Introduction to quantitative finance: Stylized facts of asset returns

Eryk Lewinson
Nov 5, 2018 · 7 min read ★

As a data scientist with a background in quantitative finance, I have always been interested in exploring the possibilities resulting from the combination of two fields. I think this is a fascinating area to explore and that is why I would like to start a series of articles describing the basics of quantitative finance.

By the end of the series, I intend to present a simple allocation strategy and hopefully show that by using data science/quantitative finance knowledge it is possible to outperform basic benchmark strategies. Of course, no one is talking about building a

There is a theory explaining why this is not possible, namely the efficient market hypothesis (EMH). It states that asset prices fully reflect all available information. This implies that it is simply impossible to consistently beat the market, due to the fact that market prices only react to new information. You can read more about it here. But we can still play around and see if — in the short term — it is possible to make some money, at least in theory :)

Before jumping straight into machine learning and building asset allocation strategies, I think it is crucial to spend some time on the basics and to understand the processes we try to model. In this article, I look into the stylized fact of asset returns and show how to verify their existence using Python. Some elementary knowledge of statistics would be helpful, but don't be discouraged as I try to intuitively explain what is going on.

Returns and why are we working with them

I begin by downloading historical stock prices using Python's `quandl` library. It is pretty straightforward and you only need to create a free account to obtain the API key.

In this article, I work with adjusted close prices, because they account for possible corporate actions such as stock splits, etc.

I choose Microsoft (ticker: MSFT) as an example and download the time series in the form of a `pandas` Data Frame. Then, I transform the prices into logarithmic returns for further analysis:

$$r_t = log(P_t \,/\, P_{t-1}) = log(P_t) - log(P_{t-1})$$

where P_t denotes the price of an asset at time *t* and log function stands for the natural logarithm (sometimes referred to as *ln*).

```
1   import quandl
2
3   # authentication ----
4   quandl_key = 'key' # paste your own API key here :)
```

　　　　　　　　　　　　　　　　　×

```
 8    df.columns = ['adj_close']

 9

10    # create simple and log returns, multiplied by 100 for convenience
11    df['simple_rtn'] = 100 * df.adj_close.pct_change()
12    df['log_rtn'] = 100 * (np.log(df.adj_close) - np.log(df.adj_close.shift(1)))

13

14    # dropping NA's in the first row
15    df.dropna(how = 'any', inplace = True)

16

17    df.head()
```
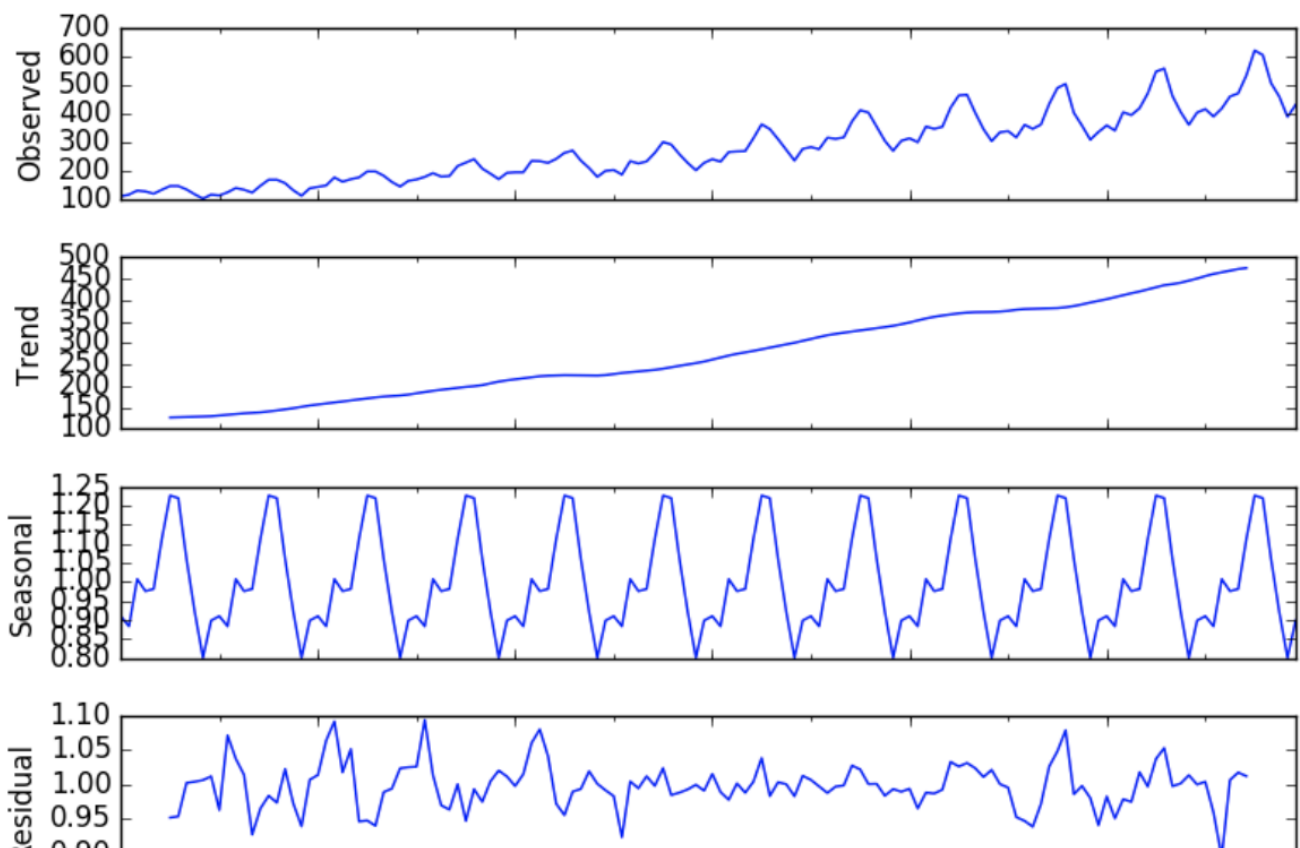
quandl_data.py hosted with ❤ by **GitHub**   **view raw**

There is a great article describing the difference between simple and log returns, so you can find out the pros and cons of using both of them there.

But why work with returns when we already have the prices? The reason is that prices are usually non-stationary, that is when statistics such as mean and variance (mathematical moments) change over time. This could also mean observing some trends or seasonality in our price series. As you can imagine, by working with returns we make the time series stationary, which is a desired property in statistical modeling. For now, let's leave it like this.
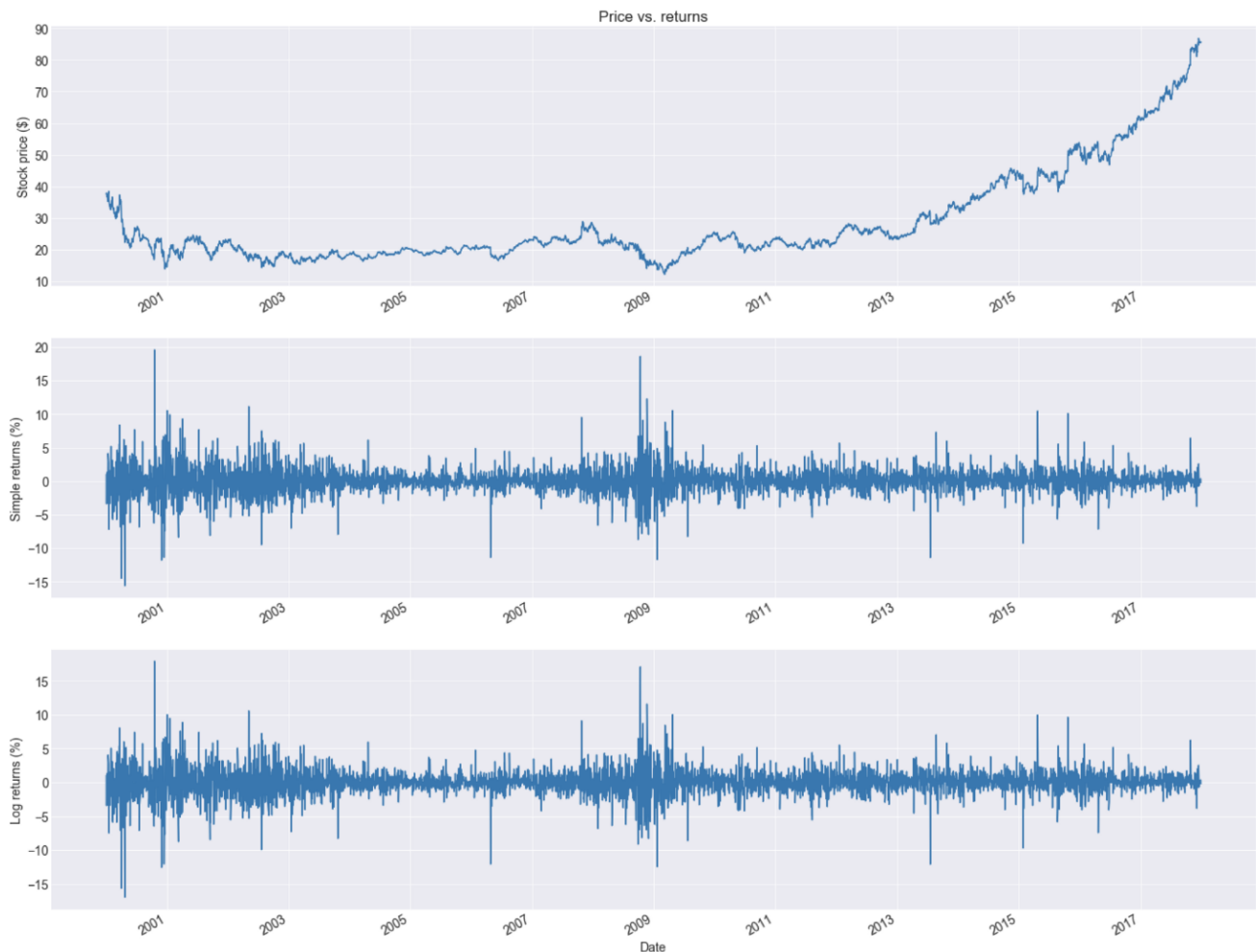
Decomposition of a time series into trend/seasonal components.

Below I present the evolution of the Microsoft prices and returns over time.

One important fact that is directly observable at the plots is the existence of 'volatility clustering' — periods of large returns alternate with periods with small returns, suggesting that volatility is not constant. I will get back to it later.
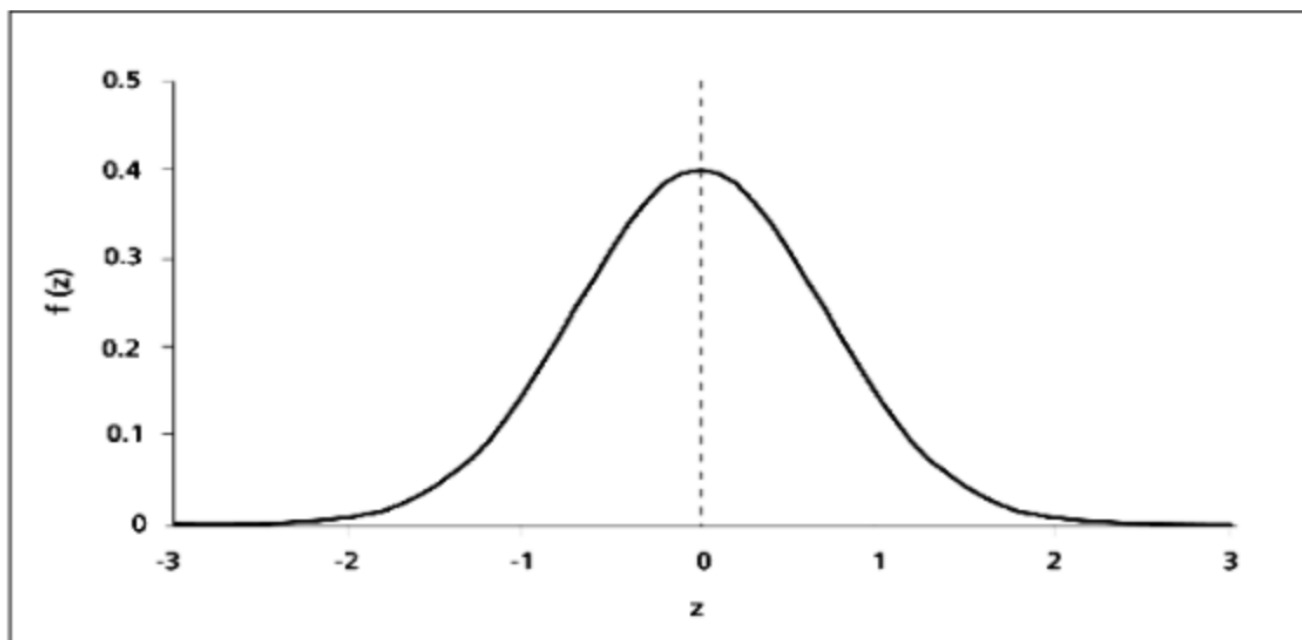
. . .

Stylized facts are, generally speaking, statistical properties that appear to be present in many empirical asset returns (across time and markets). It is important to be aware of them because when building models that are supposed to represent asset price dynamics, the models must be able to capture/replicate these properties.

Standard Normal (Gaussian) distribution

It has been observed that returns exhibit:

- negative skewness (third moment) — large negative returns occur more often than large positive ones. Visually: The left tail is longer; the mass of the distribution is concentrated on the right side of the distribution plot.

- excess kurtosis (fourth moment) — large (and small) returns occur more often than expected. Visually: fat-tailed and peaked distribution.

Below I show a histogram visualizing the distribution of Microsoft's log returns, together with a line representing the Normal probability density curve (with mean and standard deviation equal to sample means). We see that the returns do exhibit a higher peak and also more mass is located in the tails (than one would expect under normality).

For further inspection, I also look at the Q-Q Plot. The red line represents the Standard Normal distribution. In the case when the returns followed Gaussian distribution, those two lines would be aligned. However, we see that there are differences, mostly in the tails. This further verifies the above-mentioned findings.

For more details on how to create and interpret Q-Q plots, you can read this article.
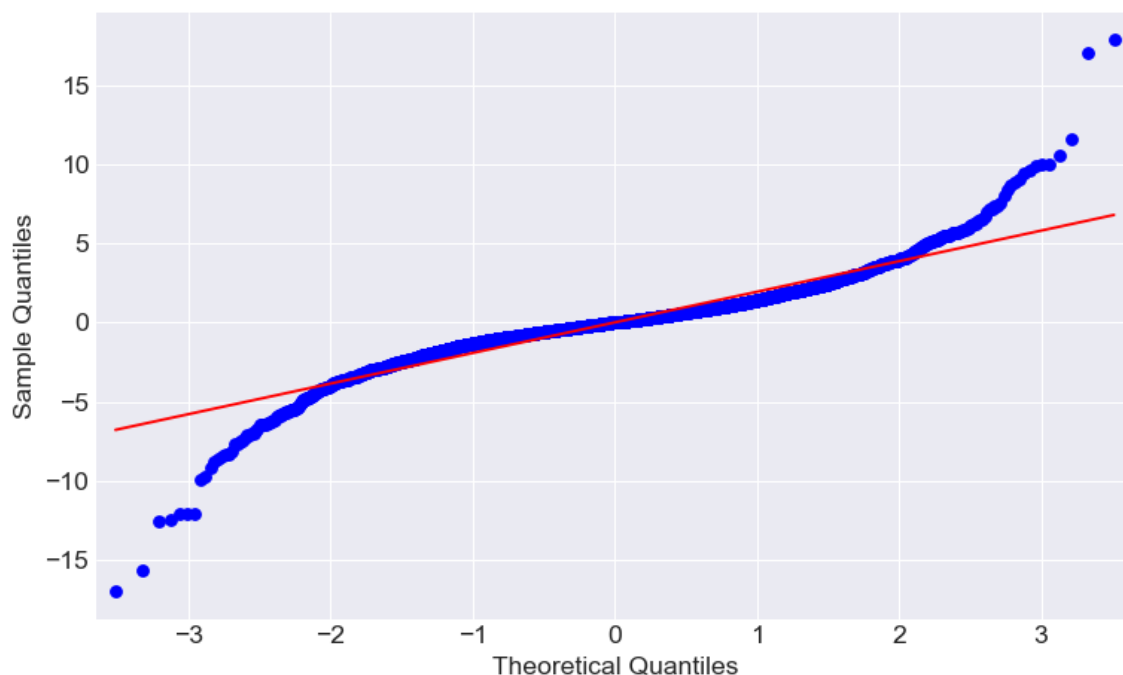
Lastly, I look at the descriptive statistics of the considered returns (they are presented as daily values, in practice, it is common to present them annualized). The Jarque-Bera Normality Test confirms our suspicions, with p-value small enough to reject the null hypothesis stating that the data follows Gaussian distribution.

```
Mean: 0.0175
Median: 0.0000
Min: -16.9683
Max: 17.8773
Standard Deviation: 1.9341
Skewness: -0.1239
Kurtosis: 9.9657
Jarque-Bera statistic: 18694.53 with p-value: 0.00
```

Descriptive statistics of MSFT's log returns

Fact 2: No (or almost no) significant autocorrelation in returns

Autocorrelation measures the degree of similarity between a given time series and the lagged version of the same series over successive time intervals. It is analogous to the correlation between two time-series: the first one in its original form and one lagged by n periods.

Example: When returns of a certain asset exhibit historically positive autocorrelation and in the past few days the price was increasing, one might reasonably expect further positive movements (of course predicting stock prices is not as simple as that, otherwise becoming a millionaire would be quite an easy task… EMH strikes again).

Fact 3: Small and slowly decreasing autocorrelation in squared and absolute returns

When modeling returns, taking into account volatility can be of paramount importance in the decision making (buy/sell) process. Volatility is commonly understood as the standard deviation (the squared root of variance) of the returns.

For now, instead of returns let's consider errors, i.e., actual values — values forecasted/explained by some model. Variance is basically the average of squared errors, while absolute deviation is the average of absolute errors. By plotting the squared/absolute errors over time we could see whether the variance (or absolute deviation which is also a measure of volatility) is constant over time. In case of asset returns that is not the case and we can observe periods of high/low volatility. That is called 'volatility clustering' and can be observed at the time series plot of log returns.

On a different note, the average of short-term (daily) returns in the long term is expected to be zero (EMH). That is why by looking at squared and absolute returns we are effectively measuring deviation from the expected mean, without looking at the direction of the error — both square and absolute functions cancel out the direction of the error.

Below I present the Autocorrelation plots for MSFT's log returns, together with squared and absolute values (inspecting both fact 2 and 3). The blue area indicates the 95% confidence interval, points outside of it are statistically significant. We see that for log returns, there are only a few significant points (which is in line with fact 2). As for fact 3, we see that the correlations are significant and their decline is easier to observe for squared returns than in case of absolute returns. Summing up, this leads us to believe that we can try to leverage the autocorrelation structure to carry out volatility modeling.
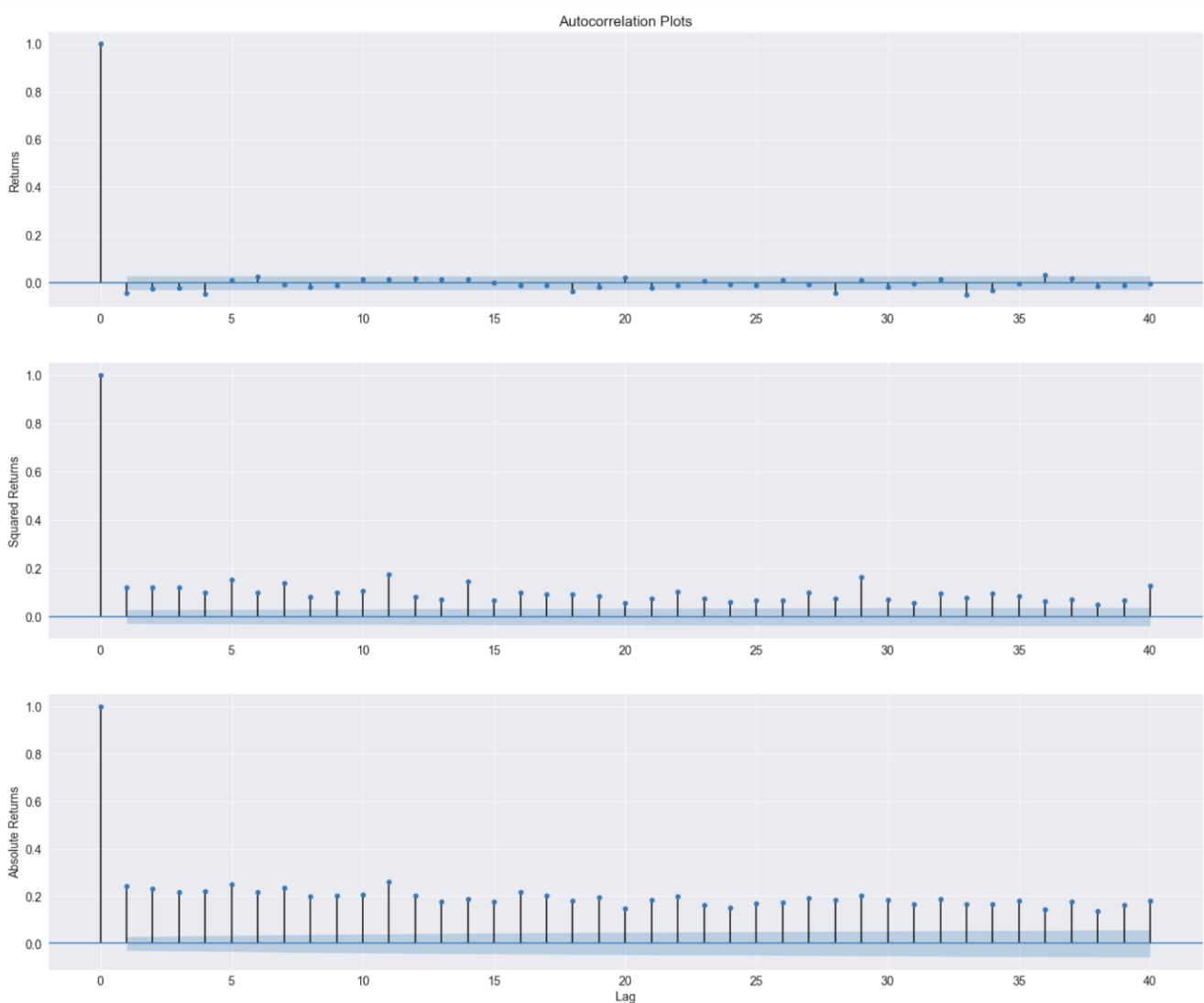
Conclusions

In this article, I presented a brief introduction to the stylized facts of asset returns. Of course, this is barely scratching the surface of the topic. There is much more to be learned and different resources also add further stylized facts. In case you are interested, I add some extra resources in the references.

One important thing to note is that there is no guarantee that all the facts will be clearly observable in each and every returns series. This can be easily influenced by the time horizon taken into account in the analysis. In this case, I take data from 2000 onwards. But there is no strict rule for that. With the provided code you can easily investigate different time horizons and assets.

Having done the introduction, in the next article I will present some common methods of predicting asset returns (and maybe their volatility as well). In the concluding article, I will try to show that with (more or less accurate) forecasts we can try to build asset allocation strategies and investigate if historically we would have made some money!

As always, any constructive feedback is welcome. You can reach out to me on Twitter or in the comments. The code used for investigating the stylized facts can be found on my GitHub.

· · ·

I recently published a book on using Python for solving practical tasks in the financial domain. If you are interested, I posted an article introducing the contents of the book. You can get the book on Amazon or Packt's website.

· · ·

References:

[1] https://orfe.princeton.edu/~jqfan/fan/FinEcon/chap1.pdf

[2] Empirical properties of asset returns: stylized facts and statistical issues —

Get the Medium app