Mai Audet	small
5	112.973731	NaN	Pearl Miller	small
19	92.639737	NaN	Yvonne Arroyo	small
25	98.201594	NaN	James Dana	small
			J.	

#### Use that info to impute missing values based on user weight

```
#map works on a column apply works on a row, which means we have access to the entire row
3 def func(row):
       if row.t shirt size is np.NaN:
           #get a list of differences between this weight and average weights
           lst vals = [abs(row.weight-val) for val in avgs.weight]
           #get the index of the minimum value
           min val = min(lst vals)
10
           min_index=lst_vals.index(min_val)
           #return t shirt size corresponding to this index
13
           return avgs.index[min_index]
14
       #its not missing, return what's there
       return row.t shirt size
16 df better['impute t shirt size'] = df.apply(func, axis=1)
```

#### Go to 31\_cleaning\_missing\_and\_duplicate\_data.ipynb

# Categorical data

Coming shortly