Data Manipulation & Analysis

Presentation by Namgyal BRISSON

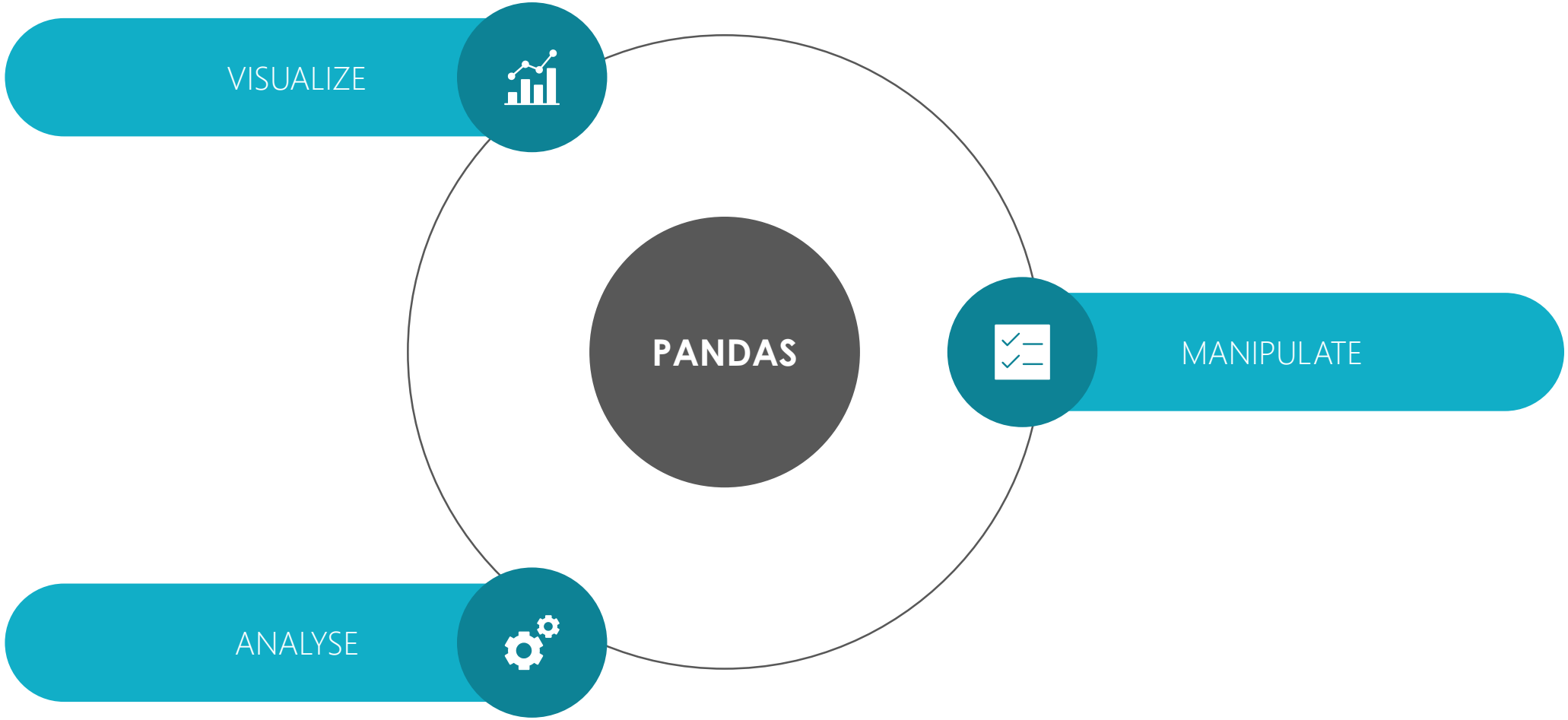
pandas

Data Analysis

Pandas a Data manipulation Tool

- ✓ Python library written by Wes McKinney
- ✓ Designed for Data manipulation & Analysis
- ✓ Allow to manipulate datatable with labels for column and row
- ✓ Those tables are called dataframes (pretty common to what you can find in R)
- ✓ Data can be easily read from or written to a tabular source (csv, excel, database, …)
- ✓ It is easy to graph data with pandas & matplotlib, bokeh, seaborn, …

Data Analysis



Data Analysis

Let's Practice

- ✓ https://github.com/nam4dev/pandas_introduction_presentation/blob/master/pandas_demo.ipynb
  - ✓ Click on « Open in Colab »
- ✓ Data collected from https://www.gapminder.org/data/

Loading and Saving Data

- ✓ DataFrame.read_<filetype>() => Reads the given filetype and loads it into a Data Frame
  - ✓ DataFrame.read_csv('filename.csv')
  - ✓ DataFrame.read_json('filename.json')

- ✓ DataFrame.to_<filetype>('filename.type') => Writes data frame to specified type
  - ✓ DataFrame.to_csv('filename.csv')
  - ✓ DataFrame.to_json('filename.json')

Viewing & Inspecting Data

- ✓ DataFrame.info() => Displays data type(s), index and memory information
- ✓ DataFrame.head(n=5) => Displays n (default to 5) first rows
- ✓ DataFrame.tail(n=5) => Displays n (default to 5) last rows
- ✓ DataFrame.shape => Displays number of rows and columns
- ✓ DataFrame.describe() => Displays summary statistics for numerical columns

Viewing & Inspecting Data (statistics)

- ✓ DataFrame.std() => Returns the standard deviation of each column
- ✓ DataFrame.min() => Returns the min value in each column
- ✓ DataFrame.max() => Returns the max value in each column
- ✓ DataFrame. corr() => Returns the correlation between columns
- ✓ DataFrame.mean() => Returns the mean of all columns
- ✓ DataFrame.median() => Returns the median of each column
- ✓ DataFrame.count() => Returns the number of non-null values in each column

Data Selection

- ✓ DataFrame['column'] => Returns a Pandas.Series of the selected column
- ✓ DataFrame[[c1, c2]] => Returns a new DataFrame of selected columns
- ✓ DataFrame.iloc[0] => Returns at position 0 in the data frame
- ✓ DataFrame.loc['index'] => Returns at index 'index in the data frame

Data Filtering & Sorting

- ✓ DataFrame[DataFrame['subscriptions'] > 100]
- ✓ => Returns a data frame with only greater than 100 subscriptions

- ✓ DataFrame.sort_values(c1)
- ✓ => Sorts the data frame in ascending order (default) for the given column

- ✓ DataFrame.sort_values([c1, c2], ascending=[False, False])
- ✓ => Sorts the data frame in given order for each given columns

Grouping Data By

- ✓ DataFrame.groupby(c1)
- ✓ => Returns a « groupby » instance for a single column

- ✓ DataFrame.groupby([c1, c2, …, cn])
- ✓ => Returns a « groupby » instance for multiple columns

Cleaning Data

- ✓ DataFrame.replace(0.0, np.nan)
- ✓ => Replaces all 0.0 values by « not a number » value (np here is the numpy import)

- ✓ DataFrame.dropna()
- ✓ => Drops « not a number » rows

- ✓ DataFrame.dropna(axis=1)
- ✓ => Drops « not a number » columns

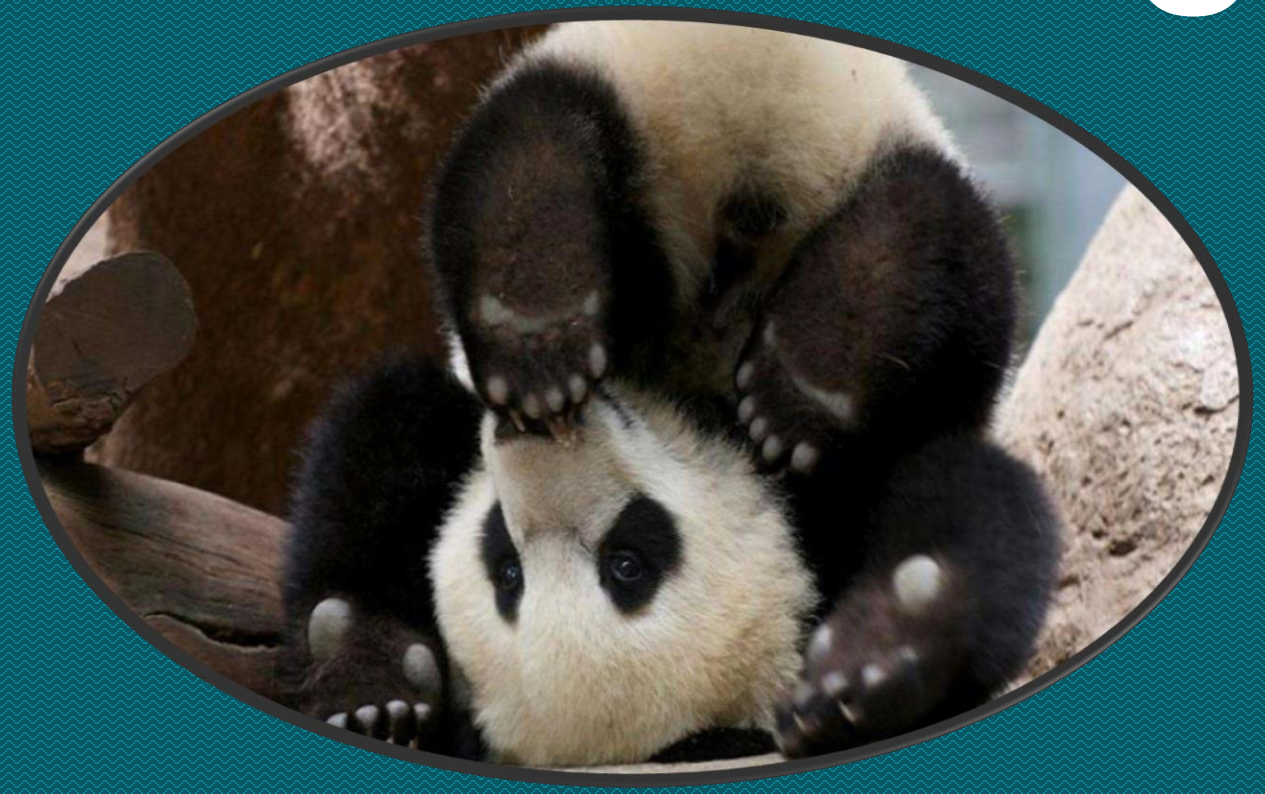
- ✓ DataFrame.fillna(DataFrame.mean())
- ✓ => Fills « not a number » with data frame mean value

Joining Data

- ✓ DataFrame1.join(DataFrame2, on=c1, how='inner')
- ✓ => It will inner join (like SQL) the columns in data frame 1 with columns on data frame 2 where the rows for c1 have the same values.

- ✓ Possible way of joining are:
  - ✓ inner
  - ✓ outer
  - ✓ right
  - ✓ left

Overwhelming?



Data Analysis

Python For Data Science *Cheat Sheet*

Pandas

Learn Python for Data Science Interactively at www.DataCamp.com

Reshaping Data

Pivot

```
>>> df3= df2.pivot(index='Date',
                   columns='Type',
                   values='Value')
```

Spread rows into columns

	Date	Type	Value
0	2016-03-01	a	11.432
1	2016-03-02	b	13.031
2	2016-03-01	c	20.784
3	2016-03-03	a	99.906
4	2016-03-02	a	1.303
5	2016-03-03	c	20.784

Type / Date	a	b	c
2016-03-01	11.432	NaN	20.784
2016-03-02	1.303	13.031	NaN
2016-03-03	99.906	NaN	20.784

Pivot Table

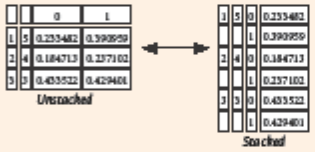
```
>>> df4 = pd.pivot_table(df2,
                         values='Value',
                         index='Date',
                         columns='Type'])
```

Spread rows into columns

Stack / Unstack

```
>>> stacked = df5.stack()
>>> stacked.unstack()
```

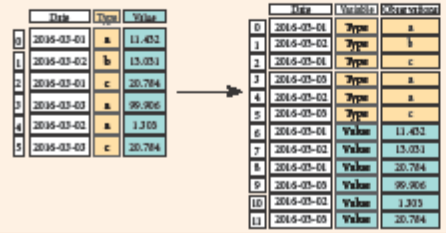
Pivot a level of column labels
Pivot a level of index labels



Melt

```
>>> pd.melt(df2,
            id_vars=["Date"],
            value_vars=["Type", "Value"],
            value_name="Observations")
```

Gather columns into rows



Iteration

```
>>> df.iteritems()
>>> df.iterrows()
```

(Column-index, Series) pairs
(Row-index, Series) pairs

Advanced Indexing

Also see NumPy Arrays

Selecting

```
>>> df3.loc[:,(df3>1).any()]
>>> df3.loc[:,(df3>1).all()]
>>> df3.loc[:,df3.isnull().any()]
>>> df3.loc[:,df3.notnull().all()]
```

Select cols with any vals>1
Select cols with vals > 1
Select cols with NaN
Select cols without NaN

Indexing With isin

```
>>> df[(df.Country.isin(df2.Type))]
>>> df3.filter(items="a","b"])
>>> df.select(lambda x: not x%5)
```

Find same elements
Filter on values
Select specific elements

Where

```
>>> s.where(s > 0)
```

Subset the data

Query

```
>>> df6.query('second > first')
```

Query DataFrame

Setting/Resetting Index

```
>>> df.set_index('Country')
>>> df4 = df.reset_index()
>>> df = df.rename(index=str,
                   columns={"Country":"cntry",
                            "Capital":"cptl",
                            "Population":"ppltn"})
```

Set the index
Reset the index
Rename DataFrame

Reindexing

```
>>> s2 = s.reindex(['a','c','d','e','b'])
```

Forward Filling

```
>>> df.reindex(range(4),
               method='ffill')
```

	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasilia	207847528
3	Brazil	Brasilia	207847528

Backward Filling

```
>>> s3 = s.reindex(range(5),
                   method='bfill')
```

0	3
1	3
2	3
3	3
4	3

MultiIndexing

```
>>> arrays = [np.array([1,2,3]),
              np.array([5,4,3])]
>>> df5 = pd.DataFrame(np.random.rand(3, 2), index=arrays)
>>> tuples = list(zip(*arrays))
>>> index = pd.MultiIndex.from_tuples(tuples,
                                      names=['first', 'second'])
>>> df6 = pd.DataFrame(np.random.rand(3, 2), index=index)
>>> df2.set_index(["Date", "Type"])
```

Duplicate Data

```
>>> s3.unique()
>>> df2.duplicated('Type')
>>> df2.drop_duplicates('Type', keep='last')
>>> df.index.duplicated()
```

Return unique values
Check duplicates
Drop duplicates
Check index duplicates

Grouping Data

Aggregation

```
>>> df2.groupby(by=['Date','Type']).mean()
>>> df4.groupby(level=0).sum()
>>> df4.groupby(level=0).agg({'a':lambda x:sum(x)/len(x),
                              'b': np.sum})
```

Transformation

```
>>> customSum = lambda x: (x+x%2)
>>> df4.groupby(level=0).transform(customSum)
```

Missing Data

```
>>> df.dropna()
>>> df3.fillna(df3.mean())
>>> df2.replace("a", "f")
```

Drop NaN values
Fill NaN values with a predetermined value
Replace values with others

Combining Data

Merge

```
>>> pd.merge(data1,
             data2,
             how='left',
             on='X1')
```

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN

```
>>> pd.merge(data1,
             data2,
             how='right',
             on='X1')
```

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
d	NaN	20.784

```
>>> pd.merge(data1,
             data2,
             how='inner',
             on='X1')
```

X1	X2	X3
a	11.432	20.784
b	1.303	NaN

```
>>> pd.merge(data1,
             data2,
             how='outer',
             on='X1')
```

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN
d	NaN	20.784

Join

```
>>> data1.join(data2, how='right')
```

Concatenate

Vertical

```
>>> s.append(s2)
```

Horizontal/Vertical

```
>>> pd.concat([s,s2],axis=1, keys=['One','Two'])
>>> pd.concat([data1, data2], axis=1, join='inner')
```

Dates

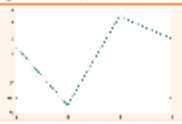
```
>>> df2['Date']= pd.to_datetime(df2['Date'])
>>> df2['Date']= pd.date_range('2000-1-1',
                               periods=6,
                               freq='M')
>>> dates = [datetime(2012,5,1), datetime(2012,5,2)]
>>> index = pd.DatetimeIndex(dates)
>>> index = pd.date_range(datetime(2012,2,1), end, freq='BM')
```

Visualization

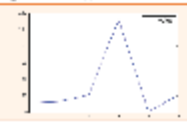
Also see Matplotlib

```
>>> import matplotlib.pyplot as plt
```

```
>>> s.plot()
>>> plt.show()
```



```
>>> df2.plot()
>>> plt.show()
```



DataCamp
Learn Python For Data Science Interactively