01-05 Incomplete Data

Compiled by Shipra De, Fall 2016

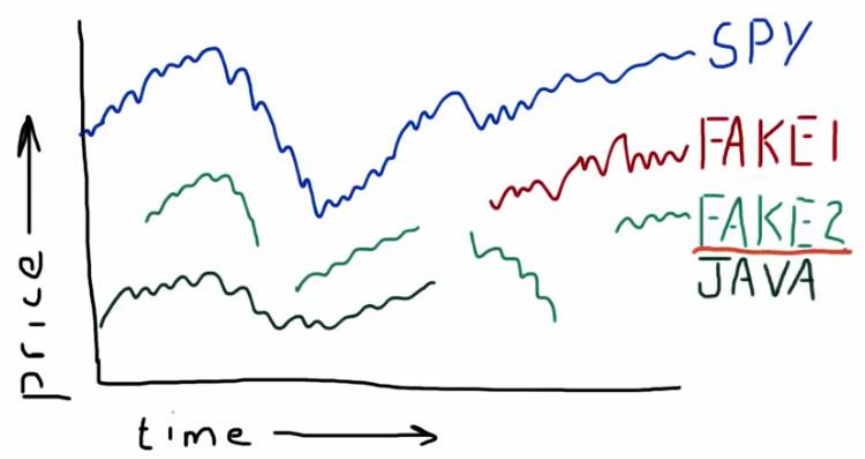
Introduction

- Historical financial data is of course essential for effective financial research.
- One reason people are attracted to finance as an area of research, is because they believe the data is very well documented. In other words, all the data is monitored and recorded by computer and saved for us to pore over later.
- It turns out though that there are many ways the data can be faulty. In this lesson, we're going to look at how missing data can occur and what we can do about it.

Pristine Data

- Now, as you might have guessed, the data isn't really pristine.
- So, here's what people think financial data is like. They imagine that it's perfectly recorded minute by minute. The prices that are recorded are exactly right. The volume data is exactly right. But that's not the case.
- But that's not the only mistake [LAUGH] people make about what they think the data's like. People assume there's no gaps in the data, that we have every single minute recorded. That data for stocks started since the beginning of time, and they continue to the very last minute.
- The reality is that our data is an amalgamation created from many, many sources. For instance, for any particular stock, it may be traded on the New York Stock Exchange, at NASDAQ, at Bats, and over any particular minute during the day, it may trade at one price at New York Stock Exchange, another price at NASDAQ.
- The reality is that there's no single price for any stock at any particular time. In fact, it's hard to say who's right. So, the reality of the data that we get is that it's a combination from all these different sources. And different data providers will provide, actually, different numbers.
- And finally, one part of reality that's especially troublesome is not all stocks trade every day. Sometimes stocks come into existence and suddenly, there's values for them and before that there was no data. Sometimes stocks go out of existence and suddenly, they quit existing and there's no data for them going forward. There's another kind of failure mode or missing data mode where a particular stock will be trading, data will be missing, suddenly, it starts trading again.
- And these are the sorts of problems we're going to take a look at and find a way to solve in this lesson.

Why Data Goes Missing

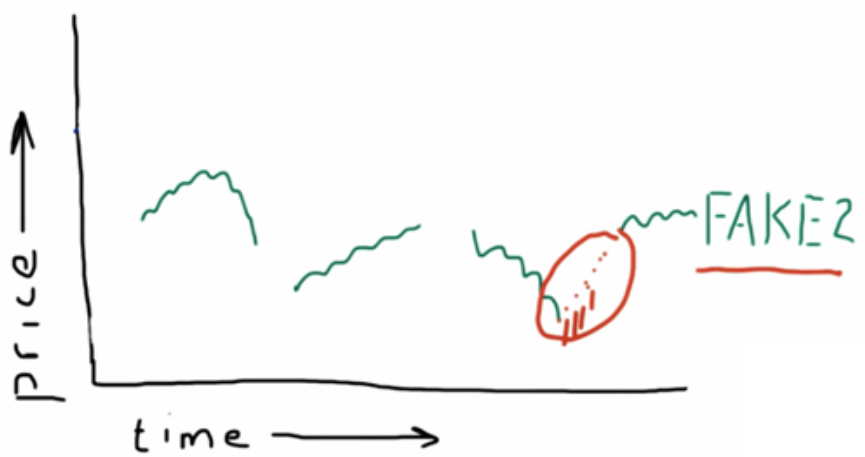


- Okay, let's take a look at some examples so we can see how prices are recorded over time.
- We'll start with SPY. This is showing a time series of that ETF over time. This is the downturn in 2008, 2009. SPY represents the S&P 500. It's one of the most liquid and actively traded ETFs out there, and we typically use it as a reference, a time and date reference for other stocks.
- Because we know if SPY was trading, the stock market was open and we can use its time history as a reference in that regard. It goes all the way back to 1993. There are of course some stocks that go back further, all the way to 1901 and so on. But most of what we're going to do, it's fine that we know it's been active since 1993.
- Now Let's look at a couple more examples. We'll add JAVA, J-A-V-A, and as you can see, it was trading from the beginning here but for some reason or another, abruptly stopped.
- Now what happened there? Well, you may remember that Sun Microsystems, which was trading under the ticker JAVA, was acquired by Oracle in 2010. And on that date, that ticker went away. So if you look at historical data for JAVA, you'll see that it ends at sometime in 2010. Something else that's interesting about this ticker JAVA, is that before it was Sun Microsystems it was actually Mr. Coffee. So if you look historically for data for JAVA you'll find two different time series. One for when it traded as Mr. Coffee and another when it traded as Sun Microsystems, but it doesn't exist any longer.
- So imagine if you're processing this time series data, and you arrive at this abrupt end for JAVA, what's going to happen? Well in the data you'll see NAN, meaning not a number, meaning there's no data there. And the focus of this lesson is what to do about that.
- Let's take a look at another example. Okay, we've added now an additional set of data, and as you can see we named it FAKE meaning this example we invented for the purpose of this discussion. Now, each of these symbols is available to you in the data that we provide you for the class. So you also will have this FAKE1.
- Now, the data represented by FAKE1 is fairly common, and we only invented it just so it would work out well in this chart that we're looking at. Anyways, what's going on with FAKE1 is, as you can see, it didn't exist before this time. So instead of having, for instance, NAN values after

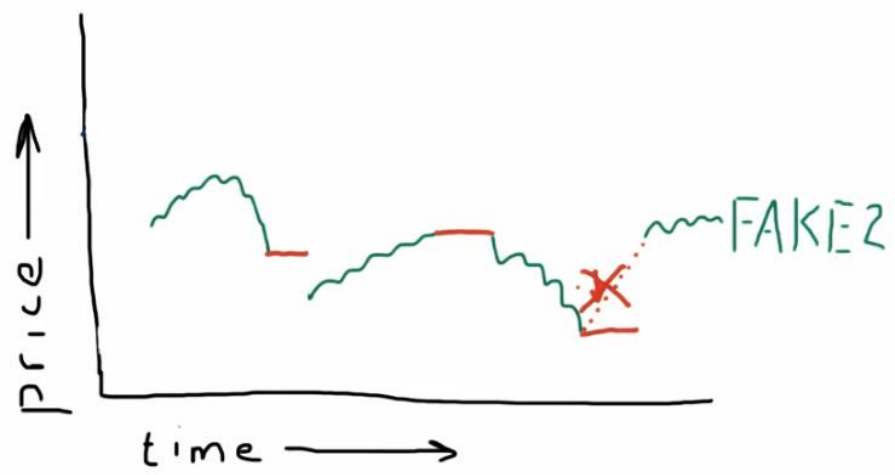
a certain date, this FAKE1 data is going to have NAN values before a certain date. So we'll have a different kind of problem trying to process that data.

- Now we'll look at one more example, and as you might have guessed, we named that FAKE2. Now what's special about this one, is it's got all of the different kinds of problems at once. So it didn't exist before this date, data was absent in between these two dates, and so on. This is not typical data for a very liquid, very large stock, for instance like Google or Apple, but indeed data like this exists for thinly traded stocks.
- In other words, companies that don't have a high market capitalization, and they trade very little if at all occasionally. So we still have to be able to deal with data like this in our studies, and so let's focus on this FAKE2 example.

Why This Is Bad—What Can We Do?

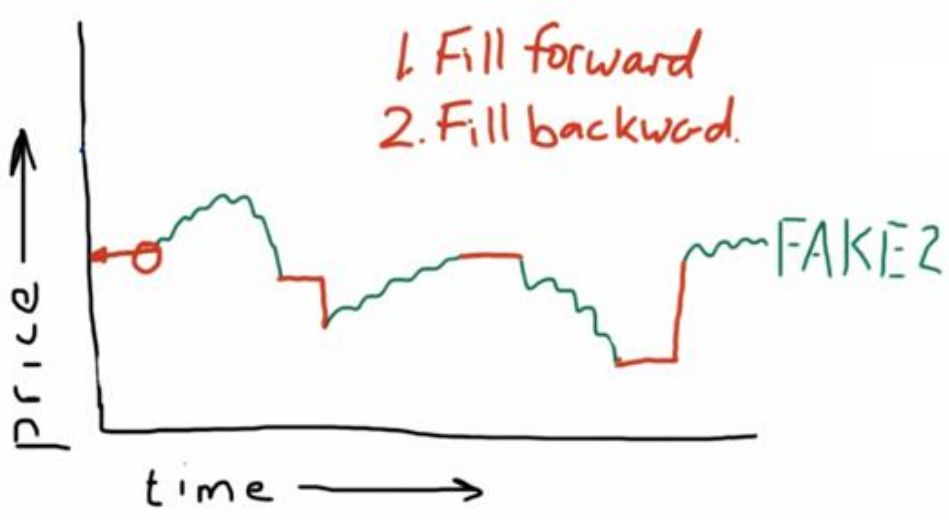


- The question is, what do we do in situations like this where we don't have data between two separate dates? Now, you might think, Gee, let's interpolate. And so we would estimate what a line is between those two dates and then fill in at each point an interpolated value. Why not do that?
- Well, the truth of the matter is, between these two dates, there was no trading. There really was no price for that data. But if we're going to do something like, for instance, compute a rolling average. Or a mean over that data, and there's nans there, that'll wipe out our entire calculation.
- So we can't leave it empty, but we shouldn't interpolate it either. So here's what we might do.



- One thing that we can do, is we fill forward, going from, we go over all the data, and when there's some missing data, we fill forward from the last, previous known value.
- So for instance, if we were to do that here, we would get these values up until that date, where it takes over there. We would fill forward here, and fill forward here.

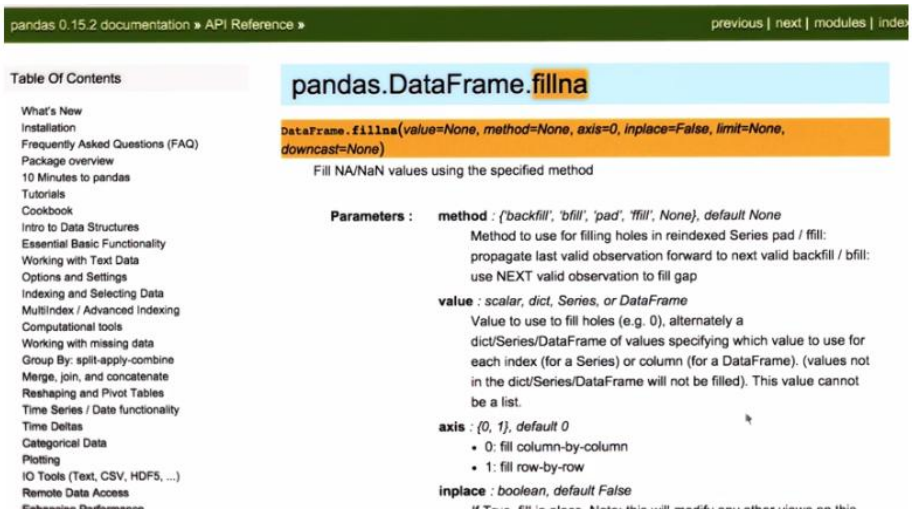
- Now notice there's a big gap between there, and there's a big gap here, but that realistically reflects what was going on with the data.
- Now the reason we do this instead of the interpolation is the following. Let's suppose we were looking for patterns in the data and we had rolled back time and we were simulating history. And let's suppose for a moment we had this interpolation. And let's suppose we're right here in time and we're trying to figure out what's going to happen next. We're looking for patterns and so on.
- We're actually giving ourselves information about the future. We're observing that the price is going up. So if we were to make a calculation here, we would actually be peeking into the future. And that is not allowed. We do not want to do that.
- So we need to stick with only filling forward a last known price. If we do that then we're not peeking into the future.



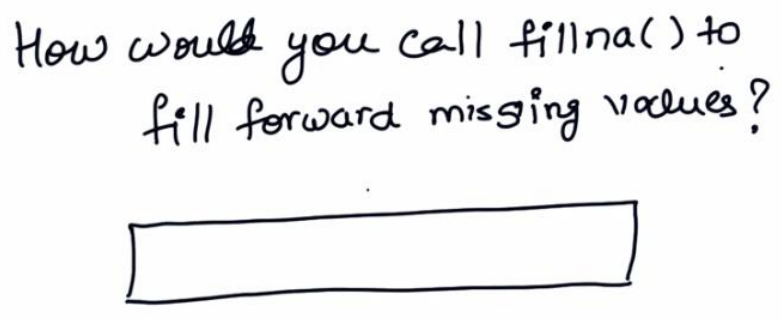
- So let's get rid of that ugly, nasty peeking into the future. Okay, so now we've actually filled in all our gaps and we have continuous data from this start point all the way to the end.
- However, there still is missing data here at the beginning. And because we need some value here in order to calculate rolling averages or whatever sort of statistics we want to do, it's better to have some value here instead of not a number. And in this case, we fill backwards.
- So remember, if you are going to fill your data to resolve problems with gaps, fill forward first and fill backward second. That way you will avoid, to the max extent possible, peeking into the future.
- Now we'll hand it over to Dave. And she's going to show you how to do this in code. Here's to you, Dave!

Pandas Fillna()

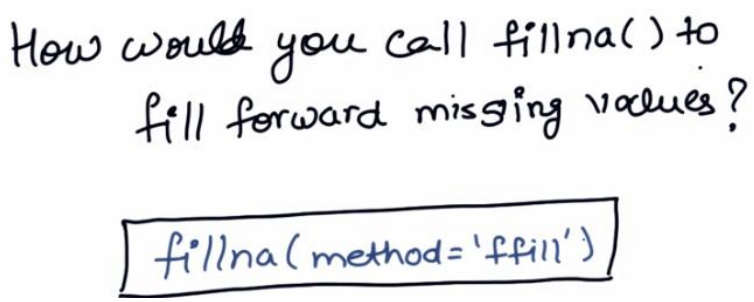
- Thanks, Professor, I'll take it from here. So hello, everyone, and we will be using Pandas fillna function to fill the missing data. So let's find the documentation of this function, and let's go to this site.
- So here it is. The link is included in the instructor notes. I encourage you to bookmark this site. So it gives the usage for any function at a glance and it will help you a lot. So let's scroll down and let's search for a function using this small search box. Let's search for fillna.



- Here it is, DataFrame.fillna function, and we'll be using this. Read and try to understand different options and how to call this fillna function, and I'll be right back with a pop quiz.



- So here's a question for you. How would you call fillna() to fill forward missing values? Go ahead and type your answer in this box.



- So here's the answer. In order to fill forward missing values we need to specify the method ffill. Note that the method value is a string, so it has to be enclosed in quotes.

Using Fillna()

```
import numpy as np
import matplotlib.pyplot as plt
import os
import  pandas as pd


#---------function to get path of the symbol--------------
def symbol_to_path(symbol, base_dir="data"):
    """Return CSV file path given ticker symbol."""
    return os.path.join(base_dir, "{}.csv".format(str(symbol)))


#-----------------Reads csv--------------------------------
def get_data(symbollist,dates):
    df_final=pd.DataFrame(index=dates)
    if "SPY" not in symbollist:
        symbollist.insert(0,"SPY")
    for symbol in symbollist:
        file_path=symbol_to_path(symbol)
        df_temp=pd.read_csv(file_path,parse_dates=True,index_col="Date",usecols=["Da
```

- So, let's do some coding. To start with, let's use an example stock with missing values. We will be using fakedo.csv and this file is included in your data folder.

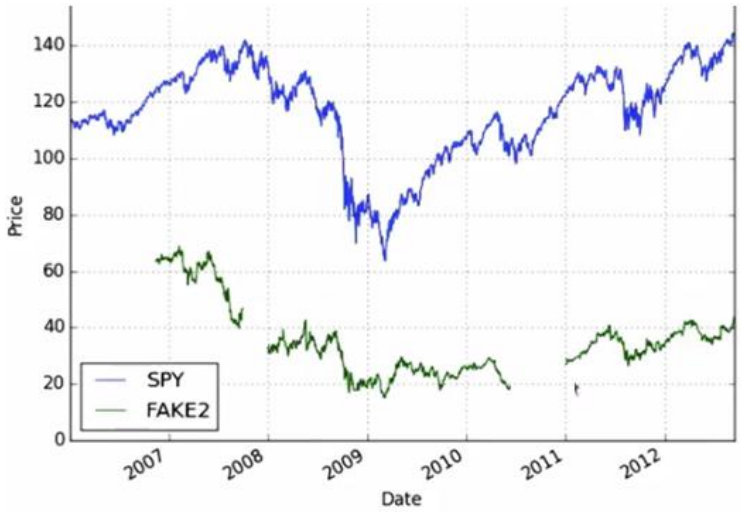
```
def get_data(symbollist,dates):
    df_final=pd.DataFrame(index=dates)
    if "SPY" not in symbollist:
        symbollist.insert(0,"SPY")
    for symbol in symbollist:
        file_path=symbol_to_path(symbol)
        df_temp=pd.read_csv(file_path,parse_dates=True,index_col="Date",usecols=["Da
        df_temp=df_temp.rename(columns={'Adj Close':symbol})
        df_final=df_final.join(df_temp)
        if symbol =="SPY":
            df_final=df_final.dropna(subset=['SPY'])
    return df_final

#---------------------plot function---------------------------
def plot(df_data):
    ax=df_data.plot(title="Incomplete Data",fontsize=2)
    ax.set_xlabel("Date")
    ax.set_ylabel("Price")
    plt.show()
```

- So as usual, we will be reading the csv into the data frame and we will do some plotting.

```
28 def plot(df_data):
29     ax=df_data.plot(title="Incomplete Data",fontsize=2)
30     ax.set_xlabel("Date")
31     ax.set_ylabel("Price")
32     plt.show()
33
34
35
36 if __name__ == '__main__':
37 #list of symbols
38  #symbollist=["PSX","FAKE1","FAKE2"]
39  symbollist=["FAKE2"]
40  #date range
41  start_date='2005-12-31'
42  end_date='2014-12-07'
43 #create date range
44  idx=pd.date_range(start_date,end_date)
45 #get adjusted close of each symbol
46  df_data=get_data(symbollist,idx)
47  plot(df_data)
```

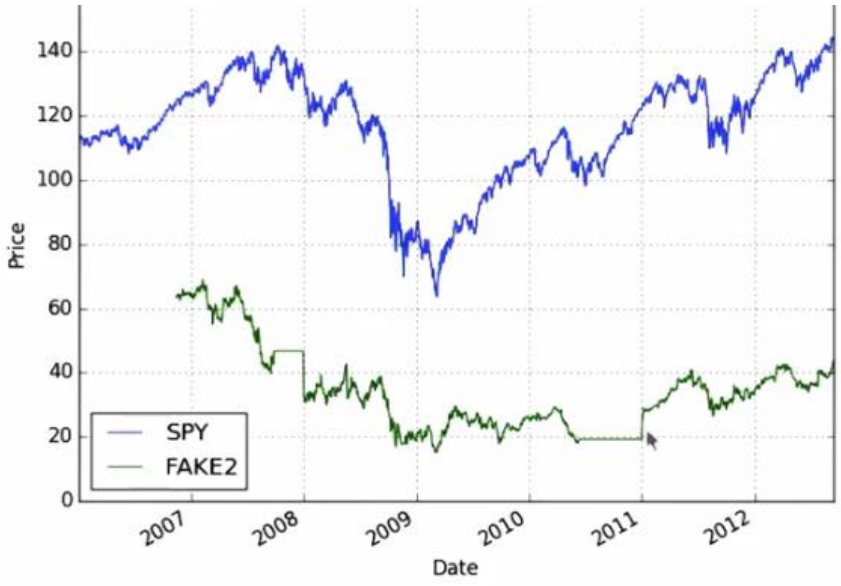
- So now let's go and plot this data and see what turns up.



- So, here is the graph. For the given range of dates, you can notice that there is a gap in the beginning and also a gap at multiple places in the middle. So now, let's try to fix this.

```
29      ax=df_data.plot(title="Incomplete Data",fontsize=2)
30      ax.set_xlabel("Date")
31      ax.set_ylabel("Price")
32      plt.show()
33
34
35
36  if __name__ == '__main__':
37  #list of symbols
38   #symbollist=["PSX","FAKE1","FAKE2"]
39   symbollist=["FAKE2"]
40   #date range
41   start_date='2005-12-31'
42   end_date='2014-12-07'
43  #create date range
44   idx=pd.date_range(start_date,end_date)
45  #get adjusted close of each symbol
46   df_data=get_data(symbollist,idx)
47   df_data.fillna(method="ffill",inplace="TRUE")
48   plot(df_data)
```

- We only need to add a single statement to fill those gaps. As you must have read in the documentation, method ffill corresponds to forward filling and inplace is equal to TRUE will save all the changes in the same data frame.
- Try removing this and see what happens. Now, let's plot and see how the graph looks now.



- So here's the graph. If you look closely, you will observe the forward filling effect. The stock prices retained their previous values throughout.
- However, note that the missing values at the beginning of the range have not been filled. Think about what you need to do to take care of that.

Fill Missing Values

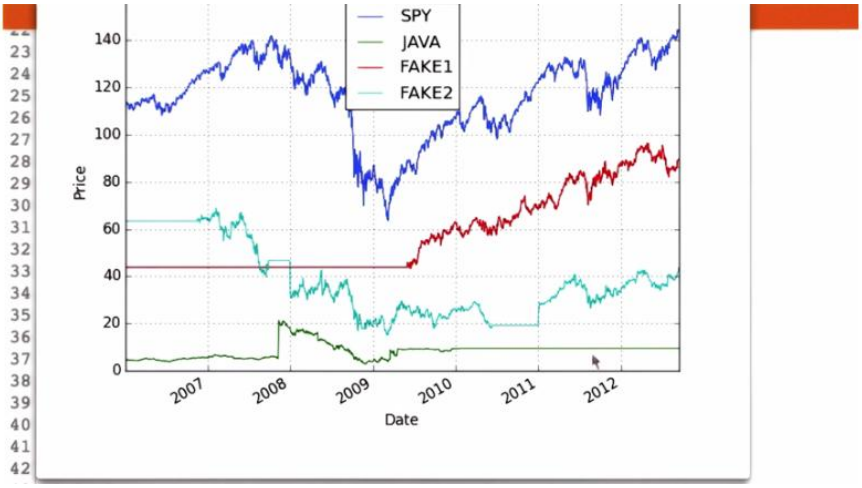
```
29      ax=df_data.plot(title="Incomplete Data",fontsize=2)
30      ax.set_xlabel("Date")
31      ax.set_ylabel("Price")
32      plt.show()
33
34
35
36  if __name__ == '__main__':
37  #list of symbols
38   symbollist=["JAVA","FAKE1","FAKE2"]
39  #date range
40   start_date='2005-12-31'
41   end_date='2014-12-07'
42   #create date range
43   idx=pd.date_range(start_date,end_date)
44  #get adjusted close of each symbol
45   df_data=get_data(symbollist,idx)
46   #to-do: fill gaps using
47  '''Forward and Backward fill code'''
48   plot(df_data)
49
```

- Time for some coding quiz. Okay. Here's some code to read in our data for multiple symbols during a specified period of time.

- If you go ahead and plot this, you see where we have data missing for JAVA, FAKE1, and FAKE2. Your task is to fill these gaps using the fillna method and yes, it can work for multiple stocks or in that case, multiple columns of the data frame simultaneously.
- You can refer to the documentation and the previous example if you need a clue. Good luck.

```
35
36 if __name__ == '__main__':
37 #list of symbols
38  symbollist=["JAVA","FAKE1","FAKE2"]
39 #date range
40  start_date='2005-12-31'
41  end_date='2014-12-07'
42  #create date range
43  idx=pd.date_range(start_date,end_date)
44 #get adjusted close of each symbol
45  df_data=get_data(symbollist,idx)
46  #to-do: fill gaps using
47  '''Forward and Backward fill code'''
48  df_data.fillna(method="ffill",inplace="TRUE")
49  df_data.fillna(method="bfill",inplace="TRUE")
50  #-------------------------------------
51  plot(df_data)
52
```

- So, here's the solution. To solve this, you need to use both forward and backward fill. The key is to use forward fill first, and then the backward fill, to avoid peeping into the future as much as possible. Now let's see how the graph looks now.



- Here is the resulting graph. Know the effects of the forward filling and the backward filling in different segments of the missing data. That's all for now. Happy coding till I get back to you.