ECE5984 – Applications of Machine Learning
Lecture 8 – Data Preparation

Creed Jones, PhD

Course Updates

- Quiz 2 was (is) today
- At the end of the semester, I will replace your lowest quiz grade with your next lowest grade
- HW2 is posted
  - Due on Tuesday, February 15!

Here are some useful documentation links

- Scikit-learn API
- https://scikit-learn.org/stable/modules/classes.html

- Numpy
- https://numpy.org/doc/stable/index.html

- Pandas
- https://pandas.pydata.org/docs/reference/index.html

Model development usually follows this sequence of steps

1. Check for data errors / anomalies
2. Examine and replace missing values
3. Add computed fields
4. Normalize
5. Partition into training/test/validation
6. Feature selection
7. Train model
8. Measure model performance

Deep model development follows a similar set of steps – with a couple of distinctions

1. Check for data errors / anomalies
2. Examine and replace missing values
3. Add computed fields
4. Normalize
5. Partition into training/test/validation
6. Feature selection
7. Train model
8. Measure model performance

---

1. Check for data errors / anomalies
2. Normalize
3. Partition into training/test/validation
4. Augment training set
5. Train model
6. Measure model performance

Different variable types and roles have different methods for replacement of missing values

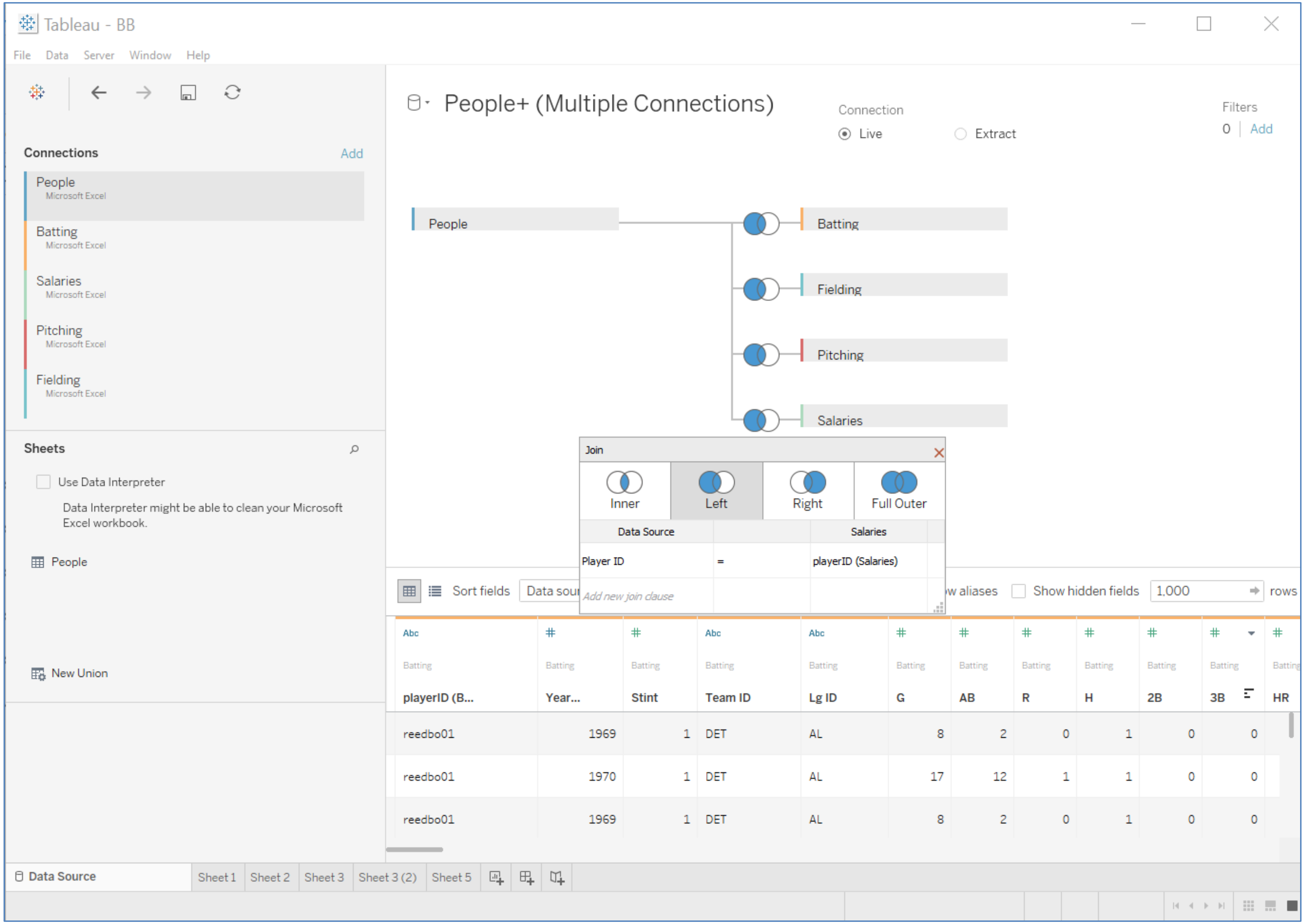
Field Type	Variable Type	If Any Missing Values	Delete Column?	Imputation methods					
ID	Any	replace with unique value	no						
Feature (Predictor)	Numeric	impute	if the "vast majority" are missing	zero	population mean	population median	kNN, SMOTE, et al	stratified mean or median	
	Ordinal (ordinal categorical)	impute				population median	kNN	stratified mode or median	
	Interval	impute				population median	kNN	stratified mode or median	
	Categorical (unordered categorical)	impute				population mode	kNN	stratified mode	"UNK“ (new category)
	Binary	impute				population mode	kNN	stratified mode	
	Text	leave blank or replace with "UNK"							
Target	Any	delete row	no						

Today's Objectives

Joining Multiple Data Sources in Tableau

Data Quality Report

3.6 – Data Preparation
- 3.6.1 Normalization
  - Range normalization
  - Mean-sigma normalization
- 3.6.2 Binning
  - Equal-width binning
  - Equal-frequency binning
- 3.6.3 Sampling
- Handling Time-series data



Note the *left join* to include all records in the left table (Players) with data from the right table <u>if it is present</u>

Inner join will only keep records present in both; outer keeps those in either

Creating an extract can speed up subsequent work

The Adult Income Data set (on Canvas) looks like this…

Name	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Target
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K
37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States	>50K
30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	India	>50K
23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0	30	United-States	<=50K
32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0	50	United-States	<=50K
40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	0	0	40	?	>50K
34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	Amer-Indian-Eskimo	Male	0	0	45	Mexico	<=50K
25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing	Own-child	White	Male	0	0	35	United-States	<=50K
32	Private	186824	HS-grad	9	Never-married	Machine-op-inspct	Unmarried	White	Male	0	0	40	United-States	<=50K
38	Private	28887	11th	7	Married-civ-spouse	Sales	Husband	White	Male	0	0	50	United-States	<=50K
43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarried	White	Female	0	0	45	United-States	>50K
40	Private	193524	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	60	United-States	>50K
54	Private	302146	HS-grad	9	Separated	Other-service	Unmarried	Black	Female	0	0	20	United-States	<=50K

A Data Quality Report, in Excel

nColumns =	15													
nRows =	32561													
Name	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Target
TYPE	numeric	text	numeric	text	numeric	text	text	text	text	text	numeric	numeric	numeric	text
MAX	90	0	1484705	0	16	0	0	0	0	0	99999	4356	99	0
75PCT	48	#NUM!	237058	#NUM!	12	#NUM!	#NUM!	#NUM!	#NUM!	#NUM!	0	0	45	#NUM!
MEAN	38.58165	0	189778.3665	0	10.08067934	0	0	0	0	0	1077.648844	87.30382973	40.43745585	0
MEDIAN	37	#NUM!	178356	#NUM!	10	#NUM!	#NUM!	#NUM!	#NUM!	#NUM!	0	0	40	#NUM!
MODE	36	#N/A	123011	#N/A	9	#N/A	#N/A	#N/A	#N/A	#N/A	0	0	40	#N/A
25PCT	28	#NUM!	117821.5	#NUM!	9	#NUM!	#NUM!	#NUM!	#NUM!	#NUM!	0	0	40	#NUM!
MIN	17	0	12285	0	1	0	0	0	0	0	0	0	1	0
RANGE	73	0	1472420	0	15	0	0	0	0	0	99999	4356	98	0
STDEV	13.64043	#DIV/0!	105549.9777	#DIV/0!	2.572720332	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	7385.292085	402.9602186	12.34742868	#DIV/0!
NBLANK	32561	0	32561	0	32561	0	0	0	0	0	32561	32561	32561	0
NZERO	0	0	0	0	0	0	0	0	0	0	29849	31042	0	0
NNEGATIVE	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N"?"	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N"#NA"	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NMEDIAN	858	0	2	0	7291	0	0	0	0	0	29849	31042	15217	0
NMODE	898	0	13	0	10501	0	0	0	0	0	29849	31042	15217	0
SKEW	0.558743	#DIV/0!	1.446980095	#DIV/0!	-0.311675868	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	11.95384769	4.594629122	0.227642537	#DIV/0!
KURTOSIS	-0.166127	#DIV/0!	6.218810978	#DIV/0!	0.623444075	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	154.7994379	20.37680171	2.916686796	#DIV/0!

The quality metrics can be computed using Excel formulas; this is maybe not the best way (cardinality is missing) but it's an easy way

- `nRows =MAX(COUNTA(data!A:A),COUNTA(data!B:B),COUNTA(data!C:C)…)`
- `nColumns =MAX(COUNTA(data!A1:ZZ1),COUNTA(data!A2:ZZ2),…)`
- `TYPE =SWITCH(TYPE(data!A$2),1,"numeric",2,"text",3,"binary",4,"error",5,"array")`
- `MAX =MAX(OFFSET(data!A$2, 0, 0, nRows, 1))`
- `75PCT =QUARTILE.EXC(OFFSET(data!A$2, 0, 0, nRows, 1), 3)`
- `MEDIAN =QUARTILE.EXC(OFFSET(data!A$2, 0, 0, nRows, 1),2)`
- `MODE =MODE(OFFSET(data!A$2, 0, 0, nRows, 1))`
- `RANGE =B6-B12`
- `NBLANK =COUNTBLANK(OFFSET(data!A$2, 0, 0, nRows, 1))`
- `NNEGATIVE =COUNTIF(OFFSET(data!A$2, 0, 0, nRows, 1),"<0")`
- `NMEDIAN =COUNTIF(OFFSET(data!A$2, 0, 0, nRows, 1),B9)`

A Data Quality Report on a simple test data set

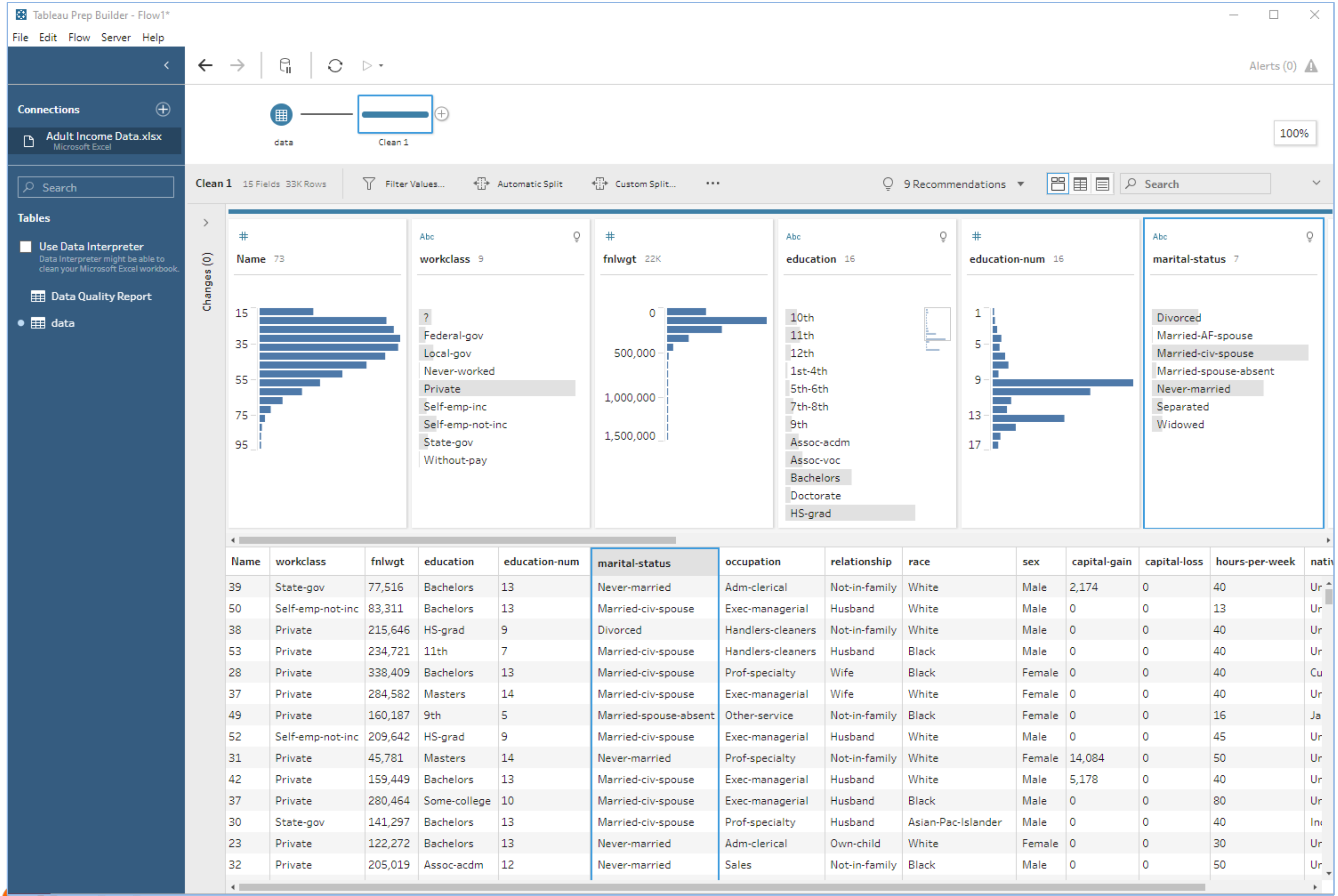
playerID	yearID	B	C	Salary
aa	2020	12	1	100
ab	2020	13		110
ac	2021	10		90
ad	2021	9	4	100
ae	2020	14	5	110
af	2020	13	6	90
ag	2020		7	100
ah	2021	8	8	110
ai	2021		9	90
aj	2021		0	100

```
File C:/Data/Baseball/test.xlsx is of size (10, 5)

           stat playerID       yearID          B         C      Salary
0   cardinality       10     2.000000   6.000000  8.000000    3.000000
1          mean      N/A  2020.500000  11.285714  5.000000  100.000000
2        median      N/A  2020.500000  12.000000  5.500000  100.000000
3   n_at_median      N/A     0.000000   1.000000  0.000000    4.000000
4          mode       aa  2020.000000  13.000000  0.000000  100.000000
5     n_at_mode        1     5.000000   2.000000  1.000000    4.000000
6        stddev      N/A     0.527046   2.288689  3.207135    8.164966
7           min      N/A  2020.000000   8.000000  0.000000   90.000000
8           max      N/A  2021.000000  14.000000  9.000000  110.000000
9         nzero        0     0.000000   0.000000  1.000000    0.000000
10     nmissing        0     0.000000   3.000000  2.000000    0.000000
```

So what do we do with a data quality report?

- Check for things to be fixed
  - Missing values
  - Improper cardinality
  - Outliers – investigate extreme points using histograms
- See if certain columns need to be dropped
- Remind us to do data normalization



VIRGINIA TECH.

BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Tableau Prep Builder also does a quality report (and it's much easier, though not as flexible)

Proper normalization of the data usually leads to better model training and performance

- In real-world data sources, continuous features often have very different numeric ranges
  - A feature representing customer ages might cover the range [16, 96], whereas a feature representing customer salaries might cover the range [10,000, 100,000].
- Range normalization (or min-max normalization) equalizes the range of all variables

	HEIGHT			SPONSORSHIP EARNINGS		
	Values	Range	Standard	Values	Range	Standard
	192	0.500	-0.073	561	0.315	-0.649
	197	0.679	0.533	1,312	0.776	0.762
	192	0.500	-0.073	1,359	0.804	0.850
	182	0.143	-1.283	1,678	1.000	1.449
	206	1.000	1.622	314	0.164	-1.114
	192	0.500	-0.073	427	0.233	-0.901
	190	0.429	-0.315	1,179	0.694	0.512
	178	0.000	-1.767	1,078	0.632	0.322
	196	0.643	0.412	47	0.000	-1.615
	201	0.821	1.017	1111	0.652	0.384
Max	206			1,678		
Min	178			47		
Mean	193			907		
Std. Dev.	8.26			532.18		

$$a_i' = \frac{a_i - min(a)}{max(a) - min(a)} \times (high - low) + low$$

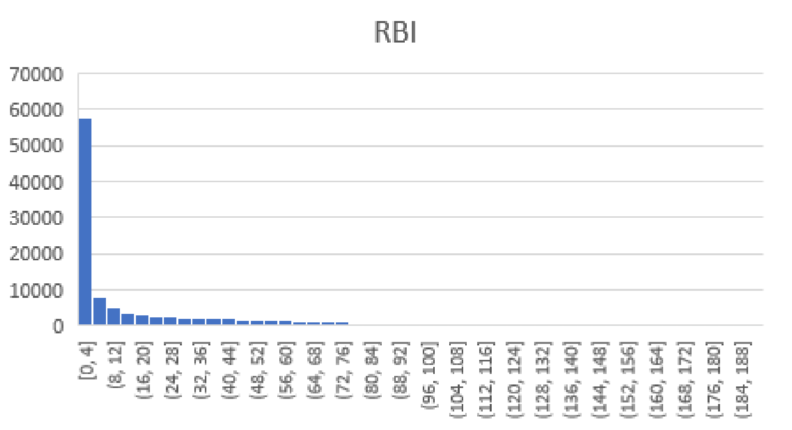
Standard score normalization (or mean-sigma normalization) transforms features to the same mean and standard deviation (often 0 and 1)

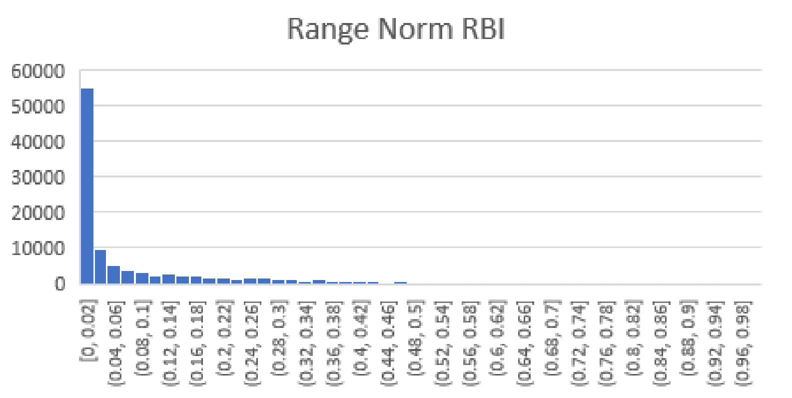
- Based on the presumption that the data is normally distributed, or close anyway
- For each column, subtract that column's mean and divide by its standard deviation

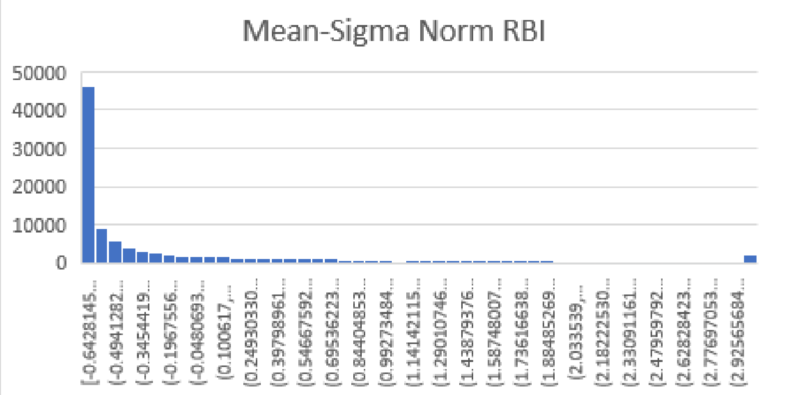
	HEIGHT			SPONSORSHIP EARNINGS		
	Values	Range	Standard	Values	Range	Standard
	192	0.500	-0.073	561	0.315	-0.649
	197	0.679	0.533	1,312	0.776	0.762
	192	0.500	-0.073	1,359	0.804	0.850
	182	0.143	-1.283	1,678	1.000	1.449
	206	1.000	1.622	314	0.164	-1.114
	192	0.500	-0.073	427	0.233	-0.901
	190	0.429	-0.315	1,179	0.694	0.512
	178	0.000	-1.767	1,078	0.632	0.322
	196	0.643	0.412	47	0.000	-1.615
	201	0.821	1.017	1111	0.652	0.384
Max	206			1,678		
Min	178			47		
Mean	193			907		
Std Dev	8.26			532.18		

E2 =(A2-minRBI)/(maxRBI-minRBI)*(highMap-lowMap)+lowMap

	A	B	C	D	E	F
1	RBI				Range Norm RBI	Mean-Sigma Norm RBI
2	0	Min	0		0	-0.642814538
3	13	Max	191		0.068062827	-0.149208214
4	19	Mean	16.92966355		0.09947644	0.07861009
5	27	Sigma	26.33677762		0.141361257	0.382367828
6	16	high	1		0.083769634	-0.035299062
7	5	low	0		0.02617801	-0.452965952
8	2				0.010471204	-0.566875104
9	34				0.178010471	0.648155849
10	1				0.005235602	-0.604844821
11	11				0.057591623	-0.225147649
12	18				0.094240838	0.040640372
13	0				0	-0.642814538
14	1				0.005235602	-0.604844821
15	5				0.02617801	-0.452965952
16	21				0.109947644	0.154549524
17	23				0.120418848	0.230488959
18	0				0	-0.642814538
19	0				0	-0.642814538
20	8				0.041884817	-0.3390568
21	0				0	-0.642814538
22	13				0.068062827	-0.149208214
23	24				0.12565445	0.268458676
24	21				0.109947644	0.154549524
25	0				0	-0.642814538
26	14				0.073298429	-0.111238497
27	10				0.052356021	-0.263117366
28	18				0.094240838	0.040640372
29	16				0.083769634	-0.035299062
30	2				0.010471204	-0.566875104
31	26				0.136125654	0.34439811
32	30				0.157068063	0.49627698
33	2				0.010471204	-0.566875104
34	3				0.015706806	-0.528905387
35	15				0.078534031	-0.073268779
36	18				0.094240838	0.040640372
37	0				0	-0.642814538







Note the use of a min-max scaler in the linear fitting routine I introduced last time

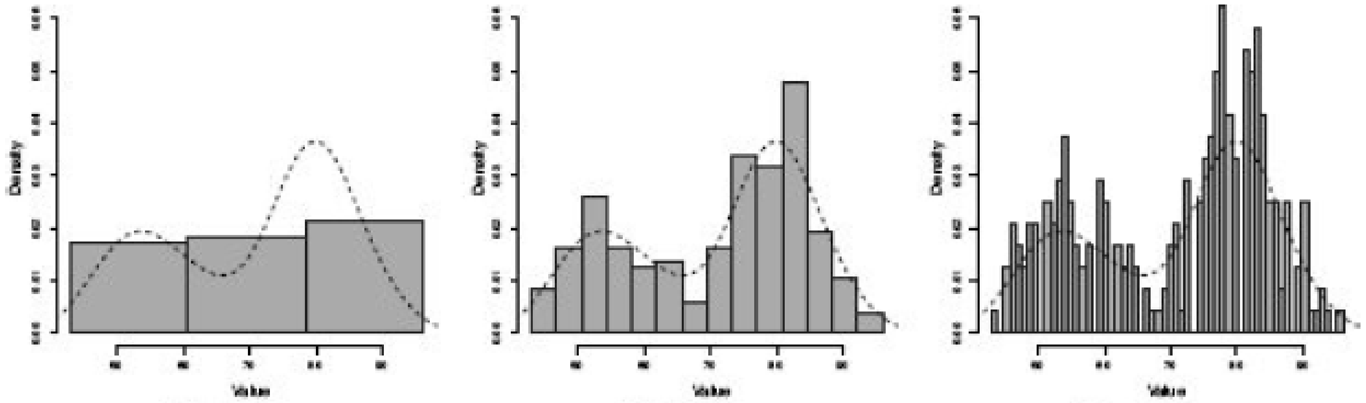
```
def trylinearfit(method, rawpred, targ, imputer):
    ranseed = 98043
    imputer.fit(rawpred)
    newpred = imputer.transform(rawpred)
    scaler = skpreproc.MinMaxScaler(feature_range=(-1, 1))
    normpred = scaler.fit_transform(newpred)
    xtrain, xtest, ytrain, ytest = skmodelsel.train_test_split(normpred, targ, test_size=0.3,
                                                                random_state=ranseed)
    model = sklinear_model.LinearRegression()
    regr = model.fit(xtrain, ytrain)
    print("Method={0}, training set R-sq={1:8.5f},  test set MSE={2:e}".format(method,
            regr.score(xtrain, ytrain), sk.metrics.mean_squared_error(ytest, regr.predict(xtest))))
```

The effect of outliers can be reduced by using a clamp transformation – this code limits all features to 200 (usually would do this column by column)

```
def trylinearfit(method, rawpred, targ, imputer):
    ranseed = 98043
    imputer.fit(rawpred)
    newpred = np.minimum(newpred, 200)
    newpred = imputer.transform(rawpred)
    scaler = skpreproc.MinMaxScaler(feature_range=(-1, 1))
    normpred = scaler.fit_transform(newpred)
    xtrain, xtest, ytrain, ytest = skmodelsel.train_test_split(normpred, targ, test_size=0.3,
                                                                        random_state=ranseed)
    model = sklinear_model.LinearRegression()
    regr = model.fit(xtrain, ytrain)
    print("Method={0}, training set R-sq={1:8.5f},  test set MSE={2:e}".format(method,
                regr.score(xtrain, ytrain), sk.metrics.mean_squared_error(ytest, regr.predict(xtest))))
```

Binning is the process of assigning a continuous variable to a categorical value – to mitigate noise and to allow use in stratifying

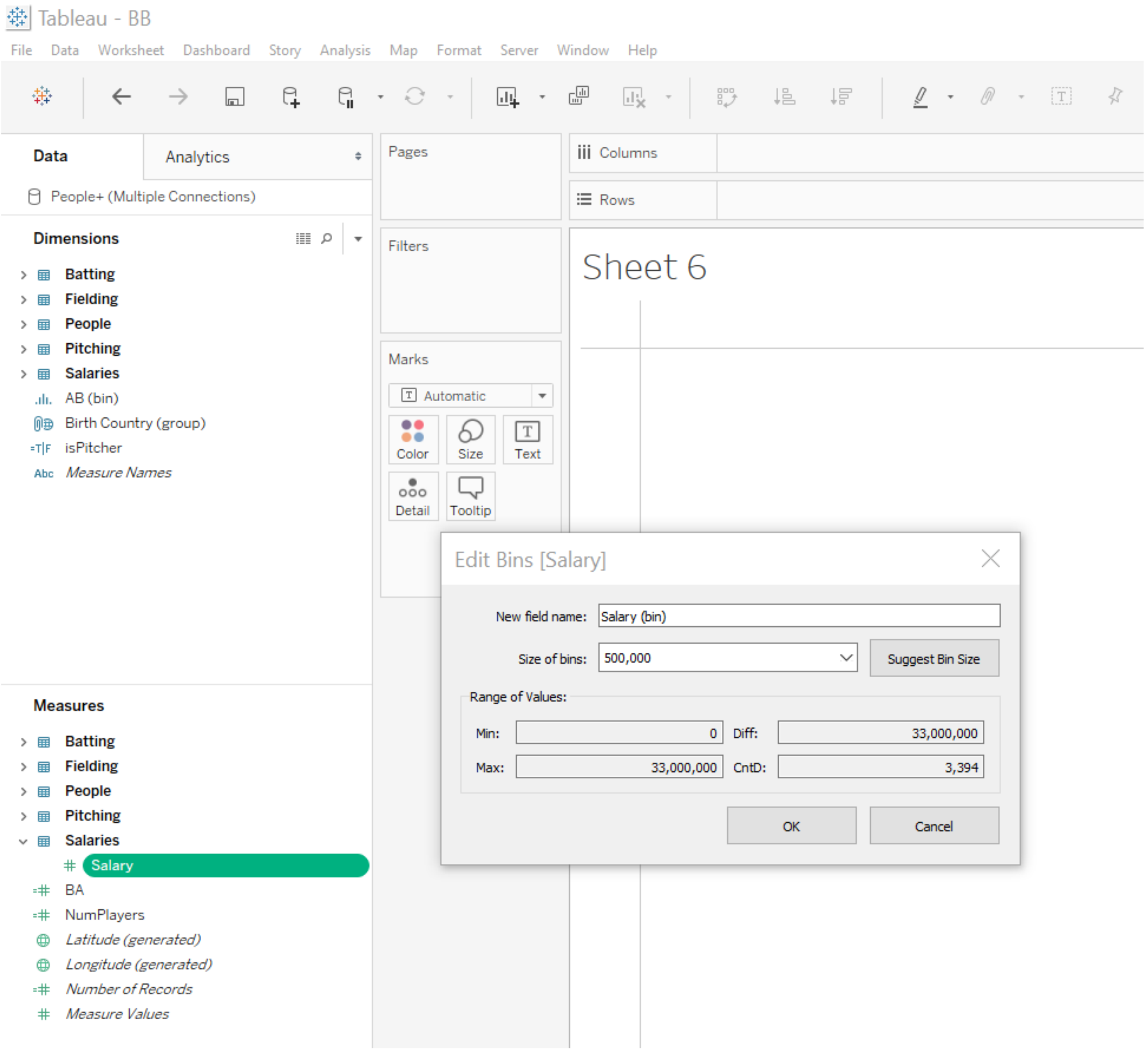
- Equal-width binning (0-10, 11-20, 21-30, etc.)
- Equal-frequency binning (lowest 10%, next 10%, etc)
- Often we will keep both the original continuous variable *and* the binned result as possible modeling features
- Need to determine the proper number of bins



(a) 3 bins (b) 14 bins (c) 60 bins

A fixed number of bins or fixed bin size can be used in Tableau

Often, fixed bin sizes will make the graphs easier to read (nice even numbers for bin boundaries)



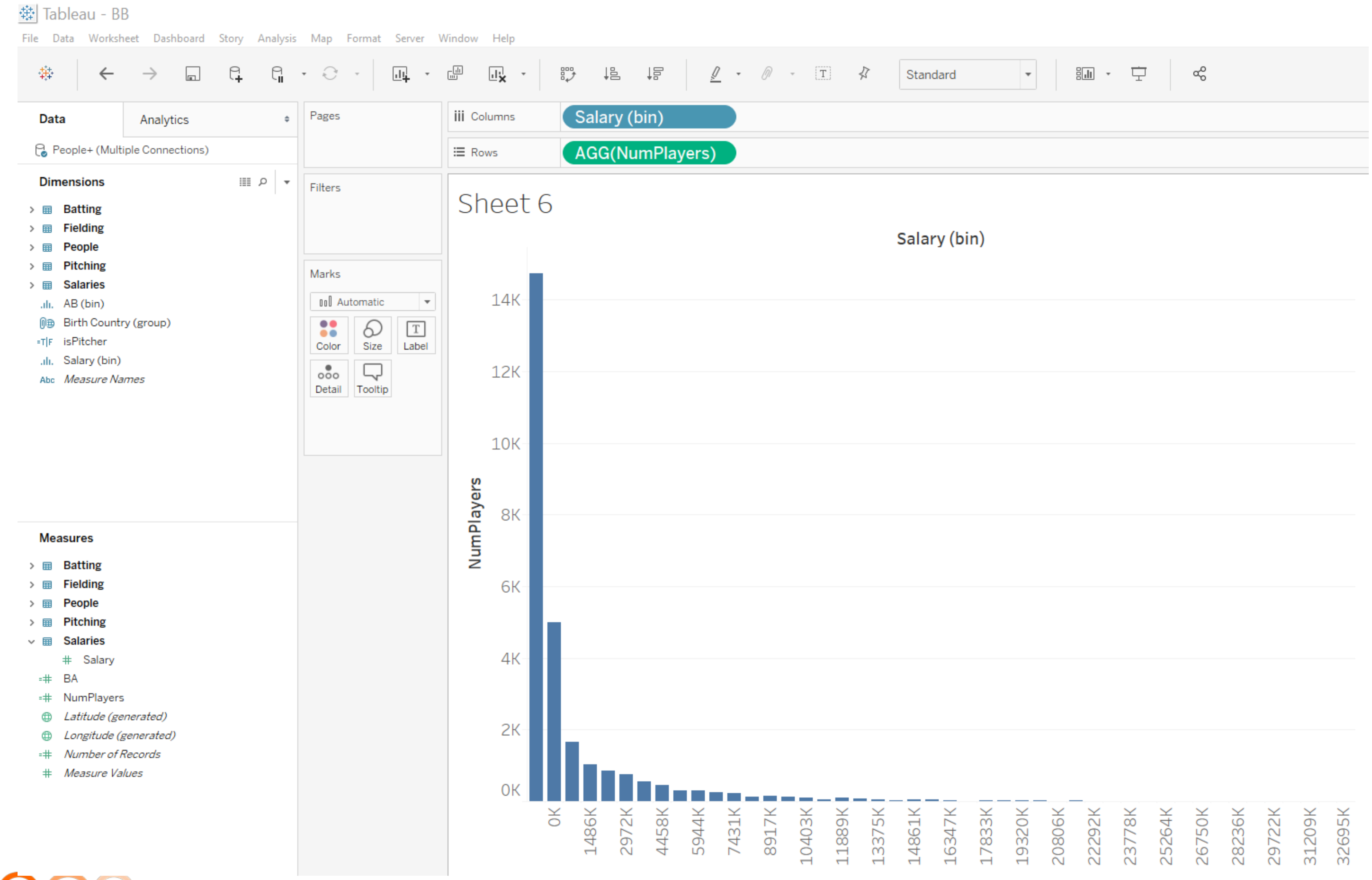
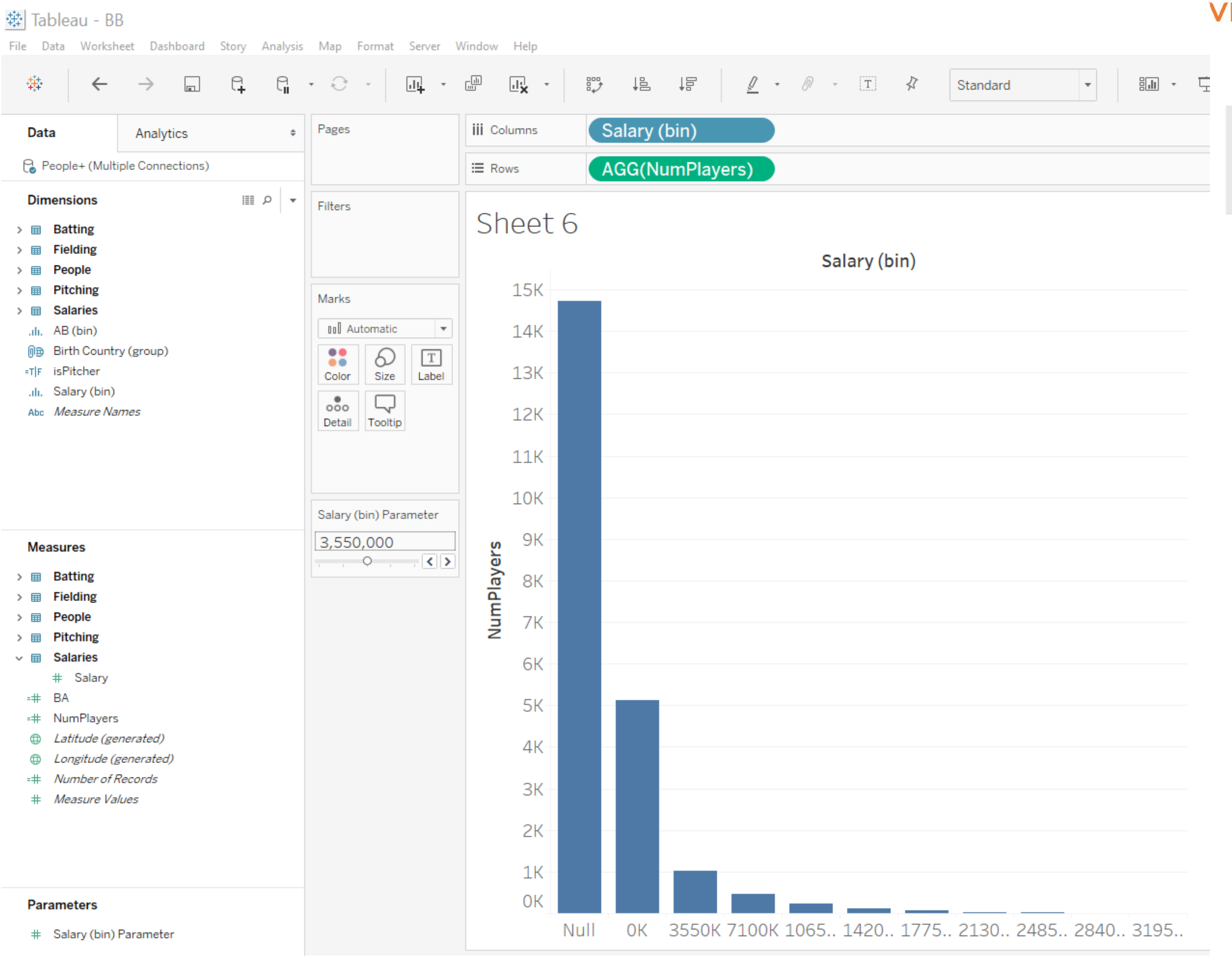


Tableau - BB
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Data
Analytics
People+ (Multiple Connections)
Dimensions
Batting
Fielding
People
Pitching
Salaries
AB (bin)
Birth Country (group)
isPitcher
Salary (bin)
Measure Names
Measures
Batting
Fielding
People
Pitching
Salaries
Salary
BA
NumPlayers
Latitude (generated)
Longitude (generated)
Number of Records
Measure Values
Pages
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Columns
Salary (bin)
Rows
AGG(NumPlayers)
Sheet 6
Salary (bin)
NumPlayers
14K
12K
10K
8K
6K
4K
2K
0K
0K
1486K
2972K
4458K
5944K
7431K
8917K
10403K
11889K
13375K
14861K
16347K
17833K
19320K
20806K
22292K
23778K
25264K
26750K
28236K
29722K
31209K
32695K

It's possible to change the bin width – even to make it set by a *Parameter* (which can be controlled by a slider on the UI)



Categorical variables are a bit of a problem They have different values but no specific order

- We could encode them as Alabama=1, Alaska=2
- But that makes no sense mathematically
  - What is the meaning of 2*Alaska?
- Even the ordering is odd
  - Do we order by population? Date of admission to the union?

- One thing we can do is create binary flags (a mapping) for each possible value
- This is *One-Hot Encoding* or Binary Encoding

UserName	TimeZone
Bob33	Eastern
StarGazer	Central
u?j_yy3$	Pacific
GrandmaJ	Mountain
PeterParker	Eastern

➔

UserName	TZ_Eastern	TZ_Central	TZ_Mountain	TZ_Pacific
Bob33	1	0	0	0
StarGazer	0	1	0	0
u?j_yy3$	0	0	0	1
GrandmaJ	0	0	1	0
PeterParker	1	0	0	0

One-Hot encoding is often done for categoricals, so that we don't interpret them as ordinals!

- There is a practical limit on how many different categories can be transformed into binaries
  - In my experience, the US states/territories (54-ish) is about the limit

Convert the categorical data

	Age	Workclass
1	41	Private
2	33	State-gov
3	27	Federal-gov

Get dummy

	Age	Work Class: Private	Work Class: State-gov	Work Class: Federal-gov
1	41	1	0	0
2	33	0	1	0
3	27	0	0	1

It's common to create an "other" bin for rare cases:

ICECREAM = {iceCreamChoc, iceCreamVan, iceCreamStraw, iceCreamOther}

A column in a pandas dataframe can be one-hot encoded using the get_dummies() function

```
pandas.get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, columns=None,
                   sparse=False, drop_first=False, dtype=None)
```

- See https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

- Can encode all non-numerical columns, or specify which ones
- Inputs determine the prefix for generating column names
- One category can be a dummy (since it's redundant; one category <u>must</u> be 1)
- The output datatype can be specified

scikit-learn contains the OneHotEncoder class for performing this transformation

```
OneHotEncoder(categories='auto', drop=None, sparse=True,
                    dtype=<class 'numpy.float64'>, handle_unknown='error')
```

- See https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

- Categories can be generated automatically or given as an input
- One category can be “dropped” (since it’s redundant; one category <u>must</u> be 1)
- The output datatype can be specified
- Any missing values can be represented by all zeroes in the one-hot outputs – or will generate an error (the default behavior)

Sometimes the dataset we have is so large that we do not use all the data available to us in an ABT and instead *sample* a smaller percentage from the larger dataset

- We need to be careful when sampling, however, to ensure that the resulting datasets are still representative of the original data and that no unintended bias is introduced during this process.

- Common forms of sampling include:
  - top sampling
  - random sampling
  - stratified sampling
  - under-sampling
  - over-sampling

When we only deal with part of the dataset, think about how to choose the instances

- **Top sampling** simply selects the top s% of instances from a dataset to create a sample
  - It can introduce bias dependent on the order of the data – <u>don't do it</u>
- **Random sampling** randomly selects a proportion of s% of the instances from a large dataset to create a smaller set.
  - The most common practice
- **Stratified sampling** ensures that the relative frequencies of the levels of a specific stratification feature are maintained in the sampled dataset.
  - The instances in a dataset are divided into groups containing only instances that have a particular level for the stratification feature
  - s% of the instances in each stratum are randomly selected
  - these selections are combined to give an overall sample of s% of the original dataset.

Sometimes we want to modify the proportion of the data set having a particular value or values; this calls for under-sampling or over-sampling

Under-sampling begins by dividing a dataset into groups, containing only instances that have a particular level for the feature to be under-sampled.

- The number of instances in the smallest group is the under-sampling target size.
- Each group containing more instances than the smallest one is then randomly sampled by the appropriate percentage to create a subset that is the under-sampling target size.
- These under-sampled groups are then combined to create the overall under-sampled dataset.

Over-sampling addresses the same issue as under-sampling but in the opposite way.

- After dividing the dataset into groups, the number of instances in the largest group becomes the over-sampling target size.
- From each smaller group, we then create a sample containing that number of instances using *random sampling with replacement (or SMOTE).*
- These larger samples are combined to form the overall over-sampled dataset.

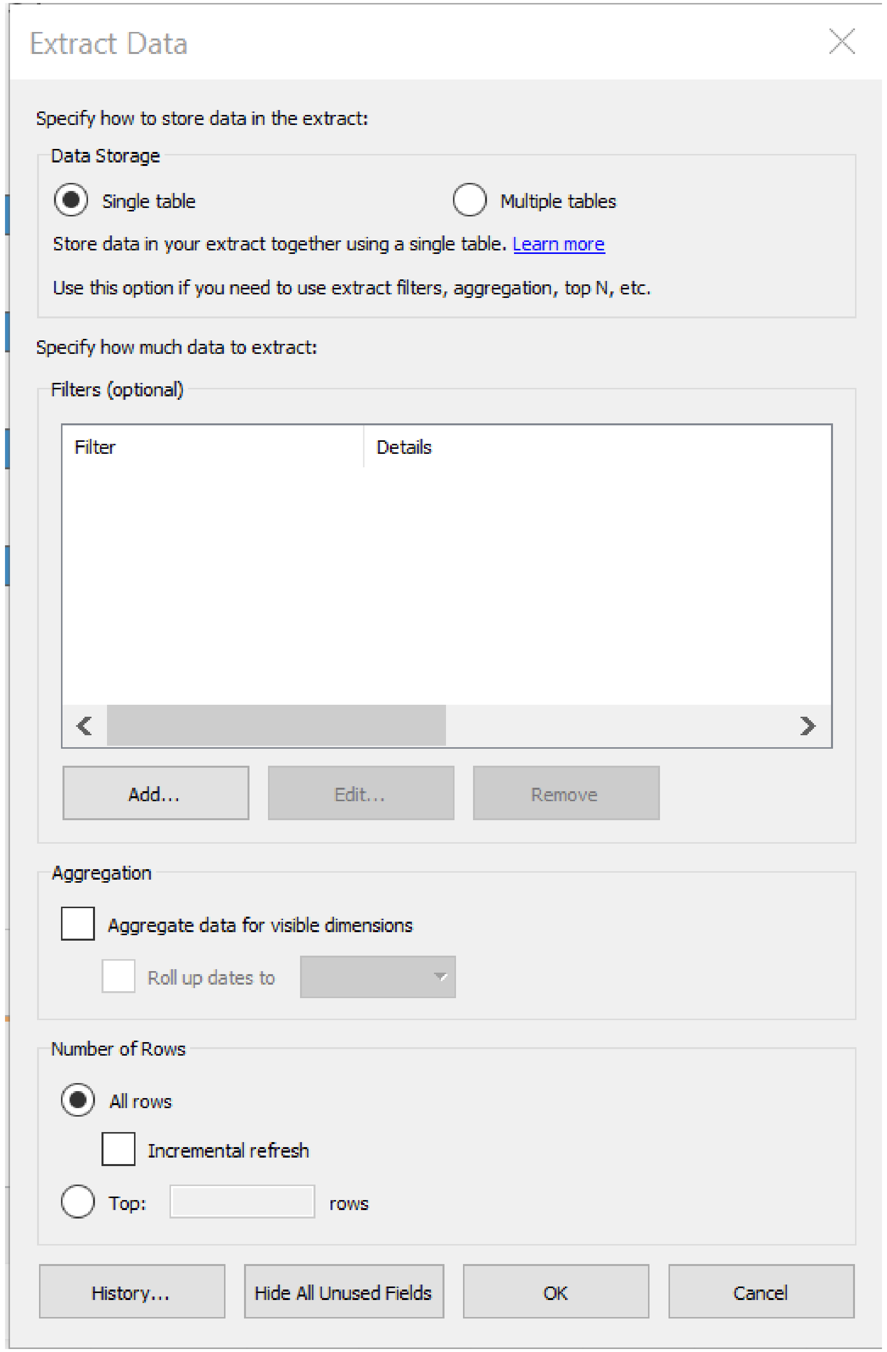
Sampling and balancing in sklearn can be accomplished using model_selection.train_test_split() and

```
sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None,
                                random_state=None, shuffle=True, stratify=None)
```

- As we will see, it's common to split the training data into either:
  - two groups, train and test (70-30? 80-20?)
  - three groups, train, test and validate (60-20-20?)

- Splitting can be random or stratified
  - stratification generally accomplishes balancing by *down-sampling*

In Tableau, we can work with a sample of the full dataset; this panel only shows using top sampling; to achieve other methods, add a randomly generated key to the data and set a programmable threshold on it



Use of time-series data can be straightforward or complex

- A simple approach is to convert timestamped events into summary statistics for a number of time periods
  - Sum, average, etc…
- Processing of missing values needs special consideration
  - Missing probably means no activity

- To use more comprehensive information from time series data, trends can be modeled
  - Here is a starting point to this topic: http://www.statsoft.com/textbook/time-series-analysis

Turning time-series data into period-based data will usually shorten but widen the dataset

- The fields in the transformed version (at the bottom) can be directly used in most modeling approaches
- This is an opportunity to employ a number of mapping approaches
  - Sums
  - Trends
  - Apply seasonal corrections
  - Disable certain time periods

UserID	Date	LoginTime	Activity
A321	1/5/2020	321	Browse
A321	1/5/2020	42	Download
A350	1/5/2020	222	Download
A300	1/6/2020	12	Browse
A321	1/6/2020	42	Browse
A350	1/6/2020	12	Download
A300	1/6/2020	55	Download
A321	1/7/2020	10	Browse

UserID	Jan-5-Dload	Jan-5-Browse	Jan-6-Dload	Jan-6-Browse	Jan-7-Dload	Jan-7-Browse	DaysOnline
A300	0	0	55	12	0	0	1
A321	42	321	0	42	0	10	3
A350	222	0	12	0	0	0	2

Today’s Objectives

Joining Multiple Data Sources in Tableau

Data Quality Report

3.6 – Data Preparation

- 3.6.1 Normalization
  - Range normalization
  - Mean-sigma normalization
- 3.6.2 Binning
  - Equal-width binning
  - Equal-frequency binning
- 3.6.3 Sampling
- Handling Time-series data