Data Treatment
Part 2

UNIVERSITY of
HOUSTON
DIVISION OF RESEARCH

Data Sorting

- *Meaning:*
  - *Change the order of the data values in a data-frame as*
    - Sorting values in one column
      - df.sort_values(*by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last'*)
    - Sorting values in multiple column
      - df.sort_values*([‘Column 1', ‘Column 2'], ascending = False)*
    - Documentation: [pandas.DataFrame.sort_values]()
      - axis: determines sorting along the row/column
      - ascending:
        - True: by default, and sort the list in ascending order
        - False: sort list in descending order
        - [True, False]: this list can be based on the column/row preference ascending/descending
  - *For example*

```
data.sort_values(['Month'], inplace = True)
data.head()
```

Your Turn!

- Try to sort the column “number_of_reviews” in descending order

- Which Airbnb has less price and top number of reviews?

Data Subsetting

- *Meaning:*
  - *To view specific group of data*
  - *To filter your data*

- Subsetting value in one column
  - *df.column_1.unique()*

- Sorting values in multiple column
  - df[*['Column 1', 'Column 2'], ascending = False]*

- *For example:*

```
subset_2 = data[['neighbourhood', 'latitude','longitude','price']]
```

Data Filtering

- ***Meaning:***
  - Filter a group of the data which we would look at for analysis

- Methods

  1. .loc method:
     - *Access a group of rows and columns by label(s) or a boolean array*
     - Documentation: [pandas.DataFrame.loc]

  2. .iloc method (i – integer)
     - Purely integer-location based indexing for selection by position
     - Documentation: [pandas.DataFrame.iloc]

  3. groupby method:
     - Documentation: [pandas.DataFrame.groupby]
     - DataFrame.groupby(***by=None, axis=0, level=None, as_index=True, sort=True, dropna=True***)

Your turn!

- Which host(host name) of which neighborhood had the last_review?

- Based on the above how to see the reviews_per_month for them.

Your turn!

- List all the Airbnb’s name ,hostname, availability_365 and year
- List the hostname and year when the availability_365 was zero

Data Melting & Reshaping

- *Meaning:*
  - Transferring data from a wide format to a long format
  - It is useful when we have a data frame where we want to create one of the columns as identifier and another column contains the measure
  - Documentation: [pandas.DataFrame.melt]()

  - pd.melt(dataFrame, id_vars = [‘Col1’ , ’Col2’], var_name='Date', value_name='GDPperCapGrowth%’)

    - *id_vars: Column which you would like to keep*
    - *var_name: Column which you create*
    - *Value_name: values*

For Example:

Country Name	Country Code	1990-12-31 00:00:00	2000-12-31 00:00:00	2011-12-31 00:00:00	2012-12-31 00:00:00	2013-12-31 00:00:00	2014-12-31 00:00:00	2015-12-31 00:00:00	2016-12-31 00:00:00	2017-12-31 00:00:00	201
Australia	AUS	3.107811e+11	4.152226e+11	1.396650e+12	1.546152e+12	1.576184e+12	1.467484e+12	1.351694e+12	1.208847e+12	1.329188e+12	1.432
Brazil	BRA	4.619518e+11	6.554206e+11	2.616201e+12	2.465189e+12	2.472807e+12	2.455994e+12	1.802214e+12	1.795700e+12	2.062831e+12	1.885
Hong Kong SAR, China	HKG	7.692829e+10	1.716682e+11	2.485136e+11	2.626294e+11	2.756969e+11	2.914594e+11	3.093836e+11	3.208076e+11	3.412443e+11	3.616
Japan	JPN	3.132818e+12	4.887520e+12	6.157460e+12	6.203213e+12	5.155717e+12	4.850414e+12	4.389476e+12	4.922538e+12	4.866864e+12	4.954
Singapore	SGP	3.614434e+10	9.607448e+10	2.793612e+11	2.950872e+11	3.075764e+11	3.148512e+11	3.080041e+11	3.186523e+11	3.418633e+11	3.732

	Country Name	Country Code	Date	GDPperCapGrowth%
0	Australia	AUS	1990-12-31	3.107811e+11
1	Brazil	BRA	1990-12-31	4.619518e+11
2	Hong Kong SAR, China	HKG	1990-12-31	7.692829e+10
3	Japan	JPN	1990-12-31	3.132818e+12
4	Singapore	SGP	1990-12-31	3.614434e+10
...	...	...	...	...
61	Brazil	BRA	2019-12-31	1.839758e+12
62	Hong Kong SAR, China	HKG	2019-12-31	3.657115e+11
63	Japan	JPN	2019-12-31	5.081770e+12
64	Singapore	SGP	2019-12-31	3.720625e+11

Data Pivoting

- *Meaning:*
  - *Let you return the reshaped data back to wide format*
  - *Let you insert the index to the data. The index can be your column*

  - *Two Ways:*
    1. df.pivot(index="lev1", columns=["lev2", "lev3"],values="values")

```
>>> df
   lev1  lev2  lev3  lev4  values
0     1     1     1     1       0
1     1     1     2     2       1
2     1     2     1     3       2
3     2     1     2     4       3
4     2     1     1     5       4
5     2     2     2     6       5
```

```
          lev3     1     2
lev1      lev2
   1         1   0.0   1.0
             2   2.0   NaN
   2         1   4.0   3.0
             2   NaN   5.0
```

Disadvantage: It does use recognize the duplicated values

Value Error: Index contains duplicate entries, cannot reshape

Data Pivoting

2)

pd.pivot_table(df, values='D', index=['A', 'B'], columns=['C'], agg=np.sum)

```
>>> df
     A    B      C  D  E
0  foo  one  small  1  2
1  foo  one  large  2  4
2  foo  one  large  2  5
3  foo  two  small  3  5
4  foo  two  small  3  6
5  bar  one  large  4  6
6  bar  one  small  5  8
7  bar  two  small  6  9
8  bar  two  large  7  9
```

```
C          large  small
A   B
bar one      4.0    5.0
    two      7.0    6.0
foo one      4.0    1.0
    two      NaN    6.0
```

Your Turn!

	id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
2	MX17004	2010	2	tmax	NaN	27.3	24.1	NaN	NaN	NaN	NaN	NaN
3	MX17004	2010	2	tmin	NaN	14.4	14.4	NaN	NaN	NaN	NaN	NaN
0	MX17004	2010	1	tmax	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	MX17004	2010	4	tmin	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	MX17004	2010	3	tmin	NaN	NaN	NaN	NaN	14.2	NaN	NaN	NaN
6	MX17004	2010	4	tmax	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	MX17004	2010	3	tmax	NaN	NaN	NaN	NaN	32.1	NaN	NaN	NaN
8	MX17004	2010	5	tmax	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	MX17004	2010	5	tmin	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	MX17004	2010	1	tmin	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

- Why should we be applying melt(reshape data) function here?
- How can you apply the melt function on this?

Data Merging

- ***Meaning:***

  Joining two data series and data frames

  - *Where to use: Two data files to extract specific query answer*

1. Concatenate Data Frames along row and column.
2. Merge Data Frames on specific keys by different join logics like left-join, inner-join, etc.

Data Concatenate

- Documentation: [pandas.concat]

pd.concat(***objs*, *axis=0*, *join='outer'*, *ignore_index=False*, *keys=None*, *levels=None*, *names=None*, *verify_integrity=False*, *sort=False*, *copy=True***

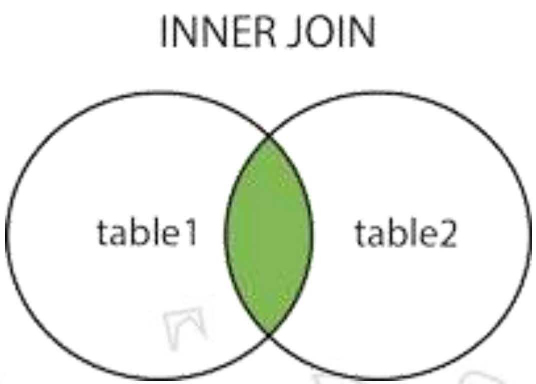
- **objs:*a sequence or mapping of Series or DataFrame objects***
- **axis: *0/index/row, 1/columns***

Data Merging

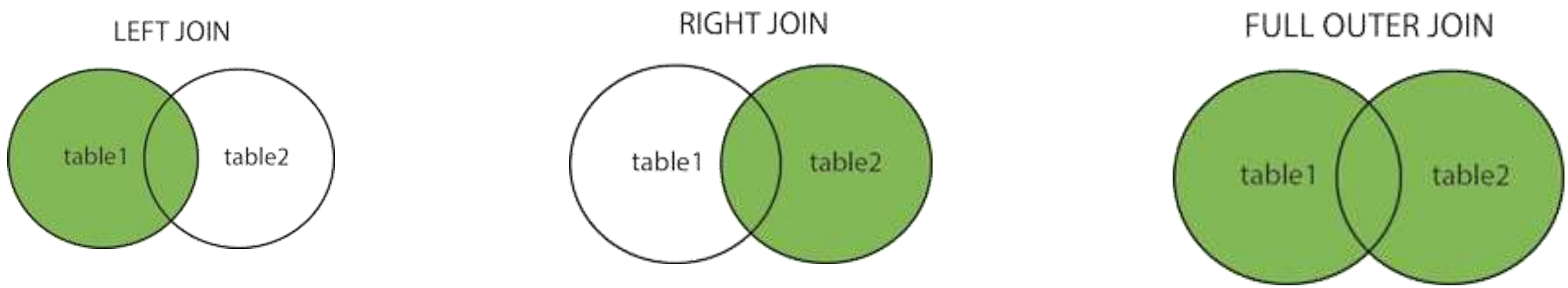
Documentation: pandas.DataFrame.merge

pd.merge(*right*, *how='inner'*, *on=None*, *left_on=None*, *right_on=None*, *left_index=False*, *right_index=False*, *sort=False*, *suffixes=('_x', '_y')*, *copy=True*, *indicator=False*, *validate=None*)

- **How :*{'left', 'right', 'outer', 'inner', 'cross'}***



Different Types of SQL JOINs



- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table

Fundamental Statistics

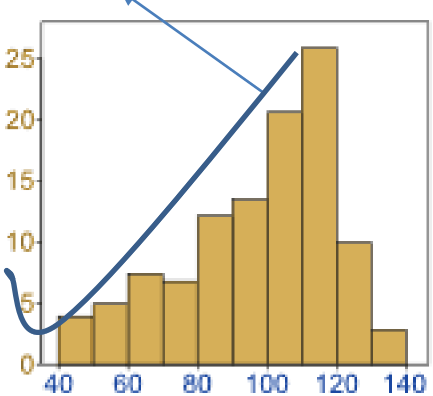
UNIVERSITY of HOUSTON
DIVISION OF RESEARCH

Skewness

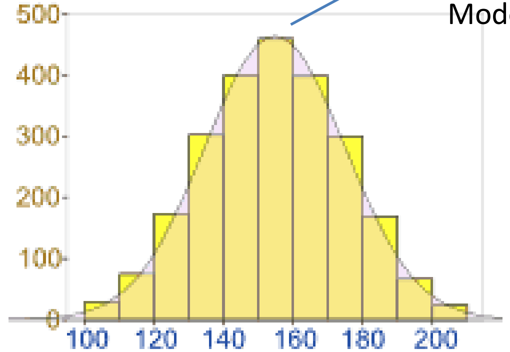
- ***Meaning:***

  Data tends to have a **long tail** on one side or the other

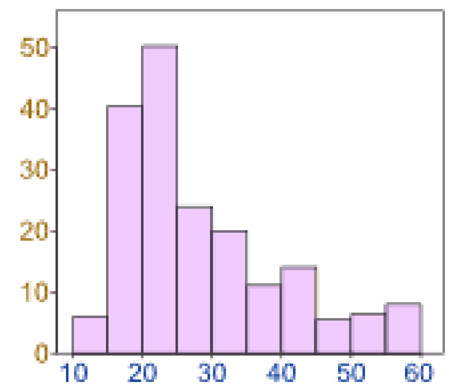
Mean



Negative skew



Normal distribution
has no skew



Positive skew

How to detect the skewness?

- ***Method 1: Using .skew() function***
  - *Documentation: [pandas.DataFrame.skew]()*
  - DataFrame.skew(***axis=None, skipna=None, level=None, numeric_only=None***)

    - **axis:** *{index (0), columns (1)} ;* Axis for the function to be applied on

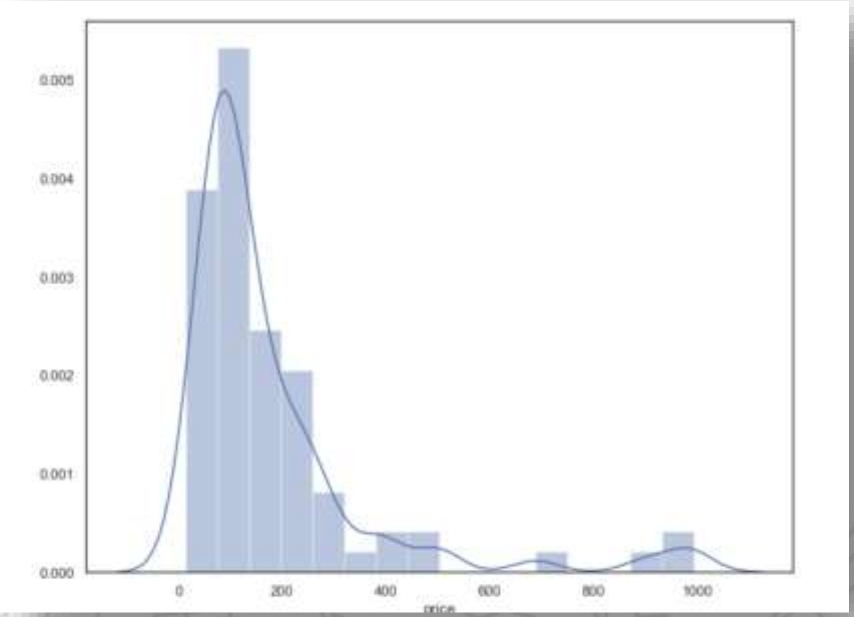
```
DataFrame:
    0   1   2   3   4   5   6
0  10  20  30  40  50  60  70
1  10  10  40  40  50  60  70
2  10  20  30  50  50  60  80
Skew:
0    0.000000
1   -0.340998
2    0.121467
dtype: float64
```

Which skewness?

How to detect the skewness?

- *Method 2: Using the distplot()*
  - *Documentation :[seaborn.distplot()]()*
  - *Comes under Seaborn library*

Flexibly plot a univariate distribution of observations



Right skewed

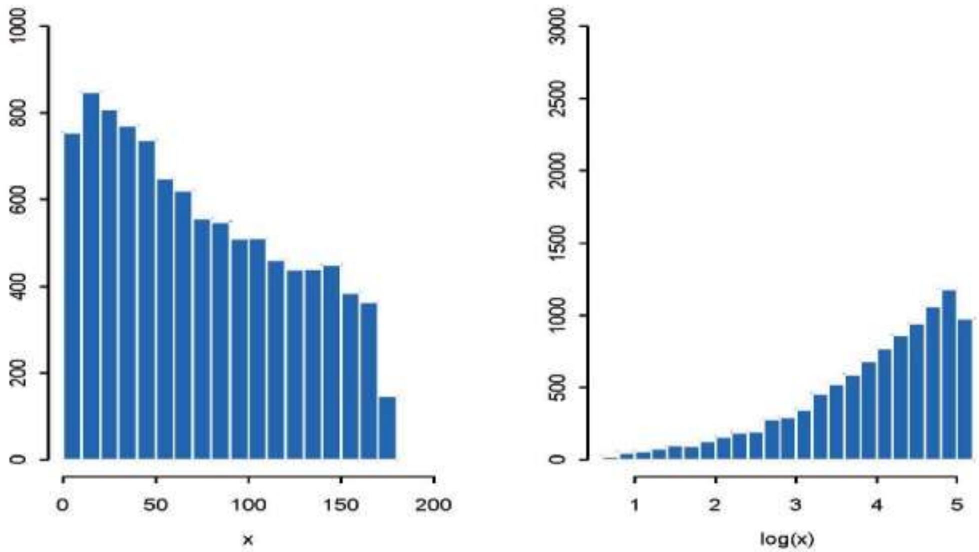
sns.distplot(data2.price.head(80))

How to remove the skewness?

- ***Method 1: Using Log transform***
  - *Documentation :numpy.log*
  - *Comes under NumPy library*
  - *np.log(x: input array)*

By default:

Log base to e



- *np.log2()*
- *np.log10()*

Histograms of original data (left plot) and log-transformed data (right plot) from a simulation study that examines the effect of log-transformation on reducing skewness.

How to remove the skewness?

When numbers are too large, one can try fractional exponents as a means of transformation

- *Method 2: Using square root or cube root transform*
  - *Documentation: numpy.sqrt()*
  - *np.sqrt(array)*
    - Return the non-negative square-root of an array, element-wise

      - df.col_name**(1/2)

Normalization

- ***Meaning:***
  - *Rescaling the values in the range of [0,1]*

- ***Why:***
  - *When your data-set has multiple features(or column) with different measurement scale*

- ***Keep in mind:***
  - *Magnitude*
  - *Units*

For example:

Employee Name	Salary	Work Ex (Years)	Band Level
John	1,10,000	13	3
Anna	67,000	8	4
Scarlette	23,000	3	7
Shiva	75,000	10	4
Sean	43,000	5	5

Notice
- Salary
- Work EX

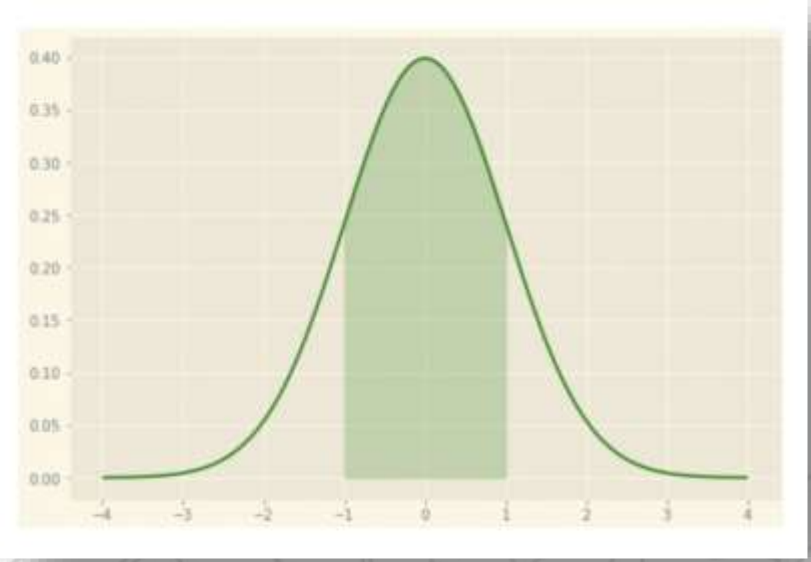
"Not every and not always feature/columns in your dataset requires normalization"

Note: Check if your data Normally/Gaussian distributed

What is Normal distribution/Gaussian distribution?
It is symmetric about the mean -> data around the mean is more frequent

- Mean is ”Zero”
- Standard deviation is “One”
- Normal distributions are symmetrical,

but not all symmetrical distributions are normal



Bell Shaped curve

Source: www.investopedia.com

How to perform Normalization?

- Simple Feature Scaling
- Min-Max Feature Scaling
- Z-Score/ **Standard scores**

**Considering the data does not follow Gaussian distribution

Simple Feature Scaling

DataFrame.loc[:,'columns/feature']

/

DataFrame.loc[:, 'columns/feature '].max()

Min-Max Feature Scaling

$$X' = (X - X_{min}) / (X_{max} - X_{min})$$

Feature scaling is used to bring all values into the range [0,1]. This is also called unity-based normalization. This can be generalized to restrict the range of values in the dataset between any arbitrary points a and b, using for example $X' = a + \frac{(X - X_{\min})(b - a)}{X_{\max} - X_{\min}}$.

Source: Normalization_(statistics)

(DataFrame.loc[:, 'Feature/column']-
DataFrame.loc[:, 'Feature/column '].min())
/
(DataFrame.loc[:, 'Feature/column '].max() -
DataFrame.loc[:, 'Feature/column '].min())

Example:
calculated_host_listings_count

1728	0.000000
4840	0.008065
2561	0.000000
8258	0.000000
3799	0.016129
	...
4369	0.209677
1606	0.016129
4020	0.008065
3107	0.016129
5413	0.008065

Z- Score/ Standard Scores

**Your data follows Normal distribution

Z-score = X - u(mu) / Sigma

Where: mu –> mean
sigma -> standard deviation(SD)

Z-Score tells how many standard deviations away from the mean is your score

For example:

- if your Z-score is 1.2 -> 1.2 SD above the mean
- if your Z-score is -0.6 -> 0.6 SD below the mean

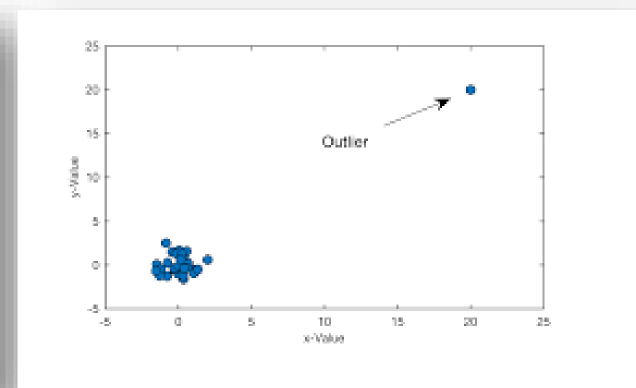
Outlier

*A z-score of **zero** tells us the value is **exactly the mean/ average** while a score of +6 tells you that the value is much higher than average (probably an outlier)*

- *Meaning: These are the points which are way to far from the regular pattern*

Outliers are two types:

- Univariate
- Multi-variate



Univariate Outlier

Univariate: These outliers are the points consists of an extreme value on one variable

How to detect these kind of outliers?

- **IQR and Box-and-Whisker's plot**

INTER-QURATILE RANGE(IQR)

IQR – THIRD QUARTILE – FIRST QUARTILE

75TH PERCENTILE – 25TH PERCENTILE

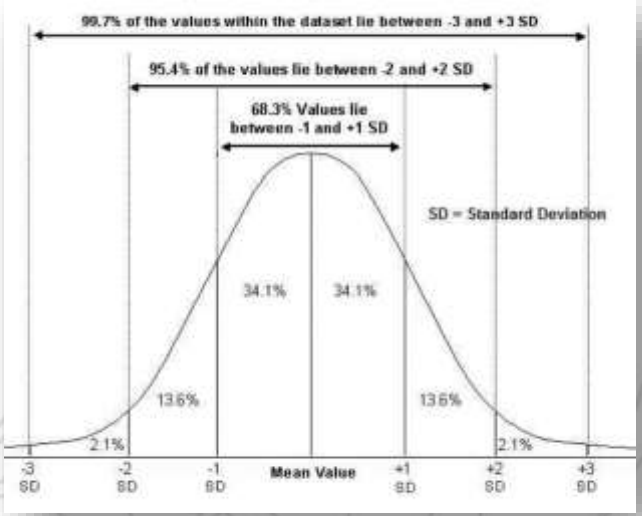
LOWER BOUND = FIRST QUARTILE – 1.5times(IQR)

UPPER BOUND = THIRD QUARTILE +1.5times(IQR)

Any values outside these values ranges:

- below lower bound
- above upper bound

OUTLIERS



Box-and-Whisker plot

A robust method for detecting outliers is the

- IQR (Inter Quartile Range) method
- It was developed by John Tukey, pioneer of exploratory data analysis
- Box-and-Whisker's plot uses quartiles to plot the shape of a variable

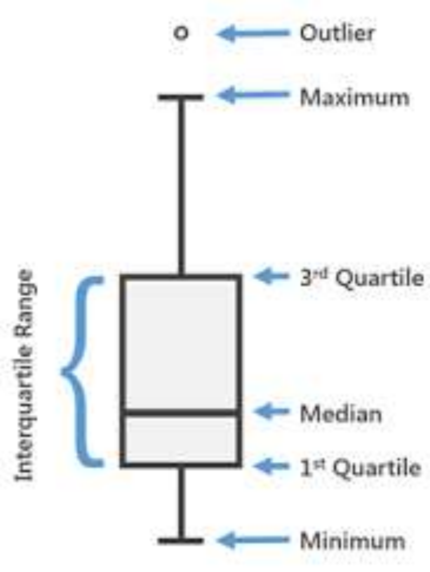
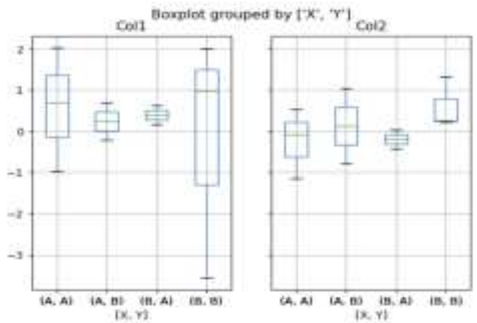


Image Source

How to create the boxplot?

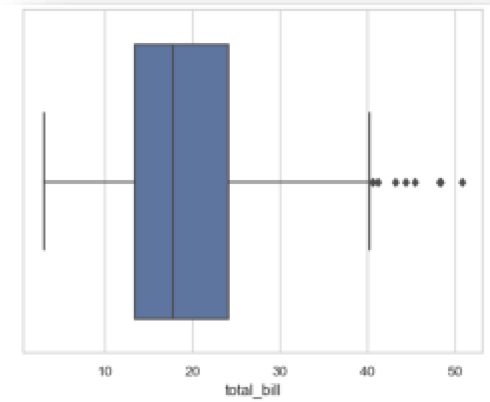
1. Using Pandas library
   - Documentation: [pandas.DataFrame.boxplot]()

```
>>> df = pd.DataFrame(np.random.randn(10, 3),
...                   columns=['Col1', 'Col2', 'Col3'])
>>> df['X'] = pd.Series(['A', 'A', 'A', 'A', 'A',
...                      'B', 'B', 'B', 'B', 'B'])
>>> df['Y'] = pd.Series(['A', 'B', 'A', 'B', 'A',
...                      'B', 'A', 'B', 'A', 'B'])
>>> boxplot = df.boxplot(column=['Col1', 'Col2'], by=['X', 'Y'])
```



by: *str or array-like, optional* Column in the DataFrame to pandas.DataFrame.groupby()

- ax = sns.boxplot(x=tips["total_bill"])



2. Using Seaborn library
   - import seaborn as sns
   - Documentation: seaborn.boxplot

- sns.boxplot(x='room_type', y='price', data=data2)