Lesson #09
Transforming Data with Pandas

March 2019

Introduction to Pandas
Exploring Data with Pandas
Data Cleaning Basics
Data Aggregation
Combining Data with Pandas
Transforming Data with Pandas
Working with String in Pandas
Working with missing and duplicate data

- Transforming data with Pandas
  a. Map
  b. Apply
  c. Applymap
  d. Melt

LEARNING TODAY

Update from repository

```
git clone https://github.com/ivanovitchm/datascience_one_2019_1
```

Or ....

```
git pull
```

Dataset

745

World Happiness Report

Happiness scored according to economic production, social support, etc.

Sustainable Development Solutions Network • updated 2 years ago (Version 2)

economics, social sciences, emotion

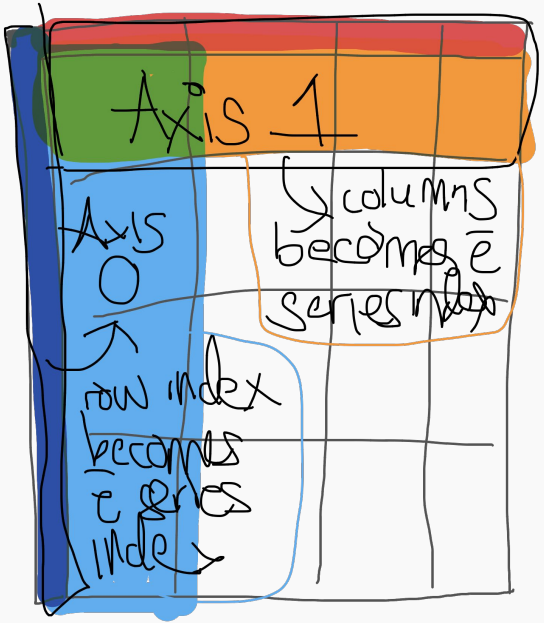
Description

Context

The World Happiness Report is a landmark survey of the state of global happiness. The first report was published in 2012, the second in 2013, the third in 2015, and the fourth in the 2016 Update. The World Happiness 2017, which ranks 155 countries by their happiness levels, was released at the United Nations at an event celebrating International Day of Happiness on March 20th. The report continues to gain global recognition as governments, organizations and civil society increasingly use happiness indicators to inform their policy-making decisions. Leading experts across fields – economics, psychology, survey analysis, national statistics, health, public policy and

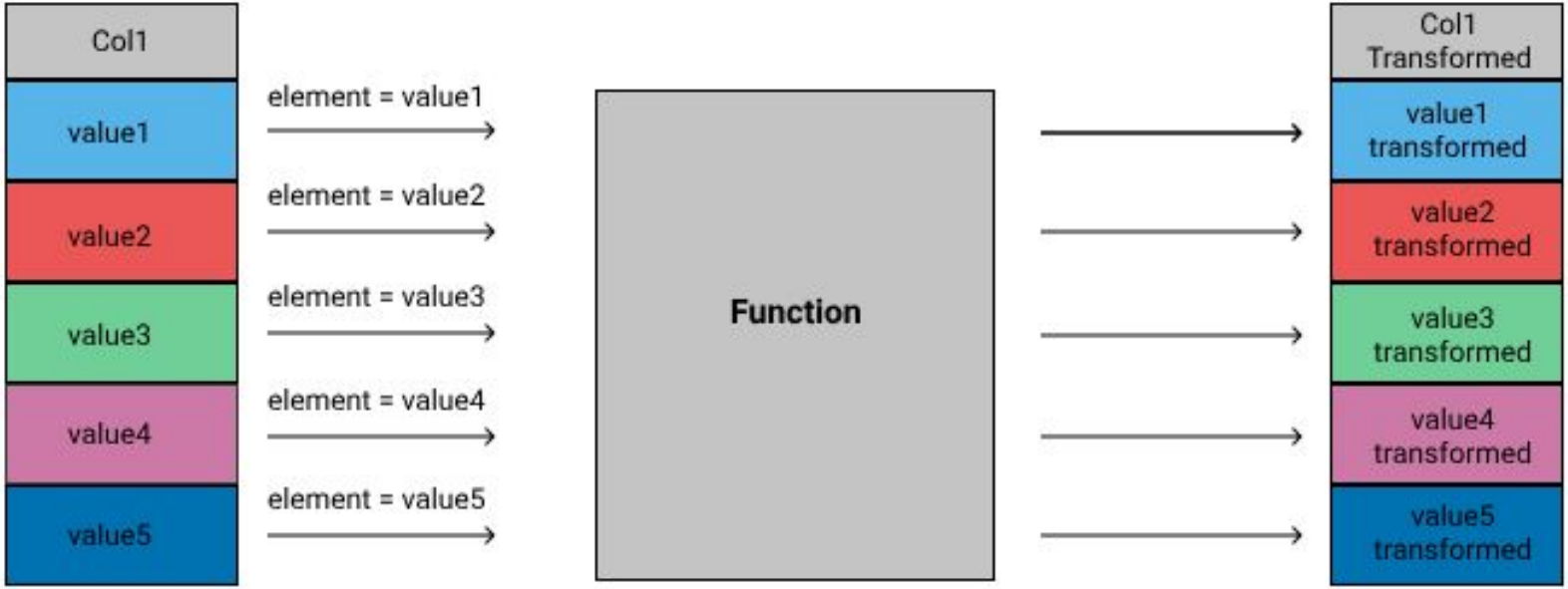
Which of these factors contribute the most to the happiness score?

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy	Family	Health	Freedom	Trust	Generosity
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811



- Series.map()
- Series.apply()
- DataFrame.applymap()
- DataFrame.apply()
- Pandas.melt()

Apply a Function Element-wise: **map**, **apply**



```python
def label(element):
    if element > 1:
        return 'High'
    else:
        return 'Low'
happiness2015['Economy Impact'] = happiness2015['Economy'].map(label)
happiness2015['Economy Impact'] = happiness2015['Economy'].apply(label)
```

Series.map()
Series.apply()

	Economy	Economy Impact
0	1.39651	High
1	1.30232	High
2	1.32548	High
3	1.45900	High
4	1.32629	High

```
def label(element, x):
    if element > x:
        return 'High'
    else:
        return 'Low'
economy_map = happiness2015['Economy'].map(label, x = .8)
```

```
TypeError: map() got an unexpected keyword argument 'x'
```

```
def label(element,x):
    if element > x:
        return 'High'
    else:
        return 'Low'
economy_impact_apply = happiness2015['Economy'].apply(label,x=0.8)
```

Apply a Function Element-wise to Multiple Columns Using Applymap Method

```
def label(element):
    if element > 1:
        return 'High'
    else:
        return 'Low'
happiness2015['Economy Impact'] = happiness2015['Economy'].apply(label)
happiness2015['Health Impact'] = happiness2015['Health'].apply(label)
happiness2015['Family Impact'] = happiness2015['Family'].apply(label)
```

Dataframe.applymap()

```
factors = ['Economy', 'Family', 'Health', 'Freedom', 'Trust', 'Generosity']
factors_impact = happiness2015[factors].applymap(label)
```

	Economy	Family	Health	Freedom	Trust	Generosity
0	High	High	Low	Low	Low	Low
1	High	High	Low	Low	Low	Low
2	High	High	Low	Low	Low	Low
3	High	High	Low	Low	Low	Low
4	High	High	Low	Low	Low	Low

```
def label(element):
    if element > 1:
        return 'High'
    else:
        return 'Low'
factors = ['Economy', 'Family', 'Health', 'Freedom', 'Trust', 'Generosity']
factors_impact = happiness2015[factors].apply(label)
```

Dataframe.apply()

```
ValueError: ('The truth value of a Series is ambiguous.
```

```
factors_impact
```

	Economy	Family	Health	Freedom	Trust	Generosity
0	High	High	Low	Low	Low	Low
1	High	High	Low	Low	Low	Low
2	High	High	Low	Low	Low	Low
3	High	High	Low	Low	Low	Low
4	High	High	Low	Low	Low	Low

Apply Functions along an Axis using the Apply Method

```
factors_impact.apply(pd.value_counts)
```

	Economy	Family	Health	Freedom	Trust	Generosity
High	66	89	2	NaN	NaN	NaN
Low	92	69	156	158.0	158.0	158.0

factors_impact.apply(pd.value_counts)

factors_impact ['Economy']. value_counts()

factors_impact ['Family']. value_counts()

factors_impact ['Health']. value_counts()

factors_impact ['Freedom']. value_counts()

factors_impact ['Trust']. value_counts()

factors_impact ['Generosity']. value_counts()

	Economy	Family	Health	Freedom	Trust	Generosity
High	66	89	2	NaN	NaN	NaN
Low	92	69	156	158.0	158.0	158.0

```python
def v_counts(col):
    num = col.value_counts()
    den = col.size
    return num/den
factors_impact.apply(v_counts)
```

	Economy	Family	Health	Freedom	Trust	Generosity
High	0.417722	0.563291	0.012658	NaN	NaN	NaN
Low	0.582278	0.436709	0.987342	1.0	1.0	1.0

	Country	Happiness Score	Economy	Family	Health
0	Switzerland	7.587	1.39651	1.34951	0.94143
1	Iceland	7.561	1.30232	1.40223	0.94784

```
pd.melt(happy_two, id_vars=['Country'], value_vars=['Economy', 'Family', 'Health'])
```

	Country	variable	value
0	Switzerland	Economy	1.39651
1	Iceland	Economy	1.30232
2	Switzerland	Family	1.34951
3	Iceland	Family	1.40223
4	Switzerland	Health	0.94143
5	Iceland	Health	0.94784

Reshaping Data with the Melt Function #tidydata

Lesson #09
1 - Transforming Data With Pandas