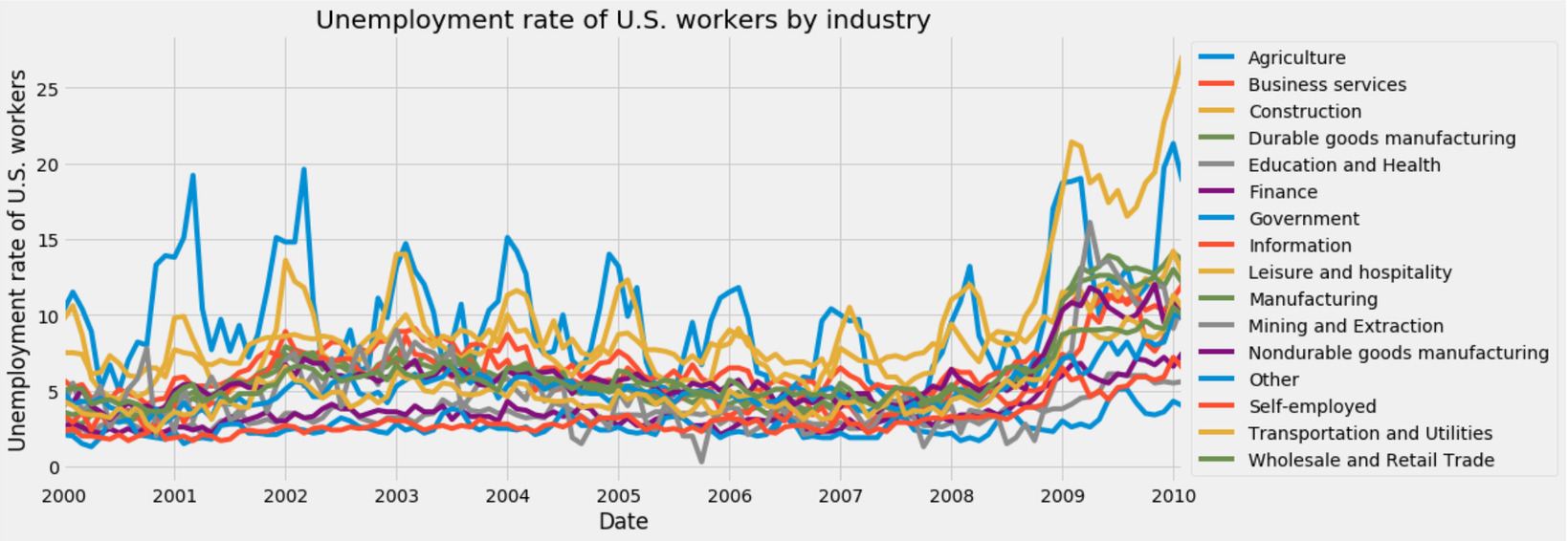
Apply your knowledge to a new dataset

VISUALIZING TIME SERIES DATA IN PYTHON

Thomas Vincent
Head of Data Science, Getty Images

The Jobs dataset



Let's get started!

VISUALIZING TIME SERIES DATA IN PYTHON

Beyond summary statistics

VISUALIZING TIME SERIES DATA IN PYTHON

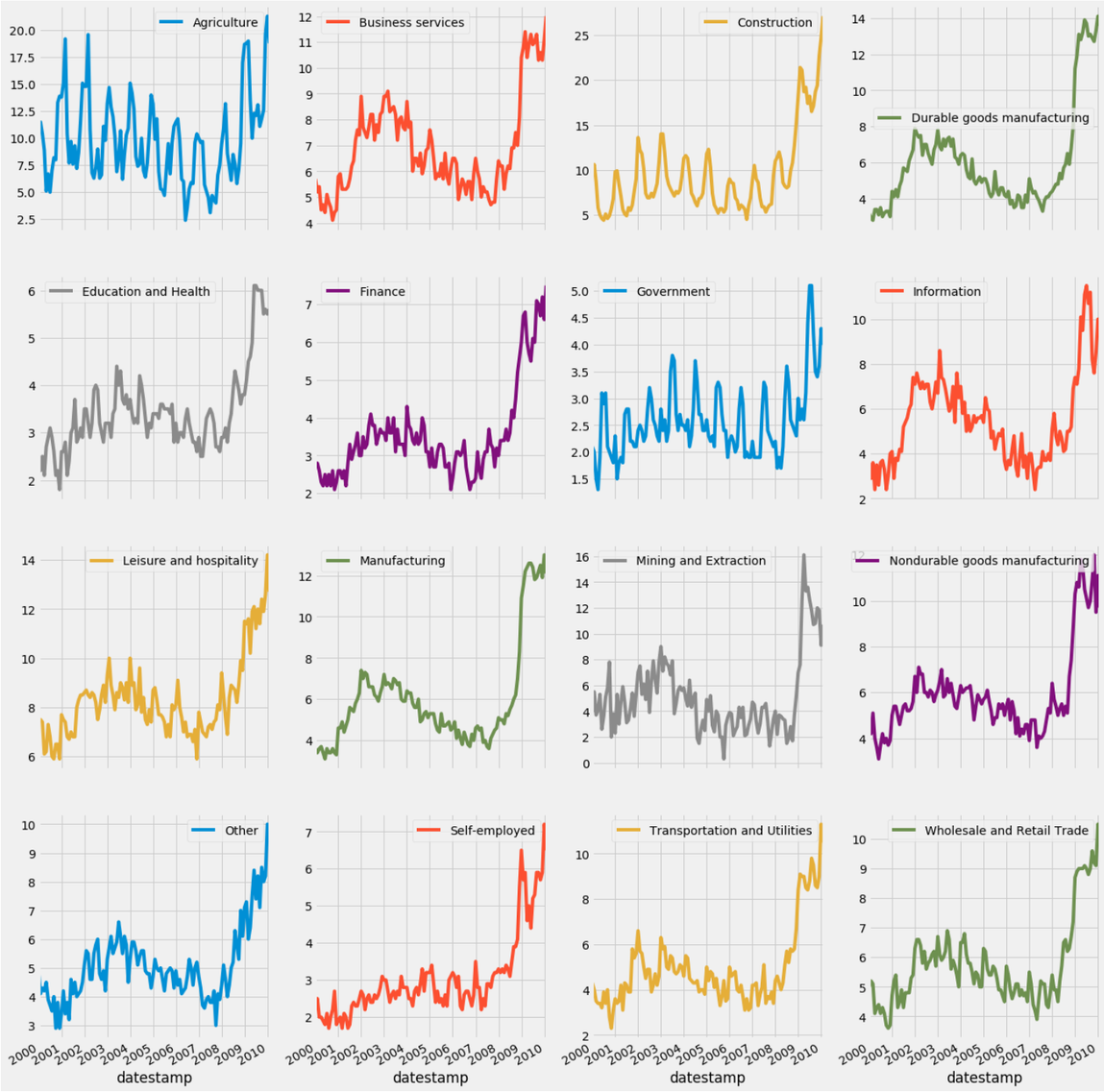
Thomas Vincent
Head of Data Science, Getty Images

datacamp

Facet plots of the jobs dataset

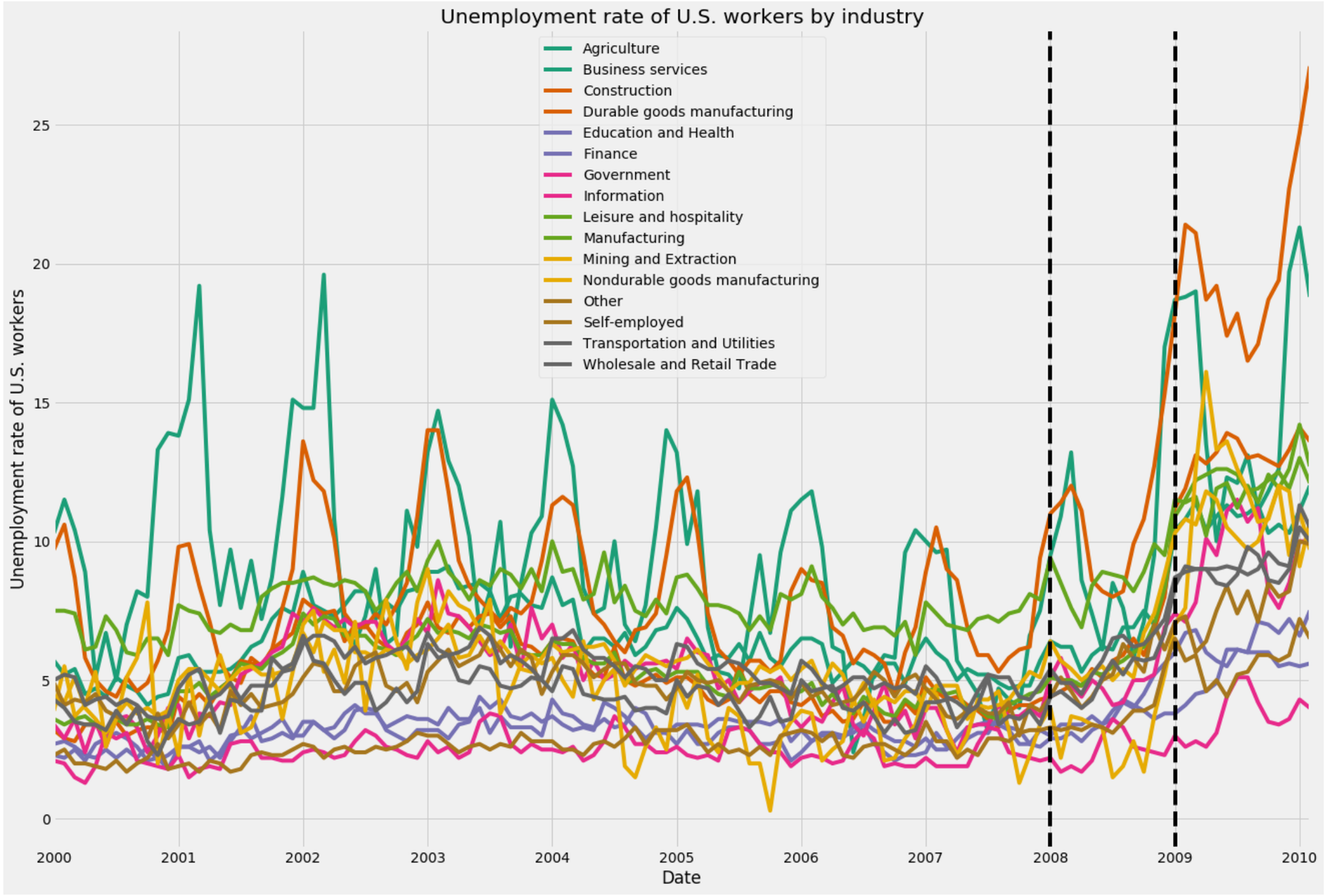
```
jobs.plot(subplots=True,
             layout=(4, 4),
             figsize=(20, 16),
             sharex=True,
             sharey=False)

plt.show()
```


Agriculture
Business services
Construction
Durable goods manufacturing
Education and Health
Finance
Government
Information
Leisure and hospitality
Manufacturing
Mining and Extraction
Nondurable goods manufacturing
Other
Self-employed
Transportation and Utilities
Wholesale and Retail Trade
datestamp
datestamp
datestamp
datestamp

Annotating events in the jobs dataset

```
ax = jobs.plot(figsize=(20, 14), colormap='Dark2')
ax.axvline('2008-01-01', color='black',
           linestyle='--')
ax.axvline('2009-01-01', color='black',
           linestyle='--')
```


Unemployment rate of U.S. workers by industry
Agriculture
Business services
Construction
Durable goods manufacturing
Education and Health
Finance
Government
Information
Leisure and hospitality
Manufacturing
Mining and Extraction
Nondurable goods manufacturing
Other
Self-employed
Transportation and Utilities
Wholesale and Retail Trade
Unemployment rate of U.S. workers
0
5
10
15
20
25
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
Date

Taking seasonal average in the jobs dataset

```
print(jobs.index)
```

```
DatetimeIndex(['2000-01-01', '2000-02-01', '2000-03-01',
'2000-04-01', '2009-09-01','2009-10-01',
'2009-11-01', '2009-12-01','2010-01-01', '2010-02-01'],
dtype='datetime64[ns]', name='datestamp',
length=122, freq=None)
```

```
index_month = jobs.index.month
jobs_by_month = jobs.groupby(index_month).mean()
print(jobs_by_month)
```

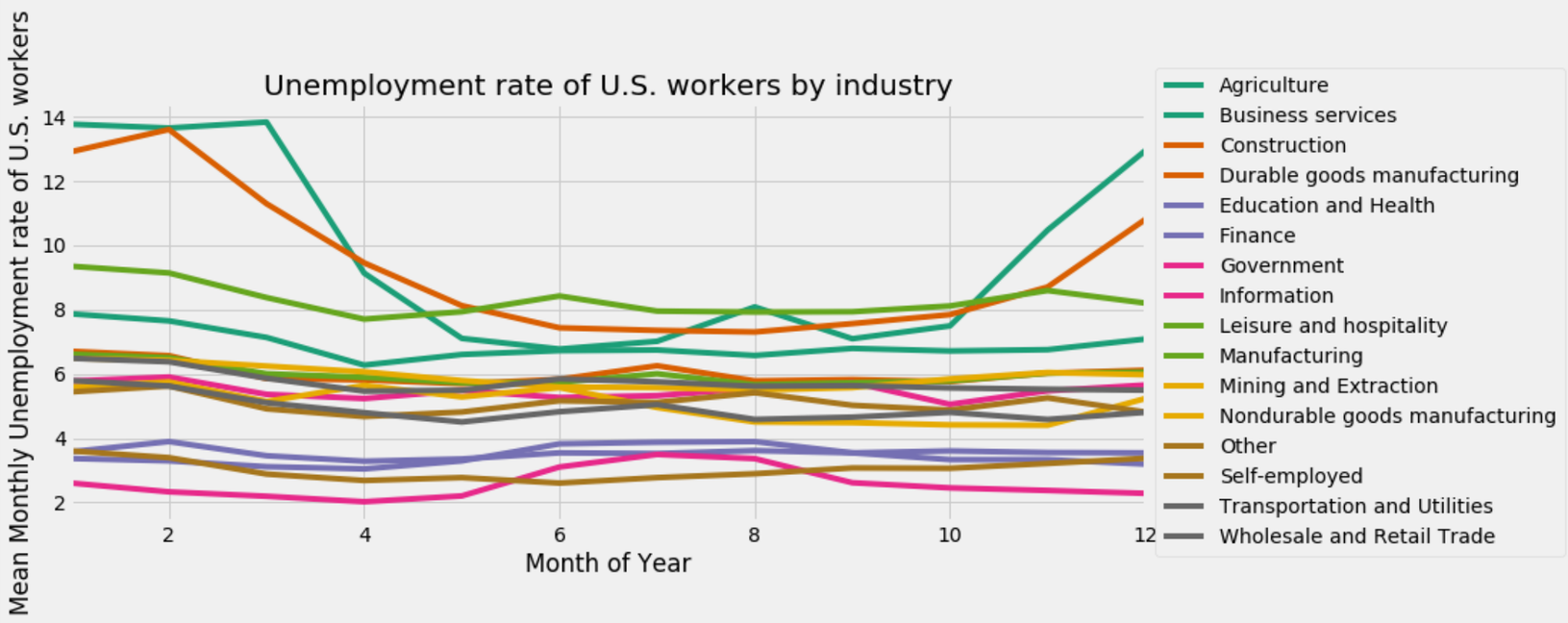
```
datestamp    Agriculture  Business services  Construction
1            13.763636           7.863636     12.909091
2            13.645455           7.645455     13.600000
3            13.830000           7.130000     11.290000
4             9.130000           6.270000      9.450000
5             7.100000           6.600000      8.120000
...
```

Monthly averages in the jobs dataset

```
ax = jobs_by_month.plot(figsize=(12, 5),
colormap='Dark2')
```

```
ax.legend(bbox_to_anchor=(1.0, 0.5),
loc='center left')
```

Monthly averages in the jobs dataset



Time to practice!

VISUALIZING TIME SERIES DATA IN PYTHON

Decompose time series data

VISUALIZING TIME SERIES DATA IN PYTHON

Thomas Vincent
Head of Data Science, Getty Images

Python dictionaries

```
# Initialize a Python dictionnary
my_dict = {}

# Add a key and value to your dictionnary
my_dict['your_key'] = 'your_value'

# Add a second key and value to your dictionnary
my_dict['your_second_key'] = 'your_second_value'

# Print out your dictionnary
print(my_dict)
```

```
{'your_key': 'your_value',
 'your_second_key': 'your_second_value'}
```

Decomposing multiple time series with Python dictionaries

```
# Import the statsmodel library
import statsmodels.api as sm
# Initialize a dictionary
my_dict = {}
# Extract the names of the time series
ts_names = df.columns
print(ts_names)
```

```
['ts1', 'ts2', 'ts3']
```

```
# Run time series decomposition
for ts in ts_names:
    ts_decomposition = sm.tsa.seasonal_decompose(jobs[ts])
    my_dict[ts] = ts_decomposition
```

Extract decomposition components of multiple time series

```
# Initialize a new dictionnary
my_dict_trend = {}
# Extract the trend component
for ts in ts_names:
    my_dict_trend[ts] = my_dict[ts].trend
# Convert to a DataFrame
trend_df = pd.DataFrame.from_dict(my_dict_trend)
print(trend_df)
```

```
            ts1  ts2 ts3
datestamp
2000-01-01  2.2  1.3 3.6
2000-02-01  3.4  2.1 4.7
...
```

Python dictionaries for the win!

VISUALIZING TIME SERIES DATA IN PYTHON

Compute correlations between time series

VISUALIZING TIME SERIES DATA IN PYTHON

Thomas Vincent
Head of Data Science, Getty Images

Trends in Jobs data

```
print(trend_df)
```

```
datestamp  Agriculture  Business services  Construction

2000-01-01          NaN                NaN           NaN
2000-02-01          NaN                NaN           NaN
2000-03-01          NaN                NaN           NaN
2000-04-01          NaN                NaN           NaN
2000-05-01          NaN                NaN           NaN
2000-06-01          NaN                NaN           NaN
2000-07-01     9.170833           4.787500      6.329167
2000-08-01     9.466667           4.820833      6.304167
...
```

Plotting a clustermap of the jobs correlation matrix

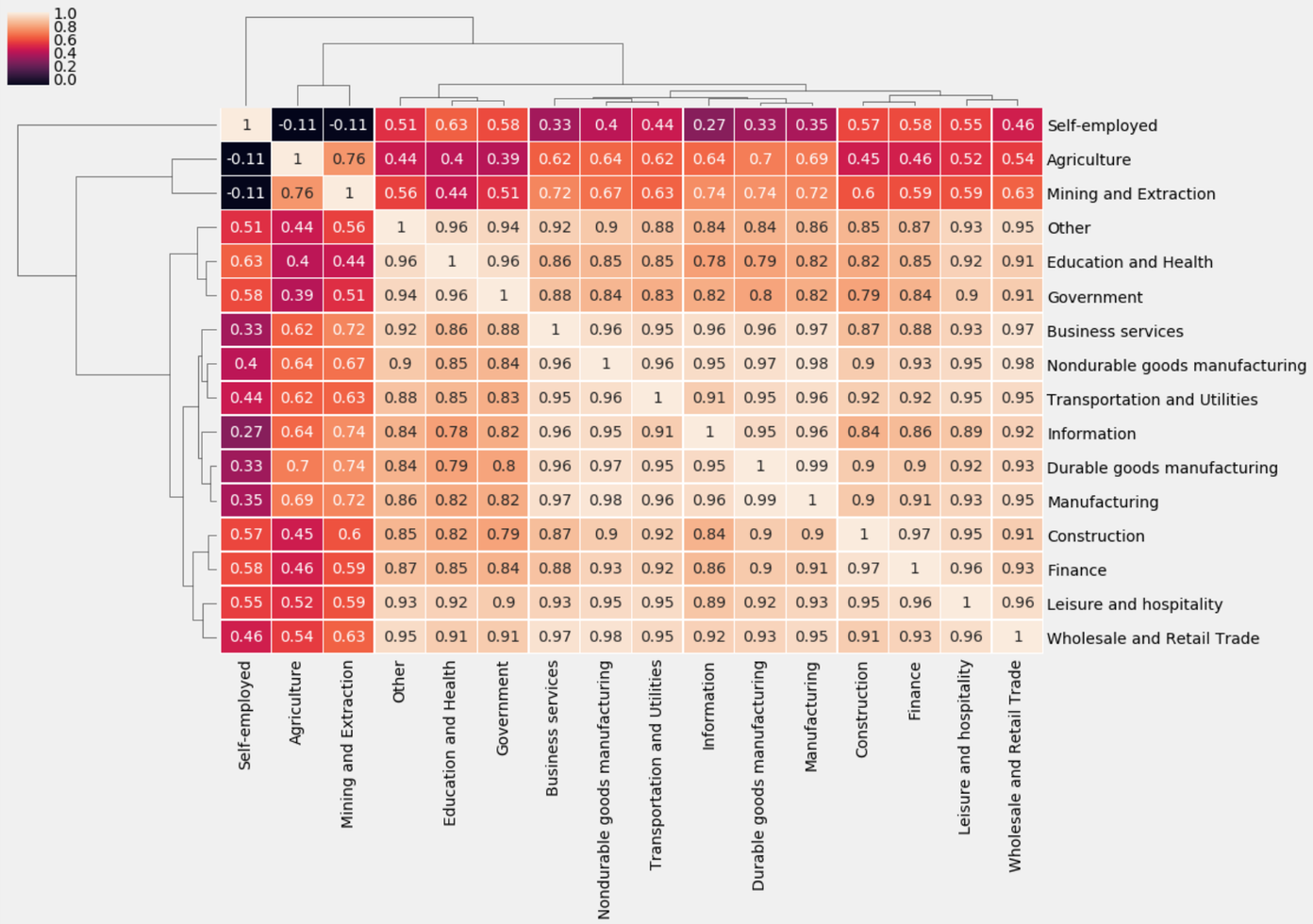
```
# Get correlation matrix of the seasonality_df DataFrame
trend_corr = trend_df.corr(method='spearman')

# Customize the clustermap of the seasonality_corr
correlation matrix
fig = sns.clustermap(trend_corr, annot=True, linewidth=0.4)

plt.setp(fig.ax_heatmap.yaxis.get_majorticklabels(),
rotation=0)

plt.setp(fig.ax_heatmap.xaxis.get_majorticklabels(),
rotation=90)
```

The jobs correlation matrix



Let's practice!

VISUALIZING TIME SERIES DATA IN PYTHON

Congratulations!

VISUALIZING TIME SERIES DATA IN PYTHON

Thomas Vincent
Head of Data Science, Getty Images

Going further with time series

- Data from Zillow Research
- Kaggle competitions
- Reddit Data

Going further with time series

- The importance of time series in business:
  - to identify seasonal patterns and trends
  - to study past behaviors
  - to produce robust forecasts
  - to evaluate and compare company achievements

Getting to the next level

- Manipulating Time Series Data in Python
- Importing & Managing Financial Data in Python
- Statistical Thinking in Python (Part 1)
- Supervised Learning with scikit-learn

Thank you!

VISUALIZING TIME SERIES DATA IN PYTHON