Learning Objectives

In the third module of this course, you will learn how to handle different scenarios in order to better manipulate your data.

Learning Objectives:

1. Locating missing values
2. Replacing missing values
3. Handling missing values

Handling Incomplete Data Sets

Missing values

Changing the representation of a dataset brings up an important subtlety of missing values. A value can be missing in one of two possible ways:

- **Explicitly**, i.e. flagged with `NA`.
- **Implicitly**, i.e. simply not present in the data.

Let’s illustrate this idea with a very simple data set:

```
stocks =[[2015,1,1.88],[2015,2,0.59],[2015,3,0.35], [2015,4]   ,
        [2016,2,0.92],[2016,3,0.17],[2016,4,2.66]]
df2 = pd.DataFrame(stocks,columns=['year','qtr','return'])
print (df2)
```

Note:

- NaN, None and NaT are standard missing values for Pandas.
- A new missing data type ‘NA’ introduced with Pandas 1.0 which is an integer type missing value representation.

There are two missing values in this dataset:

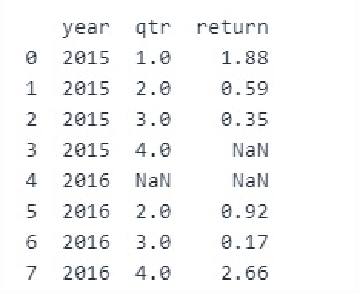
- The return for the fourth quarter of 2015 is explicitly missing, because the cell where its value should be instead contains `NaN`.

- The return for the first quarter of 2016 is implicitly missing, because it simply does not appear in the dataset.

An explicit missing value is the clear indication of the missing value.

An implicit missing value is the absence of data that is not necessarily clearly indicated.

The way that a dataset is represented can make implicit values explicit. For example, we can make the implicit missing value explicit by putting years in the columns:

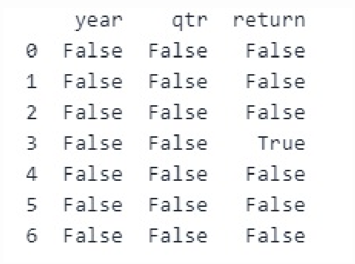


```
   year  qtr  return
0  2015  1.0    1.88
1  2015  2.0    0.59
2  2015  3.0    0.35
3  2015  4.0     NaN
4  2016  NaN     NaN
5  2016  2.0    0.92
6  2016  3.0    0.17
7  2016  4.0    2.66
```

images/img2

Return the Data Frame with boolean values indicating missing values.

```
print(df2.isna())
```



```
    year    qtr  return
0  False  False   False
1  False  False   False
2  False  False   False
3  False  False    True
4  False  False   False
5  False  False   False
6  False  False   False
```

images/cap4

Handling Missing Value

Depending on the question we are trying to answer or the task at hand, one can choose one of the following:
* Drop missing values
* Replace missing values

We can drop a row or column with missing values using the `dropna()` function. The `how` parameter is used to set the condition to drop.

`how='any'` : drop if there are any missing values
`how='all'` : drop if all values are missing

The `None` types mentioned on the previous page do not cover all the possible values of missing data. For example, “?” and “- -” characters in column of our Data Frame do not provide any valuable information or insight so they are missing values. However, these characters cannot be detected as missing value by Pandas.

1. If we know what kind of characters are used as missing values in the dataset, we can handle them while creating the Data Frame using the `na_values` parameter:

```
missing_value=['???','---']
dataset2= pd.readcsv("csv file
        location",na_values=missing_values)
```

2. Another option is to use the pandas `replace()` function to handle these values after a Data Frame is created. We can use this function to adjust our missing data into a format pandas expects such as `None`.

```
df.replace(['???':None,'---':None],inplace=True)
```

For example, we can replace some our values in df2 with `???` then try replacing them with the None value.

```
stocks =[[2015,1,'???'],[2015,2,0.59],[2015,'???',0.35],
[2015,4,None]  ,[2016,2,0.92],[2016,3,0.17],[2016,4,2.66]]
df2 = pd.DataFrame(stocks,columns=['year','qtr','return'])

df2.replace({'???' : None},inplace=True)
print (df2)
```