CFDS® – Chartered Financial Data Scientist

Introduction to Python

Prof. Dr. Natalie Packham

27 November and 11 December 2024

Table of Contents

4 Financial Time Series

- Time series are ubiquitous in finance.
- `pandas` is the main library in Python to deal with time series.

4.1 Financial Data

Financial data

- For the time being we work with locally stored data files.
- These are in `.csv` -files (comma-separated values), where the data entries in each row are separated by commas.
- Some initialisation:

In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Data import

- pandas provides a numer of different functions and DataFrame methods for importing and exporting data.
- Here we use pd.read_csv() .
- The file that we load contains end-of-day data for different financial instruments retrieved from Thomson Reuters.

In [ ]:

```
# If using colab, then uncomment the line below and comment the line after that
#filename = 'https://raw.githubusercontent.com/packham/Python_CFDS/main/data/tr_eik
filename = './data/tr_eikon_eod_data.csv' # path and filename
f = open(filename, 'r') # this will give an error when using colab; just ignore it
f.readlines()[:5]  # show first five lines
```

Data import

In [ ]:

```
data = pd.read_csv(filename,  # import csv-data into DataFrame
                   index_col=0, # take first column as index
                   parse_dates=True)  # index values are datetime
```

In [ ]:

```
data.info()  # information about the DataFrame object
```

Data import

In [ ]:

```
data.head()
```

Data import

In [ ]:

```
data.tail()
```

Data import

In [ ]:

```
data.plot(figsize=(10, 10), subplots=True);
```

Data import

- The identifiers used by Thomson Reuters are so-called RIC's.
- The financial instruments in the data set are:

In [ ]:

```
instruments = ['Apple Stock', 'Microsoft Stock',
               'Intel Stock', 'Amazon Stock', 'Goldman Sachs Stock',
               'SPDR S&P 500 ETF Trust', 'S&P 500 Index',
               'VIX Volatility Index', 'EUR/USD Exchange Rate',
               'Gold Price', 'VanEck Vectors Gold Miners ETF',
               'SPDR Gold Trust']
```

Data import

In [ ]:

```
for ric, name in zip(data.columns, instruments):
    print('{:8s} | {}'.format(ric, name))
```

Summary statistics

In [ ]:

```
data.describe().round(2)
```

Summary statistics

- The `aggregate()` -function allows to customise the statistics viewed:

In [ ]:

```
data.aggregate([min,
                np.mean,
                np.std,
                np.median,
                max]
).round(2)
```

Returns

- When working with financial data we typically (=always - you must have good reasons to deviate from this) work with performance data, i.e., **returns**.
- Reasoning:
  - Historical data are mainly used to make forecasts one or several time periods forward.
  - The daily average stock price over the last eight years is meaningless to make a forecast for tomorrow's stock price.
  - However, the daily returns are possible scenarios for the next time period(s).
- The function `pct_change()` calculates discrete returns:

$$r_t^{\mathrm{d}} = \frac{S_t - S_{t-1}}{S_{t-1}},$$

  where S_t denotes the stock price at time t.

Returns

In [ ]:

```
data.pct_change().round(3).head()
```

Returns

In [ ]:

```
data.pct_change().mean().plot(kind='bar', figsize=(10, 6));
```

Returns

- In finance, **log-returns**, also called **continuous returns**, are often preferred over discrete returns: $r_t^c = \ln\left(\frac{S_t}{S_{t-1}}\right)$.
- The main reason is that log-return are additive over time.
- For example, the log-return from $t-1$ to $t+1$ is the sum of the single-period log-returns:
$$r_{t-1,t+1}^c = \ln\left(\frac{S_{t+1}}{S_t}\right) + \ln\left(\frac{S_t}{S_{t-1}}\right) = \ln\left(\frac{S_{t+1}}{S_t} \cdot \frac{S_t}{S_{t-1}}\right) = \ln\left(\frac{S_{t+1}}{S_{t-1}}\right).$$
- Note: If the sampling (time) interval is small (e.g. one day or one week), then the difference between discrete returns and log-returns is negligible.

Returns

In [ ]:

```
rets = np.log(data / data.shift(1))  # calculates log-returns in a vectorised way
```

In [ ]:

```
rets.head().round(3)
```

Returns

In [ ]:

```
rets.cumsum().apply(np.exp).plot(figsize=(10, 6));  # recover price paths from log-r
```

4.2 Correlation analysis and linear regression

- To further illustrate how to work with financial time series we consider the S&P 500 stock index and the VIX volatility index.
- Empirical stylised fact: As the S&P 500 rises, the VIX falls, and vice versa.
- Note: This is about **correlation** not **causation**.

Correlation analysis

In [ ]:

```
# EOD data from Thomson Reuters Eikon Data API

# If using colab, then uncomment the line below and comment the line after that
#raw = pd.read_csv('https://raw.githubusercontent.com/packham/Python_CFDS/main/data/
raw = pd.read_csv('./data/tr_eikon_eod_data.csv', index_col=0, parse_dates=True)
data = raw[['.SPX', '.VIX']].dropna()
data.tail()
```

Correlation analysis

In [ ]:

```
data.plot(subplots=True, figsize=(10, 6));
```

Correlation analysis

- Transform both data series into log-returns:

In [ ]:

```
rets = np.log(data / data.shift(1))
rets.head()
```

In [ ]:

```
rets.dropna(inplace=True) # drop NaN (not-a-number) entries
```

Correlation analysis

In [ ]:

```
rets.plot(subplots=True, figsize=(10, 6));
```

Correlation analysis

In [ ]:

```
pd.plotting.scatter_matrix(rets,
                           alpha=0.2,
                           diagonal='hist',
                           hist_kwds={'bins': 35},
                           figsize=(10, 6));
```

Correlation analysis

In [ ]:

```
rets.corr()
```

OLS regression

- **Linear regression** captures the linear relationship between two variables.
- For two variables x, y, we postulate a linear relationship:
$$y = \alpha + \beta x + \varepsilon, \qquad \alpha, \beta \in \mathbb{R}.$$
- Here, α is the **intercept**, β is the **slope (coefficient)** and ε is the **error term**.
- Given data sample of joint observations $(x_1, y_1), \dots, (x_n, y_n)$, we set
$$y_i = \hat{\alpha} + \hat{\beta} x_i + \hat{\varepsilon}_i,$$
where $\hat{\alpha}$ and $\hat{\beta}$ are estimates of α, β and $\hat{\varepsilon}_1, \dots, \hat{\varepsilon}_n$ are the so-called **residuals.**
- The **ordinary least squares (OLS)** estimator $\hat{\alpha}, \hat{\beta}$ corresponds to those values of α, β that minimise the sum of squared residuals:
$$\min_{\alpha,\beta} \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2.$$

OLS regressions

- Simplest form of OLS regression:

In [ ]:

```
reg = np.polyfit(rets['.SPX'], rets['.VIX'], deg=1)  # fit a linear equation (a poly
reg.view() # the fitted paramters
```

2.62e-03 is scientific notation: $2.62e - 03 = 2.62 \cdot 10^{-3}$.

In [ ]:

```
ax = rets.plot(kind='scatter', x='.SPX', y='.VIX', figsize=(8, 5))
ax.plot(rets['.SPX'], np.polyval(reg, rets['.SPX']), 'r', lw=2);
```

OLS regression

- To do a more refined OLS regression with a proper analysis, use the package `statsmodels`.

In [ ]:

```
import statsmodels.api as sm

Y=rets['.VIX']
X=rets['.SPX']
X = sm.add_constant(X)
```

In [ ]:

```
model = sm.OLS(Y,X)
results = model.fit()
```

In [ ]:

```
results.params
```

In [ ]:

```
results.predict()[0:10]
```

OLS regression

In [ ]:

```
print(results.summary())
```

OLS regression: Interpretation of output and forecasting

- The column `coef` lists the coefficients of the regression: the coefficient in the row labelled `const` corresponds to $\hat{\alpha}$ (= 0.0026) and the coefficient in the row `.SPX` denotes $\hat{\beta}$ (= -6.6515).
- The estimated model in the example is thus:

$$.\mathrm{VIX} = 0.0026 - 6.6516 . \mathrm{SPX}.$$

- The best forecast of the VIX return when observing an S&P return of 2% is therefore $0.0026 - 6.6516 \cdot 0.02 = -0.130432 = -13.0432\%$.

OLS regression: Validation (R^2)

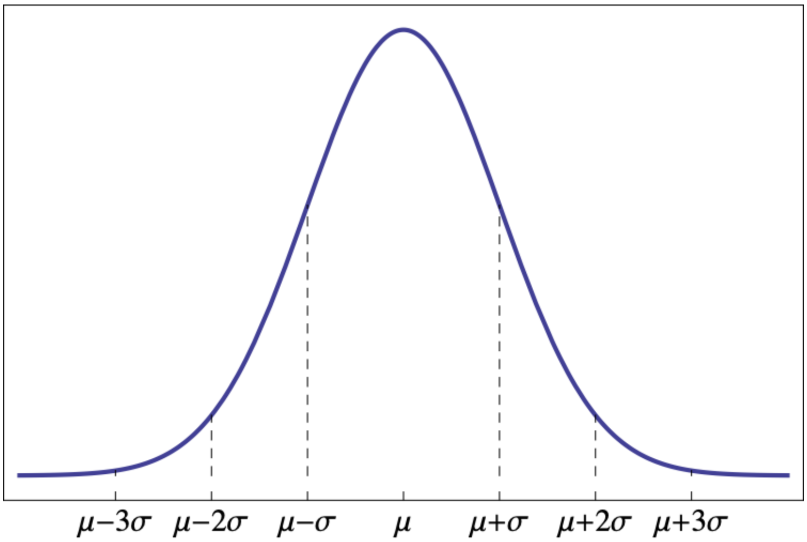
- To **validate** the model, i.e., to determine, if the model in itself and the explanatory variable(s) make sense, we look R^2 and various p-values (or confidence intervals or t-statistics).
- R^2 measures the fraction of variance in the dependent variable Y that is captured by the regression line; $1 - R^2$ is the fraction of Y-variance that remaines in the residuals $\varepsilon_i^2, i = 1, \ldots, n$.
- In the output above R^2 is given as 0.647. In other words, 64.7% of the variance in VIX returns are "explained" by SPX returns.
- A high R^2 (and this one is high) is necessary for making forecasts.

OLS regression: Validation (confidence interval)

- An important hypothesis to test in any regression model is whether the explanatory variable(s) have an effect on the independent variable.
- This can be translated into testing whether $\beta \neq 0$. ($\beta = 0$ is the same as saying that the X variable can be removed from the model.)
- Formally, we test the null hypothesis $H_0 : \beta = 0$ against the alternative hypothesis $H_1 : \beta \neq 0$.
- There are several statistics to come to the same conclusion: confidence intervals, t-statistics and p-values.
- The **confidence interval** is an interval around the estimate $\hat{\beta}$ that we are confident contains the true parameter β. A typial **confidence level** is 95%.
- If the 95% confidence interval does **not** contain 0, then we say β is **statistically significant** at the 5% (=1-95%) level, and we conclude that $\beta \neq 0$.

OLS regression: Validation (t-statistic)

- The t-statistic corresponds to the **number of standard deviations** that the estimated coefficient $\hat{\beta}$ is away from 0 (the mean under H_0).
- For a normal distribution, we have the following rules of thumb:
  - 66% of observations lie within one standard deviation of the mean
  - 95% of observations lie within two standard deviations of the mean
  - 99.7% of observations lie within three standard deviations of the mean



- If the sample size is large enough, then the t-statistic is approximately normally distributed, and if it is large (in absolute terms), then this is an indication against $\beta = 0$.
- In the example above, the t-statistics is -62.559, i.e., $\hat{\beta}$ is approx. 63 standard deviations away from zero, which is practically impossible.

OLS regression: Validation (p-value)

- The p-value expresses the probability of observing a coefficient estimate as extreme (away from zero) as $\hat{\beta}$ under H_0, i.e., when $\beta = 0$.
- In other words, it measures the probability of observing a t-statistic as extreme as the one observed if $\beta = 0$.
- If the p-value (column `P>|t|`) is smaller than the desired level of significance (typically 5%), then the H_0 can be rejected and we conclude that $\beta \neq 0$.
- In the example above, the p-value is given as 0.000, i.e., it is so small, that we can conclude the estimated coefficient $\hat{\beta}$ is so extreme (= away from zero) that is virtually impossible to obtain such an estimated if $\beta = 0$.
- Finally, the F-test tests the hypotheses $H_0 : R^2 = 0$ versus $H_1 : R^2 \neq 0$. In a multiple regression with k independent variables, this is equivalent to $H_0 : \beta_1 = \cdots = \beta_k = 0$.
- In the example above, the p-value of the F-test is 0, so we conclude that the model overall has explanatory power.