

Data Preparation

UBCO Master of Data Science – DATA 542

Fatemeh Fard



Lecture 3 - review

Data cleaning

Handling missing values

Data preparation

Prepare data for analysis

- Remove duplicates
- Replace values
- Filter outliers
- Rename Axis
- Binning

Discuss rationales for each action

Data preparation

Prepare data for analysis

- Remove duplicates
- Replace values
- Filter outliers
- Rename Axis
- Binning

- perform better statistics
- balanced dataset
- Null replacement
- String with numbers
- Standardize
- Renaming axis \rightarrow clear or easier data processing
- Transforming: $F \leftrightarrow C$
inch \leftrightarrow cm

Discuss rationales for each action

Question Try It

Question 1: What is data?

```
data = pd.DataFrame({'k1': ['one', 'two'] * 3 + ['two'],
                     'k2': [1, 1, 2, 3, 3, 4, 4]})
```

	k1	k2
0	1	1
1	2	1
2	1	2
3	2	3
4	1	3
5	2	4
6	2	4

A)

	k1	k2
0	one	1
1	two	1
2	one	2
3	two	3
4	one	3
5	two	4
6	two	4

B)

	0	1	2	3	4	5	6
k1	one	two	one	two	one	two	two
k2	1	1	2	3	3	4	4

C)

	k1	k2
0	5	1
1	8	1
2	5	2
3	8	3
4	5	3
5	8	4

D)

Duplicates

Investigate duplicates in the data:

```
data = pd.DataFrame(  
    {'k1': ['one', 'two'] * 3 + ['two'],  
    'k2': [1, 1, 2, 3, 3, 4, 4]})
```

data.duplicated(keep=) **Returns Boolean value**

keep argument is by default the first
observation

keep = 'last' will return the last observation

Duplicates cont.

```
data.duplicated()
```

```
0    False
1    False
2    False
3    False
4    False
5    False
6     True
dtype: bool
```

```
data.duplicated(keep='last')
```

```
0    False
1    False
2    False
3    False
4    False
5     True
6    False
dtype: bool
```

Drop duplicates

Drop duplicate observations in the data, or filter it by each column

```
data.drop_duplicates([column_names], keep)
```

← Optional →

```
data['v1'] = range(7)
```

```
data.drop_duplicates(['k1', 'k2'], keep='last')
```

```
data.drop_duplicates(['k1'], keep='last')
```

	k1	k2	v1
4	one	3	4
6	two	4	6

	k1	k2	v1
0	one	1	0
1	two	1	1
2	one	2	2
3	two	3	3
4	one	3	4
5	two	4	5
6	two	4	6

Replacing values

Replace values using lists or dictionaries

```
data.replace(value, replacement)
```

value and replacement can be lists

```
data.replace({value:replacement, value:replacement})
```

```
data.replace([-999, -1000], np.nan)
```

```
data.replace([-999, -1000], [np.nan, 0],  
inplace=True)
```

inplace means changes are done
on original dataframe

Try IT

Use the first dataframe you created and apply duplicate functions we learned. Drop duplicates, and replace values.

Try IT

Question 1: Create the following Dataframe:

Hint: Use numpy arrays functions for the values

```
pd.DataFrame()  
np.arange()  
np.reshape()
```

	one	two	three	four
Ohio	0	1	2	3
Colorado	4	5	6	7
New York	8	9	10	11

Renaming axis indices

Three options to rename axis indices:

	one	two	three	four
Ohio	0	1	2	3
Colorado	4	5	6	7
New York	8	9	10	11

```
data.rename(index=str.title, columns=str.upper)
data.rename(index={'Ohio': 'INDIANA'},
             columns={'three': 'peekaboo'})
data.rename(index={'Ohio': 'INDIANA'},
            inplace=True)
```

Try It

Renaming axis indices

Create, modify or rename new/existing axis labels with `map` or `rename`

	one	two	three	four
Ohio	0	1	2	3
Colorado	4	5	6	7
New York	8	9	10	11

Inplace assignment:

```
data.columns = data.columns.map(lambda x:
x.upper() )
```

```
data.index = data.index.map(lambda x:
x.upper() )
```

Try It

Anonymous Function

Anonymous function is a function that is defined without a name.

Use it as an argument to a higher-order function (a function that takes in other functions as arguments), such as map, filter

```
def keyword => lambda keyword
```

```
lambda arguments: expression
```

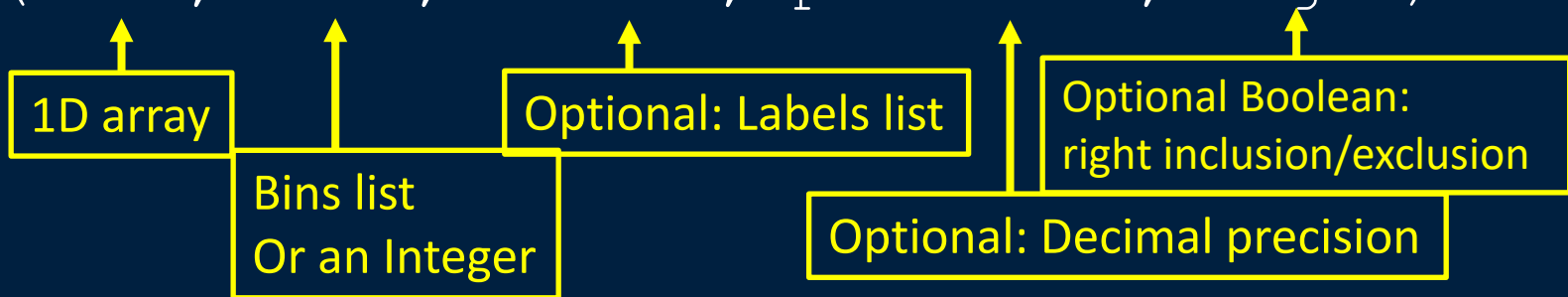
```
def double(x):  
    return x * 2
```

```
lambda x: x * 2
```

Discretization and Binning

Need for binning into categories, with pandas `cut`

```
pd.cut(data, bins, labels, precision, right)
```



```
ages = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45,
        41, 32]
```

Binning cont.

```
ages = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45,
41, 32]
```

```
bins = [18, 25, 35, 60, 100]
```

```
cats = pd.cut(ages, bins)
```

```
[(18, 25], (18, 25], (18, 25], (25, 35], (18, 25], ..., (25, 35], (60, 100], (35, 60], (35, 60], (25, 35)]
```

Right interval inclusion

```
Length: 12
```

```
Categories (4, interval[int64]): [(18, 25] < (25, 35] < (35, 60] < (60, 100]]
```

```
cats = pd.cut(ages, bins, right= False)
```

```
[(18, 25), (18, 25), (25, 35), (25, 35), (18, 25), ..., (25, 35), (60, 100), (35, 60), (35, 60), (25, 35)]
```

Changing intervals: Right exclusion

```
Length: 12
```

```
Categories (4, interval[int64]): [(18, 25) < [25, 35) < [35, 60) < [60, 100)]
```


Binning cont.

```
Categories object with attributes Codes and Categories
cats=pd.cut(x=ages,bins=bins,labels=['Youth',
'YoungAdult', 'MiddleAged', 'Senior'])
cats.categories
```

```
Index(['Youth', 'YoungAdult', 'MiddleAged', 'Senior'], dtype='object')
```

```
cats.codes
```

```
array([0, 0, 0, 1, 0, 0, 2, 1, 3, 2, 2, 1], dtype=int8)
```

```
pd.value_counts(cats)
```

Youth	5
MiddleAged	3
YoungAdult	3
Senior	1
dtype:	int64

Binning cont.

Binning with integers: computes equal-length bins based on the minimum and maximum values in the data

```
data = np.random.rand(20)
pd.cut(data, 4, precision=2)
```

```
Categories (4, interval[float64]): [(0.0049, 0.24] < (0.24, 0.48] < (0.48, 0.72] < (0.72, 0.96]]
```

```
pd.value_counts(cats)
```

```
(0.72, 0.96]      9
(0.24, 0.48]      7
(0.48, 0.72]      3
(0.0049, 0.24]    1
dtype: int64
```

1. Try It.
2. Search: what is precision?

Binning cont.

Binning with integers: using qcut for roughly equal sized bins

```
data = np.random.rand(1000)
pd.qcut(data, 4, precision = 2)
```

```
Categories (4, interval[float64]): [(-0.00969, 0.24] < (0.24, 0.5] < (0.5, 0.74] < (0.74, 1.0]]
```

```
(0.74, 1.0]      250
(0.5, 0.74]      250
(0.24, 0.5]      250
(-0.00969, 0.24] 250
dtype: int64
```

Try IT

Use the previous slides and Try binning!

Outliers

What is an outlier?

Why should we detect outliers?

What should we do with the outliers?

Outliers

Reasons:

- Data errors (data entry error, measurement error)
- high variations

Outlier types:

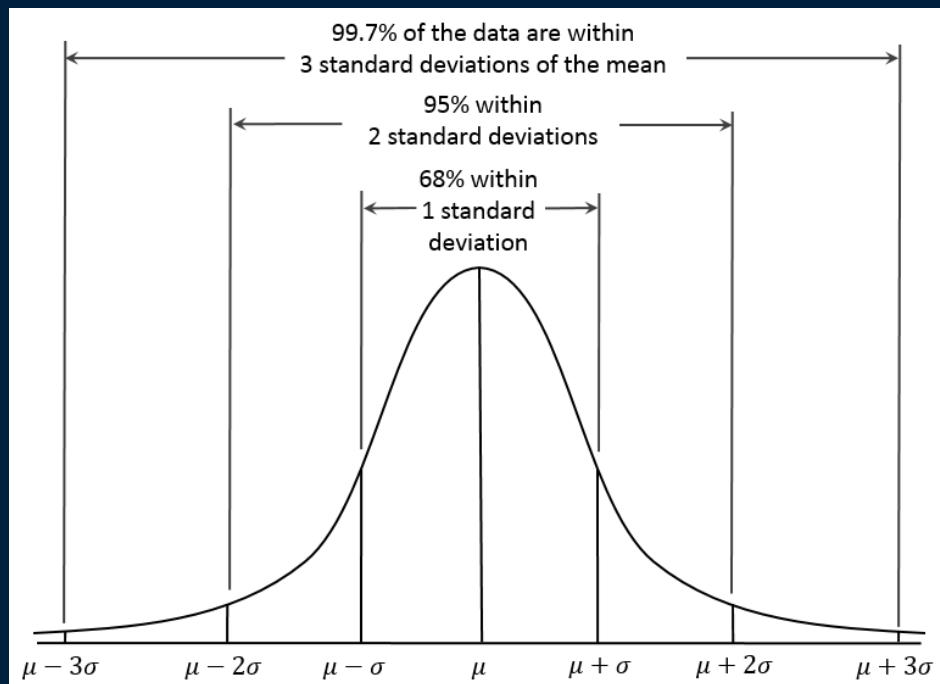
- Univariate
- Multivariate

Methods: Statistical methods and ML methods

- Z-score, Modified Z-Score, Maximum likelihood
- IQR (interquartile range)
- DBScan
- Minkowski error: Reduces the contribution of potential outliers in the training process

Univariate outlier detection with z-score

Z-score: how many standard deviation a data point is from the sample's mean



$$Z = \frac{x - \mu}{\sigma}$$

μ : mean

σ : standard deviation

Some rule of thumb threshold for outliers using z-score is: 2.5, 3, 3.5 or more.

Univariate outlier detection with modified z-score

Good for normal distribution, low dimensional features

Does not work for small number of data points (<12)

Z-score is not robust for small datasets

Modified z-score is less influenced by the outliers and works with median instead of mean in the score

modified z score: “is a standardized score that measures outlier strength or how much a particular score differs from the typical score.”

Univariate outlier detection with modified z-score cont.

mean absolute deviation (MeanAD)

median absolute deviation (MAD)

median absolute deviation = median($x - \text{median}(x)$)

Dataset: $X = X_1, X_2, \dots, X_n$

Median: $\text{median } X = \bar{X}$

MAD: $MAD = \text{median}(|X - \bar{X}|)$

If MAD does equal 0: $(X - \bar{X}) / (1.253314 * \text{MeanAD})$

If MAD does not equal 0: $(X - \bar{X}) / (1.486 * MAD)$

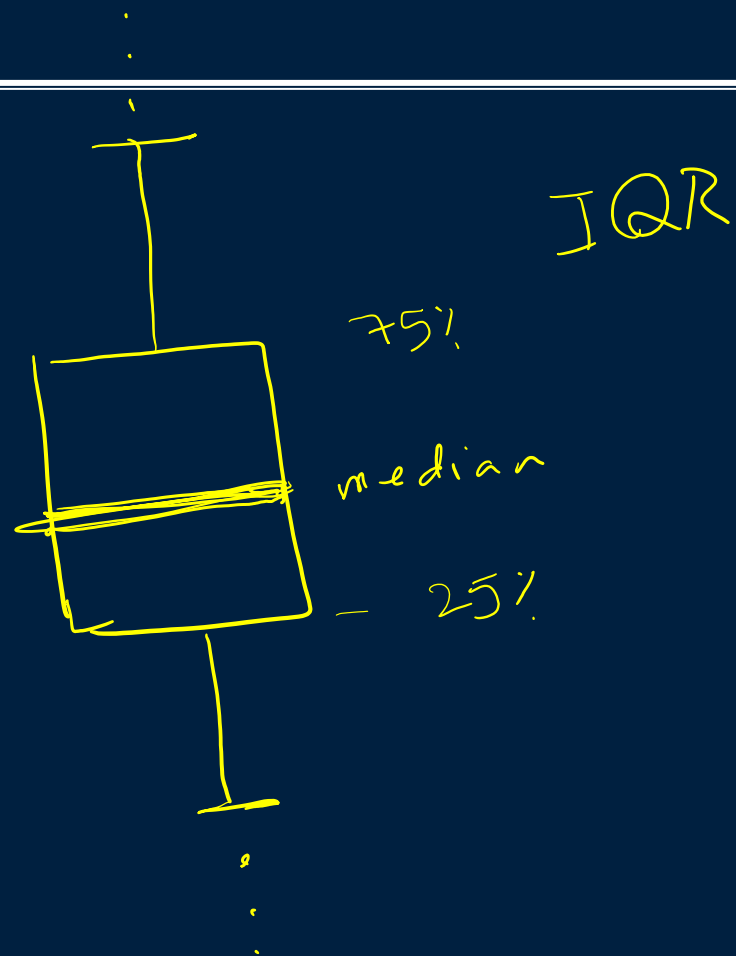
Univariate outlier detection with modified z-score cont.

```
import numpy as np
def outliers_modified_z_score(ys):
    threshold = 3.5
    median_y = np.median(ys)
    median_absolute_deviation_y = np.median([np.abs(y -
median_y) for y in ys])
    modified_z_scores = [0.6745 * (y - median_y) /
median_absolute_deviation_y for y in ys]
    return np.where(np.abs(modified_z_scores) > threshold)
```

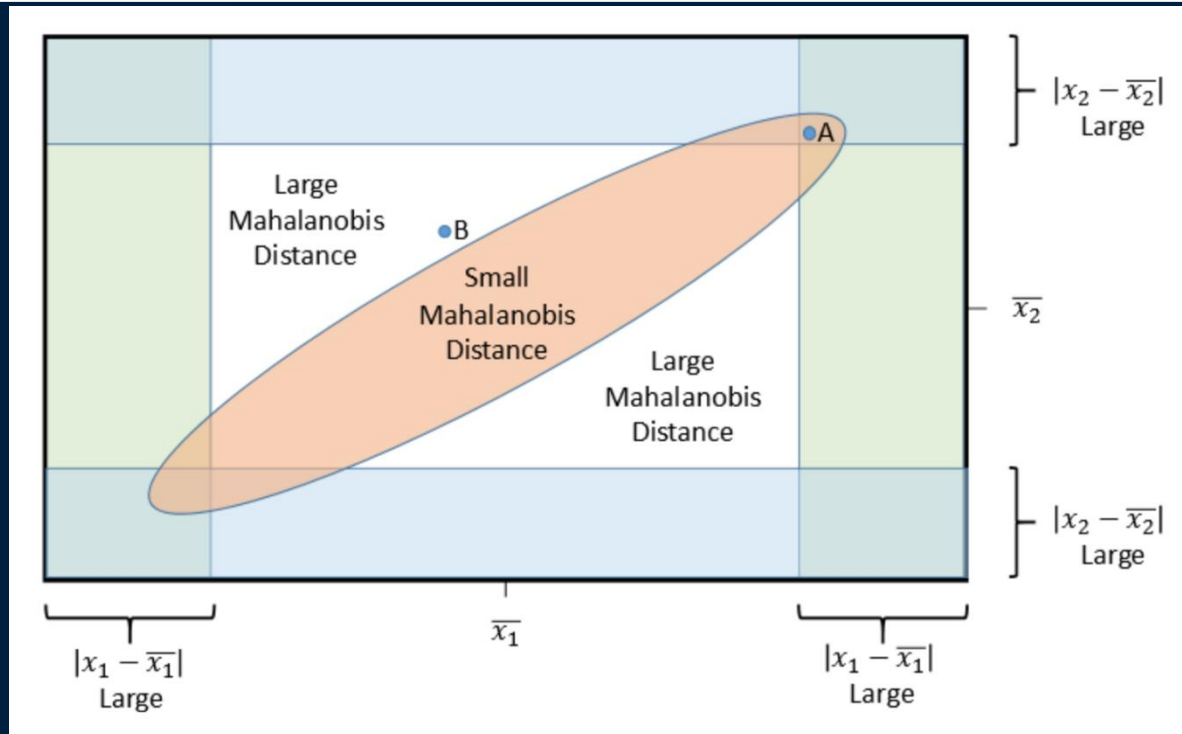
$$MAD = \text{median}\{|x_i - \tilde{x}|\}$$

$$M_i = \frac{0.6745(x_i - \tilde{x})}{MAD}$$

→ 0.6745 is the 0.75th quartile of the standard normal distribution, to which the MAD converges to.

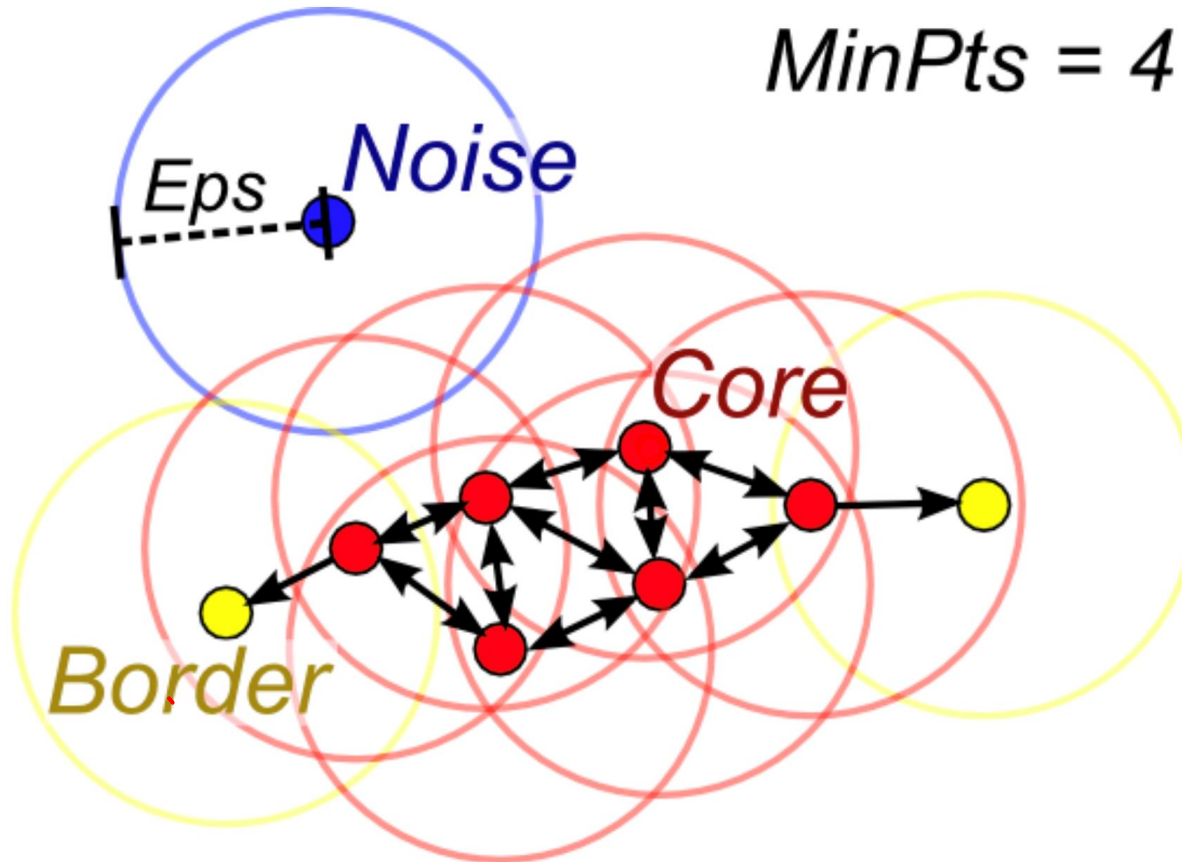


Univariate vs Multivariate Outliers



<https://blogs.sas.com/content/iml/2019/03/25/geometry-multivariate-univariate-outliers.html>

DBSCAN: Density-Based Spatial Clustering of Applications with Noise



Red: Core Points



Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but does not meet the min_points criteria

Blue: Noise point. Not assigned to a cluster

Lecture 4 learning outcomes

At the end of this lecture you should be able to:

- Perform duplicate removal in Pandas
- Perform axis renaming
- Perform Binning on datasets
- Explain some outlier removal techniques
- Perform outlier detection using z-score

More resources

Outlier detection:

1. Siffer, A., Fouque, P. A., Termier, A., & Largouet, C. (2017, August). Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1067-1075). ACM.
2. Leys, C., Delacre, M., Mora, Y. L., Lakens, D., & Ley, C. (2019). How to Classify, Detect, and Manage Univariate and Multivariate Outliers, With Emphasis on Pre-Registration. *International Review of Social Psychology*, 32(1).
3. Rousseeuw, P. J., & Hubert, M. (2011). Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1), 73-79.
4. Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.
5. The curse of dimensionality: How to define outliers in high-dimensional data?, <https://blogs.sas.com/content/iml/2012/03/23/the-curse-of-dimensionality.html>
6. Multivariate location and scatter (MCD algorithm), <https://blogs.sas.com/content/iml/2012/02/02/detecting-outliers-in-sas-part-3-multivariate-location-and-scatter.html>.
7. Mahalanobis distance, <https://blogs.sas.com/content/iml/2012/02/15/what-is-mahalanobis-distance.html>
8. <https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561>
9. <https://www.kdnuggets.com/2017/01/3-methods-deal-outliers.html>
10. <https://towardsdatascience.com/density-based-algorithm-for-outlier-detection-8f278d2f7983>

Anonymous Functions in R and Python:

1. <https://lukesingham.com/anonymous-functions-in-r-python/>

Quick catch up on Pandas DataFrames:

1. <https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python>

Try it

Discuss how to calculate the MAD. One simple example is given here:

<https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/other-measures-of-spread/a/mean-absolute-deviation-mad-review>

Try it

Follow the instructions from the following link and implement z-score, modified z-score and IQR:

<http://colingorrie.github.io/outlier-detection.html>



THE UNIVERSITY OF BRITISH COLUMBIA

