wbs
WARWICK BUSINESS SCHOOL
THE UNIVERSITY OF WARWICK

For the Change Makers

Programming for Data Analytics

Week 6 : Data Processing
Information Systems and Management
Warwick Business School

Row Selection

- Select rows by passing **integer position** (index) to .iloc[].

```
>>> df_tips.iloc[1]
```
#implicit numeric index starting from 0.

- Note: `df8.loc[1]` differs from `df8.iloc[1]`

```
>>> df_tips.iloc[1:3]
```

What is the result of df_tips.loc[1:3]?

Add rows

- Add new rows with `append()` method at the end of a dataframe.

```
>>> df_tips.append([99,99]) #existing dataframe not updated.
>>> df_tips.append(pd.DataFrame([99,99]))
```

- Python will align matching columns.

```
>>> df_tips.append(pd.DataFrame([99,99],columns=['A','B']))
>>> df_tips.append(pd.DataFrame([[99,99]],columns=['A','B']))
```

- You can reset row labels by setting ignore_index=True (default is False).
- `df_tips`.append(pd.DataFrame([[99,99]],columns=['A','B']), ignore_index=True)

Row deletion

- You can remove rows by drop() method with a list of row labels.

```
df_tips.drop([2])
```
#same as df_tips.drop(2)

```
df_tips.drop([0:2])
```
#error [0:2] is not a list. Alternatively, df_tips = df_tips[2:].

```
df_tips.drop([0,1,2])
```

```
df_tips.drop(range(3))
```
#existing dataframe not updated

```
df_tips = df_tips.drop([2])
```

```
df_tips.drop([4],inplace=True)
```
existing dataframe updated.

Important dataframe attributes

- .index returns a "list" of row **indexes (labels)**.
- `df_tips.index`
- `df_tips.loc[df_tips['tip'] > 2].index`
- Very handy to remove rows based on condition.
- `df_tips.drop(df_tips.loc[df_tips['tips'] > 2].index)`

Data processing

Things to do

- Exploration.
  - Descriptive statistics.
  - Visualization
- Cleaning.
  - Missing data.
  - Inconsistent values.
  - Duplications.
- Wrangling.
  - Feature engineering.
    - Aggregation, splitting, extraction, generation(calculation, log, etc.).
  - Scale/Standardization.
  - Encoding categorical values.

Explore Dataset

- Dimensions of the dataset.

`df_tips.shape` # quick view of the dimensions.

`df_tips.size` # total number of measurements in your data.

`df_tips.info()` # count and datatype for each column

- Preview your data with head(n) and tail(n), n is 5 by default.

`df8.head(3)` # first 3 rows; `tail(3)` for last 3 rows.

Descriptive statistics

- The **`describe()`** method computes a summary (NaN will be excluded) of statistics pertaining to the DataFrame columns. Summary is also a DataFrame.

```
df_titan.describe()
```

- You may pass proper value to argument include:
  - **number** – default, only summarizes Numeric columns.
  - **object** – only summarizes String columns.
  - **all** – Summarizes all columns together (Should not pass it as a list value).

```
df_titan.describe(include='all')
```

Descriptive statistics

- You can get descriptive measurements using methods like: sum(),min(),max(),mean(),count(),median(),std(), and more. All measurements are returned as Series.

```
df_titan.mean()
```

- By default, these statistics are for each column. You can get row-based statistics by specifying axis=1.

```
df_titan.mean(axis=1)# default axis = 0
```

- To get the unique values and the corresponding counts for a column:

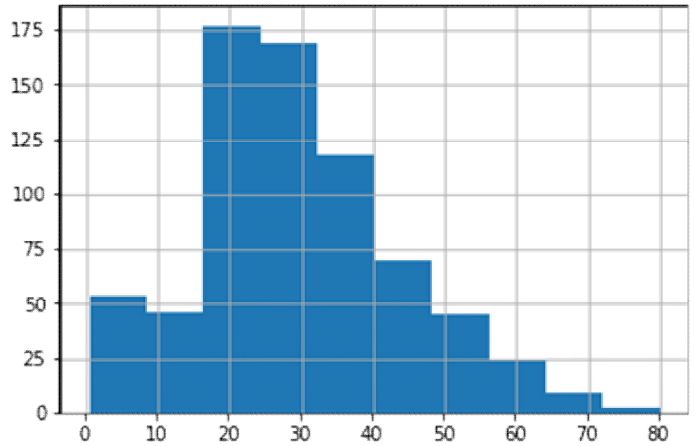
```
df_titan['Pclass'].value_counts() #only works for individual column.
```

Exploring your data with simple visualization

- Pandas offer some functions for basic plotting.
- Histogram

```
df_titan.hist()
```

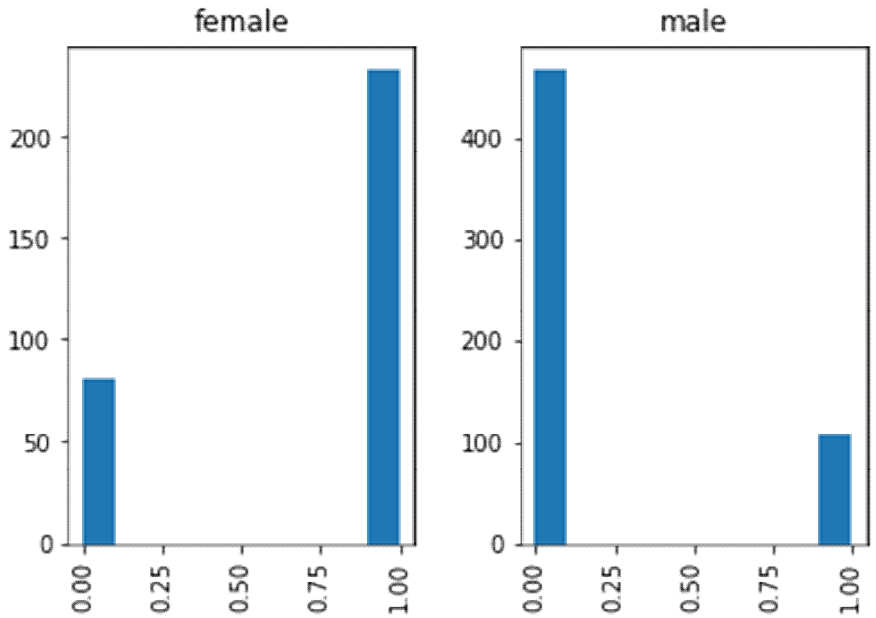
```
df_titan.hist('Age')
```



- Line
- Pie
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.html

- Instead of showing a single distribution, you can split based on another column.

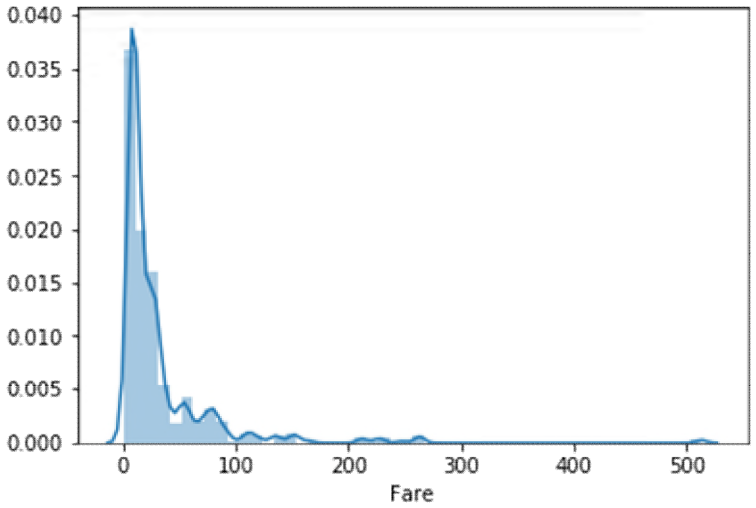
```
df_titan.hist("Survived", by="Sex")
```

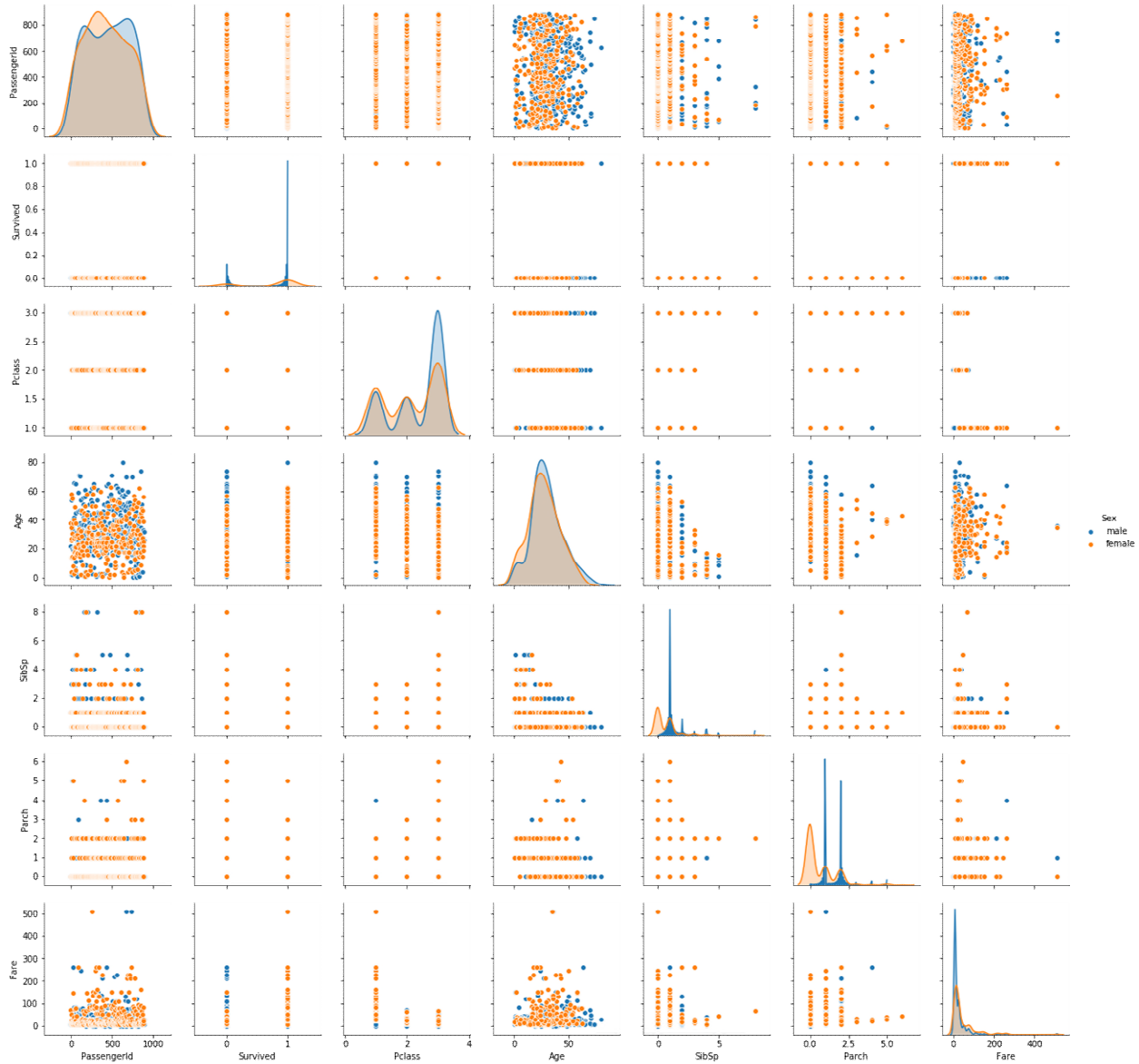


Visualization with Seaborn

- Pandas dataframe can be used directly by Seaborn as dataset, if there is no NaN value.

```
import seaborn as sns
sns.distplot(df_titan['Age'])
sns.distplot(df_titan['Fare'])
```




PassengerId
Survived
Pclass
Age
SibSp
Parch
Fare
Sex
male
female

Detect missing data

In most cases, you will have missing data issue in your dataset.

Check if there is any missing data

- `df_titan.describe(include='all')` #missing data in Age and Cabin.
- `df_titan.isnull().sum()` # another trick to find out missing data.

Reasons for missing data

1. Missing Completely at Random(MCAR): The missingness has nothing to do with any other factors.
   - ❖ Age is missing due to neglect.
2. Missing at Random(MCR): The missingness is caused by other items been measured but has nothing to do with the the its own measurement.
   - ❖ Age is missing because female don't like to report their ages.
3. Missing Not at Random (MNAR): the missingness is caused by its own measurement.
   - ❖ Age is missing because elder people don't like to report their ages.

Strategies to deal with missing data

- Delete data with missing value, with **CAUTION**.
- Removing rows with missing data by using dropna().

```
df_titan.dropna()
```
old dataframe will not be replaced automatically.

- Removing columns with missing data by specifying axis=1

```
df_titan.dropna(axis=1)
```
age and cabin dropped.

- Set threshold to remove.

```
df_titan.dropna(thresh=11)
```
keep rows with at least 11 non-missing data. i.e. rows missing both age and cabin get dropped.

Strategies to deal with missing data

- Impute missing data. Very tricky and a lot of consideration. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3668100/

1. Impute with a constant with `fillna()`.

```
df_titan.fillna(1) #fill all NaN with 1.
df_titan.fillna({'Age':20,'Cabin':'B96'}) # different value for two columns.
```

2. Impute with mean/median/mode.

```
df_titan.fillna({'Age':df_tips['Age'].mean(),'Cabin':'B96'}) #replace with mean.
```

3. Impute with statistic estimation.

Removing Unnecessary Data

- Remove irrelevant columns (variables) using .drop(columns = [column names]).

df_titan.drop(columns = [Ticket']) # remove Ticket column from data.

Outliers

- Sometimes, you may also want to remove outliers from you data.
  - For example, remove values outside 2 std of mean for normally distributed variables.

- top = df_titan['Age'].mean()+2*df_titan['Age'].std()
- bot = df_titan['Age'].mean()-2*df_titan['Age'].std()
- df_titan[df_titan['Age']>top].info()
- df_titan.drop[df_titan[df_titan['Age']>top].index]