NIAPythonDay3

January 28, 2021

NIA Python Bootcamp UNIT 3 - Wednesday July 19, 2017

1 UNIT 1 review

1. Python ecosystem of tools
2. Jupyter Notebook is code, output and documentation all in one document
3. Type code into cells, and to run them you press Shift-Enter
4. Different data types for different data
5. Tab completion reduces typing, shows you pop-up menu of all the things you can do with that piece of data
6. Operators take one or more input values and turn them into other values *based on the input values type*
7. Converting data from one type to another using the function syntax, e.g., int()

2 UNIT 2 Review

1. Exploring data types using the TAB key
2. Python syntax for taking slices of iterables
3. NumPy arrays: basic math operations in 1-D and 2-D (e.g., row-wise and column-wise eman)
4. Subselecting based on a boolean criterion
5. Example: Images as 3-D matrices

3 UNIT 3:

3. PANDAS DataFrames
4. Simple and complex sorting

3.1 PANDAS DataFrame

- pandas = Python Data Analysis Library
- Emulate R's data.frame structure.
- Basically a NumPy matrix with
  - Row and column names
  - Can have columns of different types
  - Handles missing data better

3.2 Load the PANDAS package into memory using import()

```
[1]: import pandas as pd
```

3.3 Use PANDAS read_* functions to import data

- There are many functions to import data
- Type pd.read_ then TAB to see all the import functions

```
[ ]: pd.read_
```

3.4 Read data from file or URL

```
[2]: titanic_data_url  = "https://gist.githubusercontent.com/michhar/
      ↪2dfd2de0d4f8727f873422c5d959fff5/raw/
      ↪fa71405126017e6a37bea592440b4bee94bf7b9e/titanic.csv"
```

```
[3]: # Optional - you can use R and Python at the same time
     # if you have the Python package rpy2 installed
     %load_ext rpy2.ipython
```

```
[4]: titanic = pd.read_csv( titanic_data_url )
```

```
[5]: titanic['Cabin'] = titanic['Cabin'].fillna( "" )
```

```
[6]: titanic.head()
```

```
[6]:    PassengerId  Survived  Pclass  \
     0            1         0       3
     1            2         1       1
     2            3         1       3
     3            4         1       1
     4            5         0       3

                                                    Name     Sex   Age  SibSp  \
     0                            Braund, Mr. Owen Harris    male  22.0      1
     1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
     2                             Heikkinen, Miss. Laina  female  26.0      0
     3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
     4                           Allen, Mr. William Henry    male  35.0      0

        Parch            Ticket     Fare Cabin Embarked
     0      0         A/5 21171   7.2500              S
     1      0          PC 17599  71.2833   C85        C
     2      0  STON/O2. 3101282   7.9250              S
     3      0            113803  53.1000  C123        S
     4      0            373450   8.0500              S
```

[7]:

```
%%R -i titanic
library(tidyverse)
```

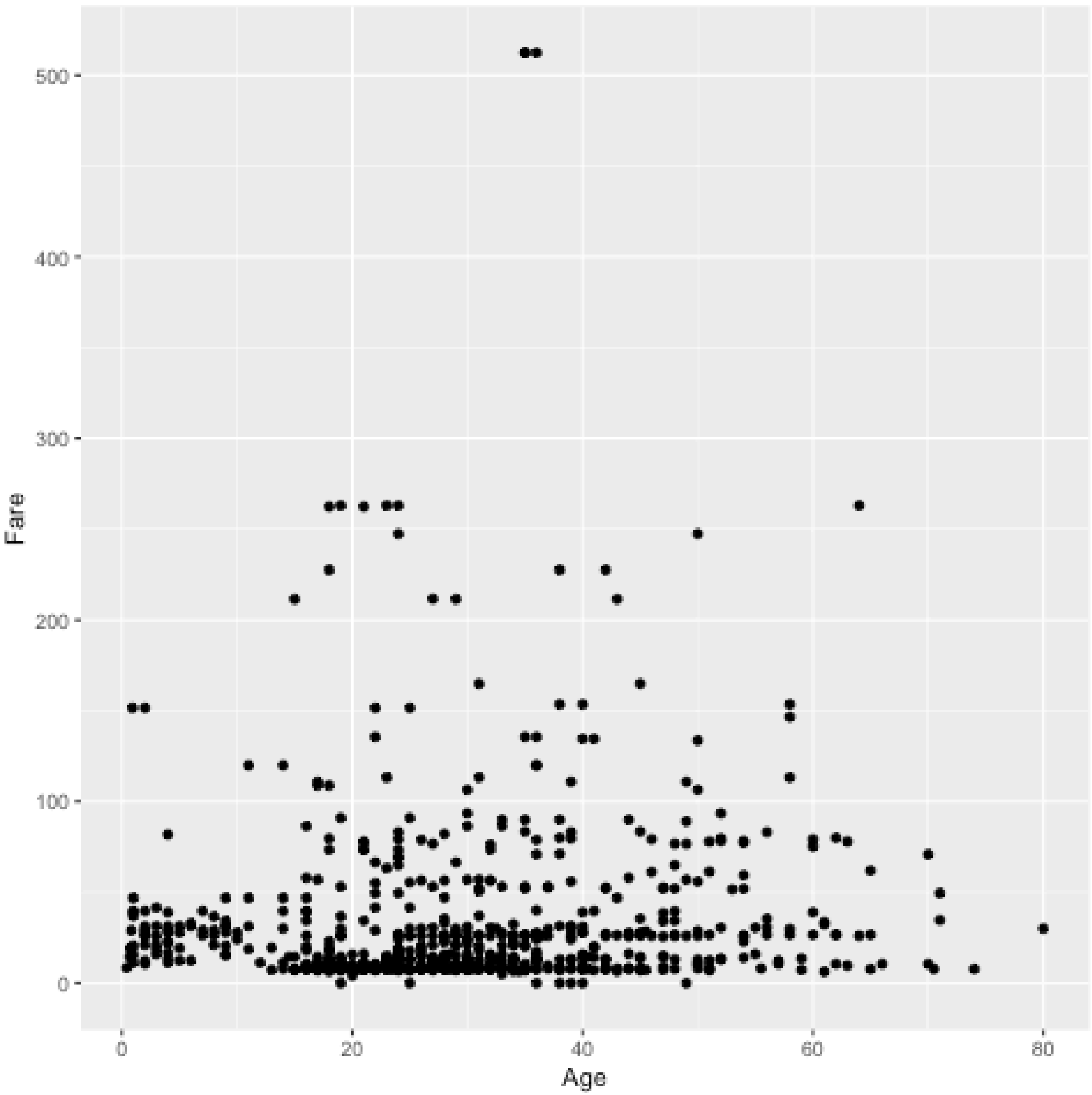
```
R[write to console]:   Attaching packages
                      tidyverse 1.3.0

R[write to console]:  ggplot2 3.3.2
purrr   0.3.4
 tibble  3.0.3      dplyr   1.0.1
 tidyr   1.1.1      stringr 1.4.0
 readr   1.3.1      forcats 0.5.0

R[write to console]:   Conflicts
                       tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()
```

[8]:

```
%%R
#glimpse( titanic)
ggplot( titanic, aes( Age, Fare ) ) + geom_point( )
```



3.5 Return type is a DataFrame

```
[9]: type(titanic)
```

```
[9]: pandas.core.frame.DataFrame
```

3.6 What did we just load?

```
[10]: titanic.shape
```

```
[10]: (891, 12)
```

3.6.1 Change the number of rows Pandas will display using the set_option() function

Use the word None if you want to display all of them.

[11]:
```
#pd.set_option( 'display.max_rows', 50 )
```

3.6.2 See the first N rows using .head(N)

Defaults to first 5

[12]:
```
titanic.head(2)
```

[12]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1

   Parch     Ticket     Fare Cabin Embarked
0      0  A/5 21171   7.2500              S
1      0   PC 17599  71.2833   C85        C
```

3.6.3 See the last N rows using .tail(N)

Defaults to last 5.

[13]:
```
titanic.tail(1)
```

[13]:
```
     PassengerId  Survived  Pclass                 Name   Sex   Age  SibSp  \
890          891         0       3  Dooley, Mr. Patrick  male  32.0      0

     Parch  Ticket  Fare Cabin Embarked
890      0  370376  7.75              Q
```

3.6.4 See random N rows using .sample(N)

[14]:
```
titanic.sample(3)
```

[14]:
```
     PassengerId  Survived  Pclass                          Name     Sex  \
825          826         0       3               Flynn, Mr. John    male
702          703         0       3         Barbara, Miss. Saiide  female
136          137         1       1  Newsom, Miss. Helen Monypeny  female

      Age  SibSp  Parch  Ticket     Fare Cabin Embarked
825   NaN      0      0  368323   6.9500              Q
702  18.0      0      1    2691  14.4542              C
```

```
136  19.0      0      2   11752  26.2833   D47         S
```

3.7 len() return number of observations (rows)

```
[15]: len(titanic)
```

```
[15]: 891
```

3.8 .shape attribute gives the shape

```
[16]: titanic.shape
```

```
[16]: (891, 12)
```

3.9 .describe(): Get basic statistics across all columns

- Detects which columns are quantitative gives descriptive stats for those

```
[17]: titanic.describe()
```

```
[17]:        PassengerId    Survived      Pclass         Age       SibSp  \
      count   891.000000  891.000000  891.000000  714.000000  891.000000
      mean    446.000000    0.383838    2.308642   29.699118    0.523008
      std     257.353842    0.486592    0.836071   14.526497    1.102743
      min       1.000000    0.000000    1.000000    0.420000    0.000000
      25%     223.500000    0.000000    2.000000   20.125000    0.000000
      50%     446.000000    0.000000    3.000000   28.000000    0.000000
      75%     668.500000    1.000000    3.000000   38.000000    1.000000
      max     891.000000    1.000000    3.000000   80.000000    8.000000

                  Parch        Fare
      count  891.000000  891.000000
      mean     0.381594   32.204208
      std      0.806057   49.693429
      min      0.000000    0.000000
      25%      0.000000    7.910400
      50%      0.000000   14.454200
      75%      0.000000   31.000000
      max      6.000000  512.329200
```

3.10 .count() give number of non-empty cells

```
[18]: titanic.count()
```

```
[18]: PassengerId    891
      Survived       891
```

```
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          891
Embarked       889
dtype: int64
```

3.11 DataFrame row and column headers

- Like a NumPy array, but with column and row headers.
- Enables slicing by headers, and not just indices like with NumPy arrays
- The collection of row headers is stored in the .index attribute.
- The collection of column headers is stored in the .columns attribute.

```
[19]: titanic.columns
```

```
[19]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
             'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
            dtype='object')
```

```
[20]: titanic.index
```

```
[20]: RangeIndex(start=0, stop=891, step=1)
```

3.12 Get a single column

Two ways to do it:

1. Use the "object-oriented" style of API, i.e., the "dot."
2. Use the dict style, i.e., key-value style (put the column name into brackets, get the column)
3. The returned data type is a PANDAS Series object, which keeps the index from the DataFrame attached

```
[21]: titanic.columns
```

```
[21]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
             'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
            dtype='object')
```

```
[22]: titanic['Name']
```

```
[22]: 0                                  Braund, Mr. Owen Harris
      1       Cumings, Mrs. John Bradley (Florence Briggs Th…
2                                 Heikkinen, Miss. Laina
3           Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                               Allen, Mr. William Henry
                             …
886                                Montvila, Rev. Juozas
887                         Graham, Miss. Margaret Edith
888             Johnston, Miss. Catherine Helen "Carrie"
889                                Behr, Mr. Karl Howell
890                                  Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

3.13 using .values

[24]:
```
titanic['Name'].values[:10]
```

[24]:
```
array(['Braund, Mr. Owen Harris',
       'Cumings, Mrs. John Bradley (Florence Briggs Thayer)',
       'Heikkinen, Miss. Laina',
       'Futrelle, Mrs. Jacques Heath (Lily May Peel)',
       'Allen, Mr. William Henry', 'Moran, Mr. James',
       'McCarthy, Mr. Timothy J', 'Palsson, Master. Gosta Leonard',
       'Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)',
       'Nasser, Mrs. Nicholas (Adele Achem)'], dtype=object)
```

3.14 .value_counts()

[25]:
```
titanic['Sex']
```

[25]:
```
0        male
1      female
2      female
3      female
4        male
        …
886      male
887    female
888    female
889      male
890      male
Name: Sex, Length: 891, dtype: object
```

[26]:
```
titanic['Sex'].value_counts()
```

[26]:
```
male      577
female    314
Name: Sex, dtype: int64
```

3.15 Use .pivot_table() to have a breakdown of the data

3.15.1 For categorical data, use aggfunc='count'

```
[ ]: titanic.pivot_table?
```

```
[27]: titanic.count()
```

```
[27]: PassengerId    891
      Survived       891
      Pclass         891
      Name           891
      Sex            891
      Age            714
      SibSp          891
      Parch          891
      Ticket         891
      Fare           891
      Cabin          891
      Embarked       889
      dtype: int64
```

```
[28]: titanic.pivot_table( values='Survived', index='Pclass',
                          columns='Sex', aggfunc='count',
                          margins=True)
```

```
[28]: Sex     female  male  All
      Pclass
      1           94   122  216
      2           76   108  184
      3          144   347  491
      All        314   577  891
```

3.15.2 For non-categorical data, can use another statistical measure for aggregation, like mean

```
[29]: titanic.pivot_table( values='Age', index='Sex',
                          columns='Pclass',
                          aggfunc='mean', margins=True)
```

```
[29]: Pclass          1          2          3        All
      Sex
      female  34.611765  28.722973  21.750000  27.915709
      male    41.281386  30.740707  26.507589  30.726645
      All     38.233441  29.877630  25.140620  29.699118
```

3.16 Quick figures

- Execute this Jupyter command `%matplotlib inline` before executing code that makes figures to get Jupyter to render them as output.

```
[ ]: %matplotlib inline
```

3.16.1 Univarate histograms

```
[ ]: titanic['Age'].hist?
```

```
[30]: thing = titanic['Age']
```

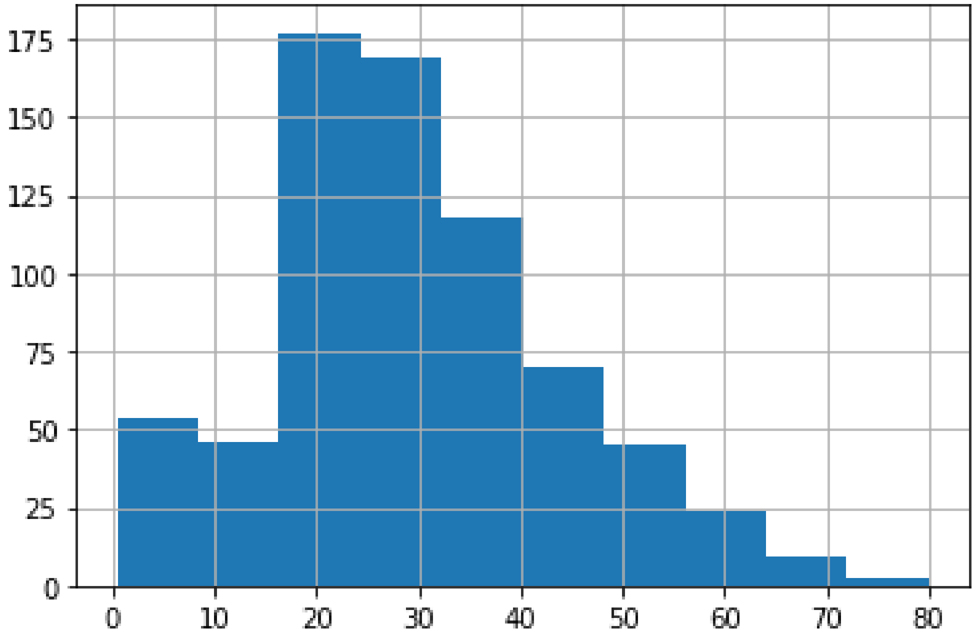
```
[31]: type( thing)
```

```
[31]: pandas.core.series.Series
```

```
[ ]: thing.hist?
```

```
[32]: titanic['Age'].hist()
```

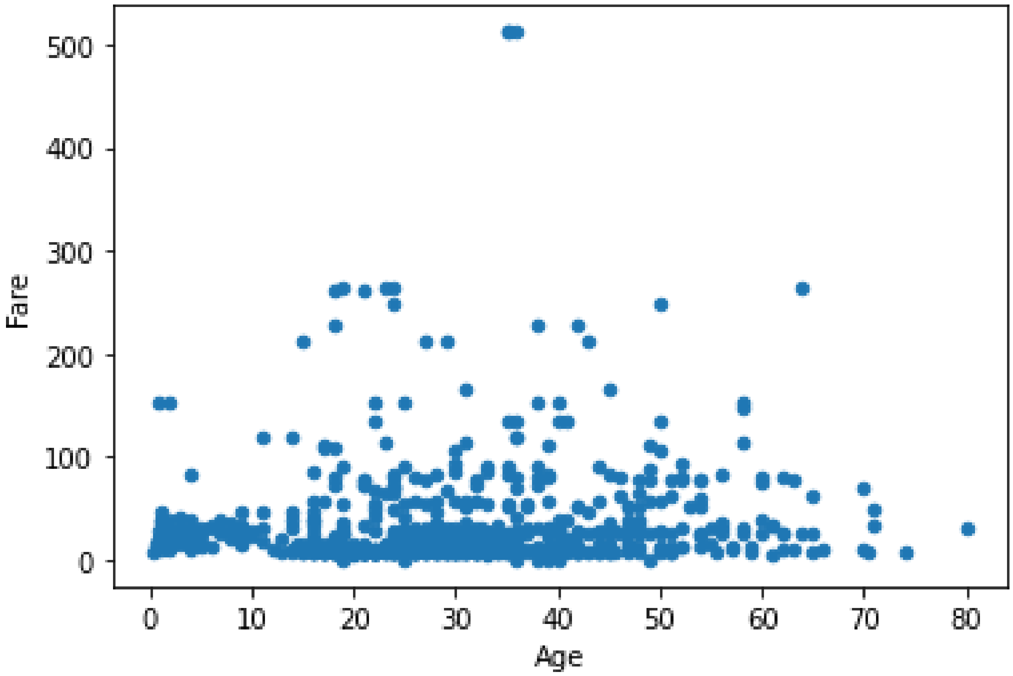
```
[32]: <AxesSubplot:>
```



3.16.2 Bivariate scatter plot using the .plot attribute

[33]:
```
titanic.plot.scatter( 'Age', 'Fare' )
```

[33]:
```
<AxesSubplot:xlabel='Age', ylabel='Fare'>
```



3.17 Missing data in PANDAS

- Represented as np.nan, which stands for “Not A Number”
- NaN has type float
- No missing data representation for an integer!
  - Either convert all to floats to use NaN (recommended!), or
  - Convert values into strings and store empties as ”” (less recommended)
  - Establish a “flag” value, e.g., -999 and filter out those before using (not recommended!)

[34]:
```
import numpy as np
```

[35]:
```
np.nan
```

[35]:
```
nan
```

[36]:
```
type( np.nan )
```

[36]:
```
float
```

3.18 Column data types

- A single column of data within a PANDAS DataFrame is called a Series.
- All values within a Series must be of the same type.
- Use the .dtypes attribute to check data types for each column

[37]:
```
titanic.head(3)
```

[37]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500                S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250                S
```

[38]:
```
titanic.count()
```

[38]:
```
PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          891
Embarked       889
dtype: int64
```

[39]:
```
titanic.dtypes
```

[39]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
```

```
Parch             int64
Ticket           object
Fare            float64
Cabin            object
Embarked         object
dtype: object
```

3.19 Column data types may hint at missing values

When using pd.read_csv() and pd.read_excel() to load a file form disk, PANDAS will try to pick a data type for a column that makes sense.

- If a float64 (just a fancy float), then missing values in the form of NaN are possible
  - Use .count() to count non-empty (non-NaN) values
- If an int64 (just a fancy int), then probably no missing values in that column
- If an object, this almost always means it's a string in there
  - Can represent missing values as ””, but .count() only works for float data types!

[40]:
```
some_emptys = pd.Series( ["","asdf","","","","27",""] )
print( some_emptys.dtype )
some_emptys.count()
```

```
object
```

[40]:
```
7
```

3.19.1 Coerce to numeric values using pd.to_numeric()

[41]:
```
some_emptys = pd.to_numeric( some_emptys, errors='coerce')
```

[42]:
```
some_emptys
```

[42]:
```
0     NaN
1     NaN
2     NaN
3     NaN
4     NaN
5    27.0
6     NaN
dtype: float64
```

[43]:
```
print( some_emptys.dtype )
some_emptys.count()
```

```
float64
```

[43]:
```
1
```

3.20 Statistics on a DataFrame ignore NaNs (as one might expect)

- In other words, doesn't count missing values as 0

[44]:
```
titanic.count()
```

[44]:
```
PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          891
Embarked       889
dtype: int64
```

[45]:
```
titanic['Age'].describe()
```

[45]:
```
count    714.000000
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

3.21 Using the Seaborn Package for visualization

- Browse this page to see all the types of nice figures you can make

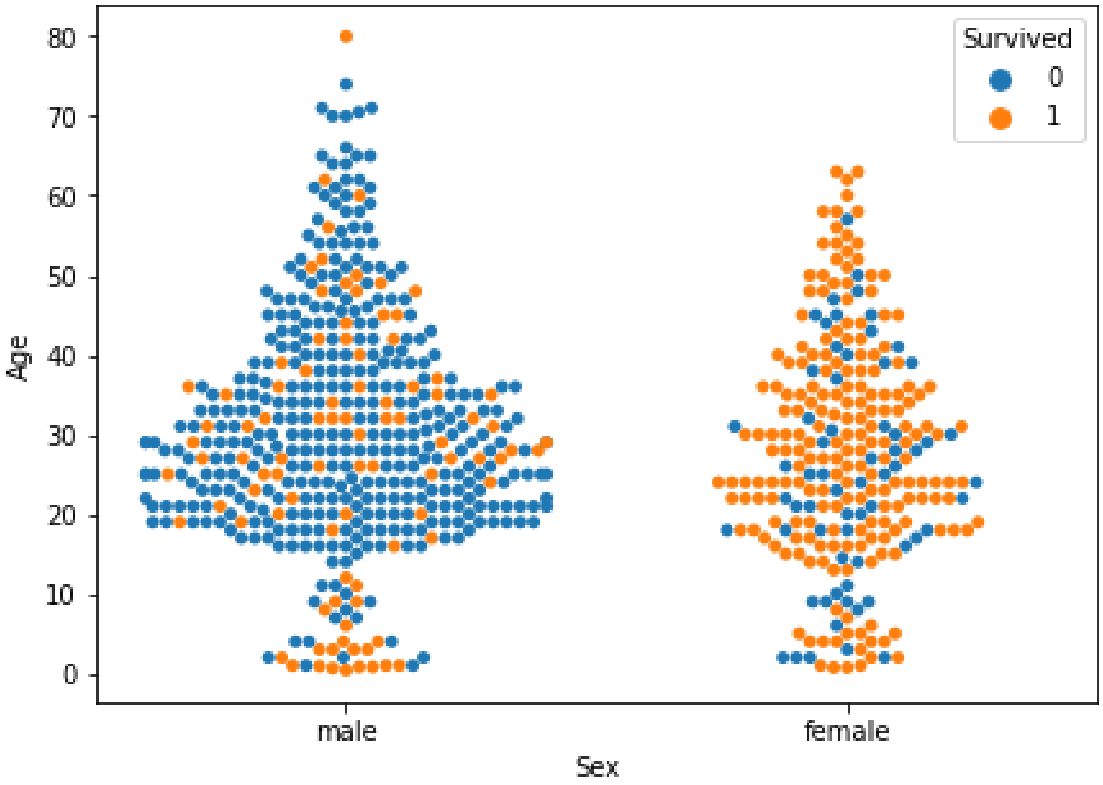
[46]:
```
import seaborn as sns
```

[47]:
```
import matplotlib.pyplot as plt
```

[52]:
```
fig, axis = plt.subplots( figsize=(7,5) )
sns.swarmplot( x='Sex', y='Age', hue='Survived', data=titanic, ax=axis )

#fig.savefig( 'testytest.pdf')
```

[52]:
```
<AxesSubplot:xlabel='Sex', ylabel='Age'>
```



```
[53]: type( fig )
```

```
[53]: matplotlib.figure.Figure
```

```
[55]: type( axis)
```

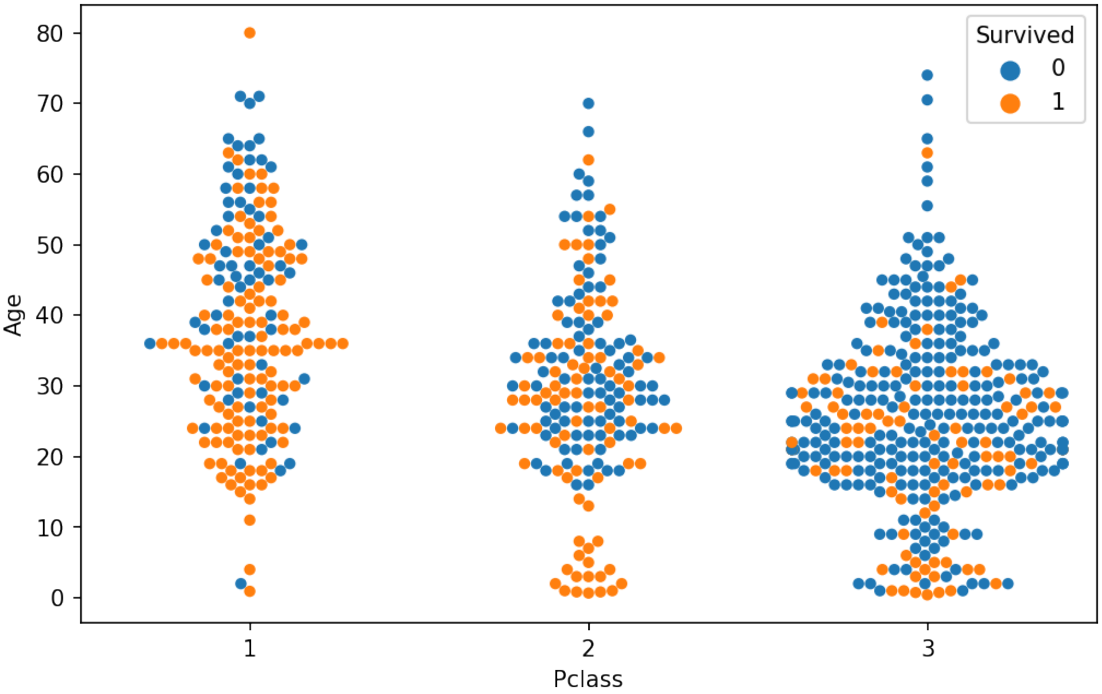
```
[55]: matplotlib.axes._subplots.AxesSubplot
```

```
[ ]: sns.swarmplot?
```

```
[90]: fig, axis = plt.subplots( figsize=(8, 5), dpi=150 )

      sns.swarmplot( x='Pclass', y='Age', hue='Survived', data=titanic, ax=axis)
```

```
[90]: <AxesSubplot:xlabel='Pclass', ylabel='Age'>
```



3.22 Subselecting based on one of the variables

```
[57]: titanic.shape
```

```
[57]: (891, 12)
```

```
[58]: titanic['Sex'].value_counts()
```

```
[58]: male      577
      female    314
      Name: Sex, dtype: int64
```

```
[59]: titanic['Sex'].head()
```

```
[59]: 0      male
      1    female
      2    female
      3    female
      4      male
      Name: Sex, dtype: object
```

```
[60]: titanic['Sex'] == 'male'
```

```
[60]: 0       True
      1      False
      2      False
      3      False
      4       True
             …
      886     True
      887    False
      888    False
      889     True
      890     True
      Name: Sex, Length: 891, dtype: bool
```

```
[61]: bool_array = titanic['Sex'] == 'male'
```

```
[62]: len(bool_array)
```

```
[62]: 891
```

```
[63]: (titanic['Sex'] == 'male').head()
```

```
[63]: 0     True
      1    False
      2    False
      3    False
      4     True
      Name: Sex, dtype: bool
```

```
[66]: males_only = titanic[ titanic['Sex'] == 'male' ]
```

```
[65]: males_only.head()
```

```
[65]:    PassengerId  Survived  Pclass                            Name   Sex   Age  \
      0            1         0       3         Braund, Mr. Owen Harris  male  22.0
      4            5         0       3        Allen, Mr. William Henry  male  35.0
      5            6         0       3                Moran, Mr. James  male   NaN
      6            7         0       1         McCarthy, Mr. Timothy J  male  54.0
      7            8         0       3  Palsson, Master. Gosta Leonard  male   2.0

         SibSp  Parch     Ticket     Fare Cabin Embarked
      0      1      0  A/5 21171   7.2500              S
      4      0      0     373450   8.0500              S
      5      0      0     330877   8.4583              Q
      6      0      0      17463  51.8625   E46        S
      7      3      1     349909  21.0750              S
```

```
[67]: males_only.shape
```

[67]: (577, 12)

[68]:
```
# Boolean selector array have to be the same shape as the array itself!!
bool_array = [True]*1000
```

[74]:
```
titanic[ bool_array ]
```

```
          ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-74-c15d9c7565a6> in <module>
   ----> 1 titanic[ bool_array ]


       /usr/local/lib/python3.8/site-packages/pandas/core/frame.py in␣
↪__getitem__(self, key)
      3013         # Do we have a (boolean) 1d indexer?
      3014         if com.is_bool_indexer(key):
   -> 3015             return self._getitem_bool_array(key)
      3016
      3017         # We are left with two options: a single key, and a␣
↪collection of keys,


       /usr/local/lib/python3.8/site-packages/pandas/core/frame.py in␣
↪_getitem_bool_array(self, key)
      3060             )
      3061         elif len(key) != len(self.index):
   -> 3062             raise ValueError(
      3063                 f"Item wrong length {len(key)} instead of {len(self.
↪index)}."
      3064             )


       ValueError: Item wrong length 1000 instead of 891.
```

[70]:
```
males_only.shape
```

[70]: (577, 12)

[71]:
```
gender_tf = titanic['Sex'] == 'male'
```

[72]:
```
gender_tf.shape
```

[72]:
```
(891,)
```

[73]:
```
males_only.shape
```

[73]:
```
(577, 12)
```

[77]:
```
females_only = titanic[ titanic['Sex'] == 'female']
```

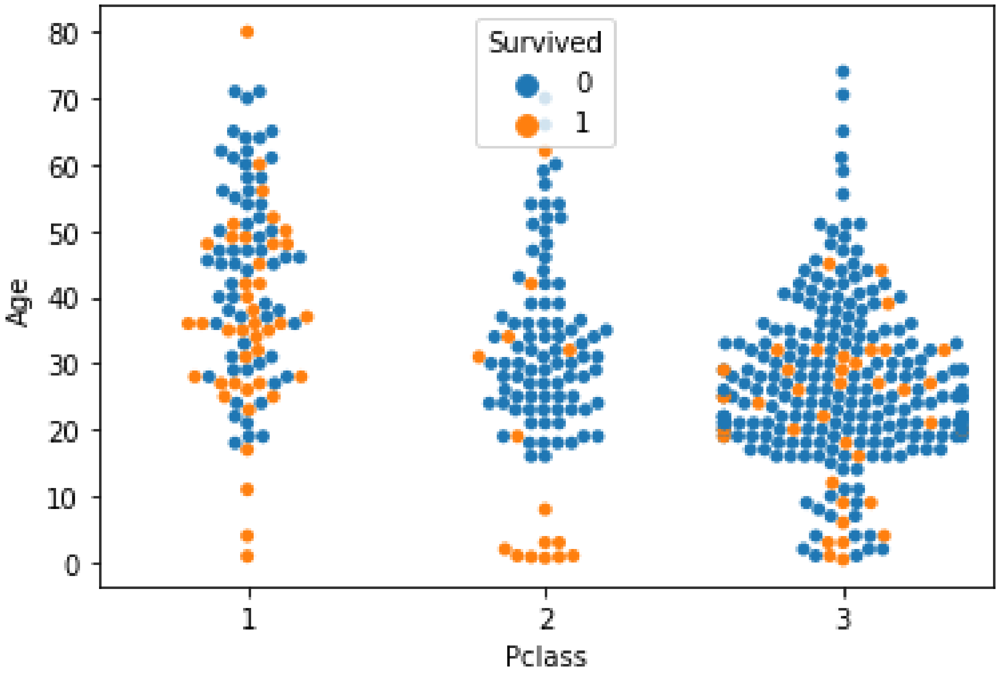
[78]:
```
females_only.shape
```

[78]:
```
(314, 12)
```

[80]:
```
sns.swarmplot( x='Pclass', y='Age', hue='Survived',
               data=males_only)
```

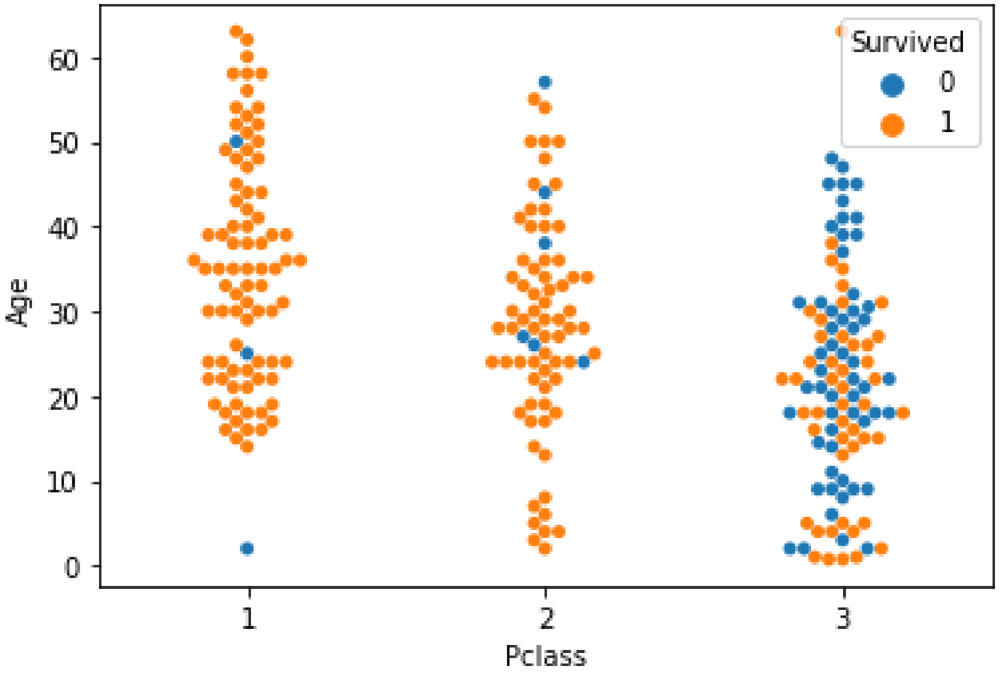
```
/usr/local/lib/python3.8/site-packages/seaborn/categorical.py:1296: UserWarning:
8.4% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

[80]:
```
<AxesSubplot:xlabel='Pclass', ylabel='Age'>
```



[91]:
```
sns.swarmplot( x='Pclass', y='Age', hue='Survived', data=females_only )
```

```
[91]: <AxesSubplot:xlabel='Pclass', ylabel='Age'>
```



3.23 Slicing by rows and columns using .loc[]

```
[92]: subset = titanic[ titanic['Age'] < 25 ]
```

```
[93]: subset.shape
```

```
[93]: (278, 12)
```

```
[94]: subset = titanic.loc[ titanic['Age'] < 25 ]
```

```
[95]: subset.shape
```

```
[95]: (278, 12)
```

4 Complex sort

```
[96]: age_bool = titanic['Age'] < 10
```

```
[97]: age_bool.value_counts()
```

```
[97]: False    829
      True      62
      Name: Age, dtype: int64
```

```
[98]: class_bool = titanic['Pclass'] == 1
```

```
[99]: class_bool.value_counts()
```

```
[99]: False    675
      True     216
      Name: Pclass, dtype: int64
```

```
[100]: age_class_bool = age_bool & class_bool
```

```
[101]: age_class_bool.value_counts()
```

```
[101]: False    888
       True       3
       dtype: int64
```

```
[103]: titanic.loc[ age_class_bool, 'Age' ]
```

```
[103]: 297    2.00
       305    0.92
       445    4.00
       Name: Age, dtype: float64
```

```
[104]: len(subset)
```

```
[104]: 278
```

4.1 Using .sort_values() for simple or complex sorting

```
[ ]: titanic.sort_values?
```

```
[106]: titanic.shape
```

```
[106]: (891, 12)
```

```
[107]: titanic.sort_values( by=['Pclass','Age'] ).head()
```

```
[107]:      PassengerId  Survived  Pclass                              Name  \
       305          306         1       1        Allison, Master. Hudson Trevor
       297          298         0       1          Allison, Miss. Helen Loraine
       445          446         1       1             Dodge, Master. Washington
       802          803         1       1  Carter, Master. William Thornton II
       435          436         1       1             Carter, Miss. Lucile Polk
```

```
        Sex    Age  SibSp  Parch  Ticket      Fare    Cabin Embarked
305    male   0.92      1      2  113781  151.5500  C22 C26        S
297  female   2.00      1      2  113781  151.5500  C22 C26        S
445    male   4.00      0      2   33638   81.8583      A34        S
802    male  11.00      1      2  113760  120.0000  B96 B98        S
435  female  14.00      1      2  113760  120.0000  B96 B98        S
```

```
[108]: titanic.sort_values( by=['Pclass','Age'], ascending=False ).head()
```

```
[108]:      PassengerId  Survived  Pclass                       Name     Sex   Age  \
       851          852         0       3        Svensson, Mr. Johan    male  74.0
       116          117         0       3       Connors, Mr. Patrick    male  70.5
       280          281         0       3           Duane, Mr. Frank    male  65.0
       483          484         1       3     Turkula, Mrs. (Hedwig)  female  63.0
       326          327         0       3  Nysveen, Mr. Johan Hansen    male  61.0

            SibSp  Parch  Ticket    Fare Cabin Embarked
       851      0      0  347060  7.7750              S
       116      0      0  370369  7.7500              Q
       280      0      0  336439  7.7500              Q
       483      0      0    4134  9.5875              S
       326      0      0  345364  6.2375              S
```