# Data and Analytics Specialist – Written Assessment

Note: all code in Jupyter notebooks for ease of review along with supporting files located <https://github.com/donsohn/BOC>

**Question A – total 10 points avalable**

Using the programming language of your choice, perform the following tasks. Submit your code, plots, and anything else that we might need to rerun your code. Feel free to use any software or libraries.

1. Download the Bitcoin price/volume/market cap data from this website: [https://coinmarketcap.com/currencies/bitcoin/historical-data/](https://can01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fcoinmarketcap.com%2Fcurrencies%2Fbitcoin%2Fhistorical-data%2F&data=05%7C01%7CSGomezdeSilvaSosa%40bank-banque-canada.ca%7C0fa3f11675974e5aaf7708db3f606333%7C164f988ba2f44584aeaa21bd4a0234bc%7C0%7C0%7C638173455425426075%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=aMFtVlmbHI%2FgTJU0%2B1R%2FY%2F%2FlhrUhAnEHbXXGRXxhin4%3D&reserved=0) and reproduce the chart of Bitcoin to USD as shown in this link: <https://coinmarketcap.com/currencies/bitcoin/>. Be sure to include all three panels (Price, Market Cap, and Trading View); interactivity (hover, date expansion) is optional (extra 1 bonus point).

**Using Python:**

import pandas as pd

import os

import sys

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.dates as mpl\_dates

from mpl\_finance import candlestick\_ohlc

df\_bitcoin = pd.read\_csv(os.path.join(sys.path[0], 'bitcoindata.csv'))

df\_bitcoin.head()

# price chart

df\_bitcoin.plot(kind='line', x='Date', y='Close', title='Price')

# market cap chart

df\_bitcoin.plot(kind='line', x='Date', y='Market Cap', title='Market Cap')

# trading view

ohlc = df\_bitcoin.loc[:, ['Date', 'Open', 'High', 'Low', 'Close']]

ohlc['Date'] = pd.to\_datetime(ohlc['Date'])

ohlc['Date'] = ohlc['Date'].apply(mpl\_dates.date2num)

ohlc = ohlc.astype(float)

fig, ax = plt.subplots()

candlestick\_ohlc(ax, ohlc.values, width=0.6, colorup='green', colordown='red', alpha=0.8)

ax.set\_xlabel('Date')

ax.set\_ylabel('Price')

fig.suptitle('Trading View')

date\_format = mpl\_dates.DateFormatter('%d-%m-%Y')

ax.xaxis.set\_major\_formatter(date\_format)

fig.autofmt\_xdate()

fig.tight\_layout()

plt.show()

**Results (screenshot taken from output of execution in Jupyter notebook):**

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Answer the following questions:

1. If you needed to publish this chart on a dashboard, which tool would you choose and why?

Power BI or Tableau. Besides improved aesthetics, it would be easier to maintain including ability to have dynamic updates to the dashboard when underlying data is refreshed/updated, integrate with scheduled batch runs, version control, and facilitating sharing and publishing of results to consumers. Essentially it would be a data asset with a pipeline that can be better operationally supported.

1. If you needed to update this chart on a daily basis, how would you automate the process?

Although external sources of data could be web scraped using Python libraries such as Selenium or Beautiful Soup, a better option would be to leverage the data source’s API if made available. The challenge in relying on web scraping is the dependency on the web site itself not changing its format.

If this chart is required on a daily basis, then we would want to measure the service level, have redundancy or backup, and contingencies in the absence of source data. Leveraging an enterprise scheduler such as Tidal or JAMS would tie in any upstream data dependencies should there be delays. There would also be notifications and a runbook to ensure that exceptions and issues are handled within reasonable time.

1. If the data were provided to you as part of a big dataset stored on the cloud (Azure) of one million records (1 terabyte), would your choice for question iii change? Why?

Yes, data would be stored on the cloud and could leverage cloud native tools such as orchestrated workflow amongst the compute storage, spin up dynamic container instances, cheaper compute options such as Azure functions or AWS lambdas, ETL/ELT with options such as Azure Data Factory, and alerts and handling of failures in the pipeline. The compute could be scaled up/down dynamically as needed with redundancy options, especially if there is a SLA that needs to be met.

1. Using the same dataset and at least one additional data source of your choice, build any two visualizations using your choice of variables and complement them with notes and descriptions to explain why you have chosen these visualizations.

Explored the correlation between Bitcoin and S&P500 index which tracks the performance of stocks for the top 500 companies. If correlated, this would imply that Bitcoin has become neither 'digital gold' nor a 'safe-haven asset' in times of crisis.

A limitation with this analysis is being able to retrieve only the most recent month’s data for the S&P500 based on the source used, whereas more historical data was more easily available for Bitcoin.

**Python code:**

import pandas as pd

import os

import sys

import numpy as np

import matplotlib.pyplot as plt

df\_bitcoinapr = pd.read\_csv(os.path.join(sys.path[0], 'bitcoindataapril.csv'))

df\_sp = pd.read\_csv(os.path.join(sys.path[0], 'sp500.csv'))

#plt.plot(df\_bitcoin.Date, df.bitcoin.Close)

f = plt.figure()

f.set\_figwidth(30)

f.set\_figheight(10)

plt.plot(df\_bitcoinapr.Date, df\_bitcoinapr.SPClose)

plt.plot(df\_bitcoinapr.Date, df\_bitcoinapr.Close)

plt.show()

**Result:**

Given the limitation with amount of data for S&P 500 it was inconclusive to show correlation with the price of bitcoin. Ideally would have observed two plotted lines synchronous at same points in time.

Graphical user interface, application

Description automatically generated

Another area for exploration was to perform timeseries analysis using Facebook’s Prophet (<https://facebook.github.io/prophet/>) to predict the next 60 day price of bitcoin. The hypothesis is that similar to stocks, that there might be aspect of seasonality that the price could be predicted within a certain range.

import pandas as pd

from fbprophet import Prophet

import matplotlib.pyplot as plt

datap = df\_bitcoin.reset\_index()

# Select only the important features i.e. the date and price

datap = datap[["Date","Close"]] # select Date and Price

# Rename the features: These names are NEEDED for the model fitting

datap = datap.rename(columns = {"Date":"ds","Close":"y"}) #renaming the columns of the dataset

m = Prophet(daily\_seasonality = True) # the Prophet class (model)

m.fit(datap) # fit the model using all data

future = m.make\_future\_dataframe(periods=60) #we need to specify the number of days in future

prediction = m.predict(future)

m.plot(prediction)

plt.title("Prediction of Bitcoin using the Prophet")

plt.xlabel("Date")

plt.ylabel("Price")

plt.show()

**Result:**

The visualization shows a steady in the price for bitcoin but there is a large range of variance per the shaded light blue area in the chart.

**Chart, scatter chart

Description automatically generated**

**Question B – total 5 points avalable**

1. Clean the dataset found in the csv file, including interpolating missing datapoints. Please include a short explanation (maximum 50 words) of how you cleaned the data set.

Considered strategy of using mean, median, or most frequent to impute missing datapoints. If there were other variables, would also have considered linear regression model to predict the missing data. After graphing for a sample of 3 years decided to use the mean between the months before and after the missing datapoints. When exploring the data using histogram and scatterplot, found outliers:

1. Date 1/1/2005 rate of 53 – could remove but decided based on other values of that year that most likely typo and should be 5.3
2. Date 1/2/2018 rate of 41 – similar to previous changed to 4.1

It was also observed that April to July 2020 had high values consistently in the teens but likely not an error. A quick google search confirmed that the summer of 2020 had high unemployment rate due to COVID-19 pandemic.

**Python code:**

df\_unrate = pd.read\_csv(os.path.join(sys.path[0], 'unrate.csv'))

df\_unrate.head()

# plot 3 individual years to get feel for the linear

f = plt.figure()

f.set\_figwidth(40)

f.set\_figheight(10)

plt.plot(df\_unrate.iloc[:12].DATE, df\_unrate.iloc[:12].UNRATE)

plt.show()

f = plt.figure()

f.set\_figwidth(40)

f.set\_figheight(10)

plt.plot(df\_unrate.iloc[24:36].DATE, df\_unrate.iloc[24:36].UNRATE)

plt.show()

f = plt.figure()

f.set\_figwidth(40)

f.set\_figheight(10)

plt.plot(df\_unrate.iloc[36:48].DATE, df\_unrate.iloc[36:48].UNRATE)

plt.show()

f = plt.figure()

f.set\_figwidth(40)

f.set\_figheight(10)

plt.plot(df\_unrate.iloc[48:64].DATE, df\_unrate.iloc[48:64].UNRATE)

plt.show()

#histogram to get idea of distrbution

plt.hist(df\_unrate.UNRATE, bins=12)

plt.show()

#scatterplot to identify outliers

df\_unrate.plot(kind='scatter', x='DATE', y='UNRATE', title='scatter')

#load imputed values for unrate

df\_unrate\_rev = pd.read\_csv(os.path.join(sys.path[0], 'unrate\_rev.csv'))

**Result (select graphs):**

**Chart, line chart

Description automatically generated**

Graphical user interface, application

Description automatically generated

1. Create one table of the following summary statistics for the unemployment rate of the US: Mean, Standard Deviation, 25th Percentile, 50th Percentile, 75th Percentile, and Max. Display the results using the following format

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Summary statistics for the US unemployment rate | | | | | | |
| Mean | Standard Deviation | Min | 25th Percentile | 50th Percentile | 75th Percentile | Max |
| 5.988803 | 1.952039 | 3.5 | 4.6 | 5.5 | 7.05 | 14.7 |

Note that these results are calculated after I had cleaned the datapoints in prior step.

Using Python I loaded the data as dataframe and ran describe function:

import pandas as pd

import os

import sys

import numpy as np

df\_unrate = pd.read\_csv(os.path.join(sys.path[0], 'unrate.csv'))

np.percentile(df\_unrate\_rev.UNRATE, q=[0, 25, 50, 75, 100])

df\_unrate\_rev.describe()

**Results:**

Table

Description automatically generated