SIT220/731 2024.T3: Task 2P

Working with `numpy` Vectors (Unidimensional Data)

Last updated: 3rd November 2024

Contents

1 Introduction

This task is related to Module 1 and 2; see the *Learning Resources* on the unit site.

This task is due on **Week 3 (Friday)**. Start tackling it as early as possible. If we find your first solution incomplete or otherwise incorrect, you will still be able to amend it based on the generous feedback we will give you. In case of any problems/questions, do hot hesitate to attend our on-campus/online classes or use the Discussion Board on the unit site.

Submitting after the aforementioned due date will incur a late penalty. This task is part of the **hurdle requirements** in this unit. Not submitting the correct version on time results in failing the unit.

All submissions will be checked for plagiarism. You are expected to work independently on your task solutions. Never share/show parts of solutions with/to anyone.

2 Question 1 - 4

Create a single Jupyter/IPython notebook (see the *Artefacts* section below for all the requirements – read the whole task specification first!), where you perform what follows.

> The use of **pandas** is forbidden. You can use **scipy**, though.
>
> Do not use `for` loops or list comprehensions – this is an exercise on **numpy**.

Q1. Download the daily close BTC-to-USD data, from 2023-01-01 up to 2023-12-31, available at https://finance.yahoo.com/quote/BTC-USD (the *Historical Data* tab).

Q2. Use `numpy.genfromtxt` or `numpy.loadtxt` to read the above BTC-to-USD data as a **numpy** vector named `rates`.

Option: You can use a spreadsheet application such as LibreOffice Calc or MS Excel to *manually* remove everything except the numeric values in the *Close* column. The column labels should also be *manually* deleted. Export these observations to a CSV file (which should only contain numbers, one per line). You can also use features in `numpy.genfromtxt` or `numpy.loadtxt` to remove them.

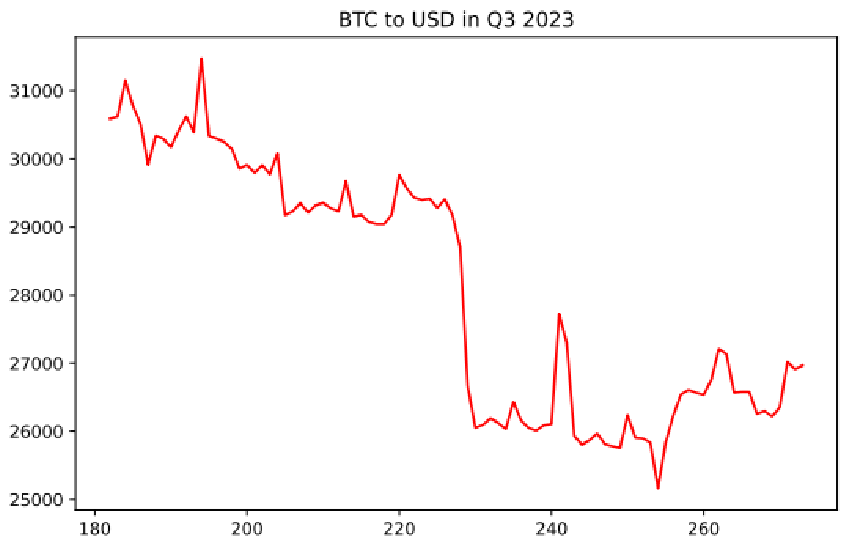
Q3. For the third quarter of the year *only* (Q3 2023; days 182–273 inclusive), determine and display (in a readable manner) the following aggregates:

- arithmetic mean,
- minimum,
- the first quartile,
- median,
- the third quartile,
- maximum,
- standard deviation,
- interquartile range.

Reference result from Q3 2024 (yours can be prettier):

```
##                       arithmetic mean:  28091.33

##                               minimum:  25162.65

##                                    Q1:  26225.56

##                                median:  28871.82

##                                    Q3:  29767.07

##                               maximum:  31476.05

##                    standard deviation:   1827.04

##                                   IQR:   3541.51
```

4. Call `matplotlib.pyplot.plot(days, rates, <...further_arguments...>)` to draw the Q3 2023 data (with 182 denoting 1 July), using *red solid line segments*. Call `matplotlib.pyplot.title` to add the plot title. Discuss what you see.



5. Determine the day numbers (with 182 denoting 1 July) with the lowest and highest observed prices in Q3 2023.

```
## Lowest  price was on day 254 (25162.65).
```

```
## Highest price was on day 194 (31476.05).
```

All packages must be imported and data must be loaded at the beginning of the file (only once!).

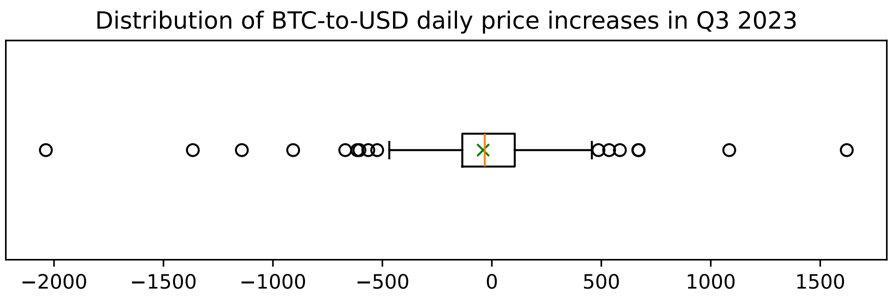
3 Question 5 - 6 for Postgraduate (SIT731)

Postgraduate students are **additionally required** to solve/address/discuss what follows.

Q5. Using `matplotlib.pyplot.boxplot`, draw a *horizontal* box-and-whisker plot for the Q3 2023 daily price increases/decreases as obtained by a call to `numpy.diff`.

Using an additional call to `matplotlib.pyplot.plot`, mark the arithmetic mean on the box plot with a green "x".

In your own words, explain what we can read from the plot. Below is a reference plot from Q3 2024.



Q6. Count (programmatically, using the vectorised relational operators from **`numpy`**) how many outliers the boxplot contains (for the definition of an outlier, consult Section 5.1 in the Book). In your own words, explain what such outliers might mean in the current context.

```
## There are 16 outliers.
```

4 Artefacts

The solution to the task must be included in a single Jupyter/IPython notebook (an .ipynb file) running against a Python 3 kernel. The use of G**gle Colab is discouraged. Nothing beats a locally-installed version where you have full control over the environment. Do not become dependent on third-party middlemen/distributors. Choose freedom instead.

Make sure that your notebook has a **readable structure**; in particular, that it is divided into sections. Use rich Markdown formatting (text in dedicated Markdown chunks – not just Python comments).

Do not include the questions/tasks from the task specification. Your notebook should read nicely and smoothly – like a report from data analysis that you designed yourself. Make the flow read natural (e.g., *First, let us load the data on... Then, let us determine... etc.*). Imagine it is a piece of work that you would like to show to your manager or clients — you certainly want to make a good impression. Check your spelling and grammar. Also, use formal language.

At the start of the notebook, you need to provide: the **title** of the report (e.g., *Task 42: How Much I Love This Unit*), your **name**, **student number**, **email address**, and whether you are an **undergraduate (SIT220) or postgraduate (SIT731)** student.

Then, add 1–2 introductory paragraphs (an introduction/abstract – what the task is about).

Before each nontrivial code chunk, briefly **explain** what its purpose is. After each code chunk, **summarise and discuss the obtained results** (in a few sentences).

Conclude the report with 1–2 paragraphs (summary/discussion/possible extensions of the analysis etc.).

Limitations of the OnTrack ipynb-to-pdf renderer:

Ensure that your report as seen in OnTrack is aesthetic (see *Download submision PDF* after uploading the .ipynb file). The OnTrack ipynb-to-pdf renderer is imperfect. We work with what we have. Here are the most common Markdown-related errors.

- Do not include any externally loaded images (via the `![label](href)` Markdown command), for they lead to upload errors.
- Do not input HTML code in Markdown.
- Make sure you leave one blank line before and after each paragraph and bullet list. Do not use backslashes at the end of the line.
- Currently, also $LaTeX formulae$ and Markdown tables are not recognised. However, they do not lead to any errors.

Checklist:

1. Header, introduction, conclusion (Markdown chunks).
2. Text divided into sections, all major code chunks commented and discussed in your own words (Markdown chunks).
3. Every subtask addressed/solved. In particular, all reference results that are part of the task specification have been reproduced (plots, computed aggregates, etc.).
4. The report is readable and neat. In particular:
   - all code lines are visible in their entirety (they are not too long),
   - code chunks use consecutive numbering (select *Kernel - Restart and Run All* from the Jupyter menu),
   - rich Markdown formatting is used (`# Section Title`, `* bullet list`, `1. enumerated list`, `| table |`, `*italic*`, etc.),
   - the printing of unnecessary/intermediate objects is minimised (focus on reporting the results specifically requested in the task specification).

Submissions which do not *fully* (100%) conform to the task specification *on* the cut-off date will be marked as FAIL.

Good luck!

5 Intended Learning Outcomes

ULO	Is Related?
ULO1 (Data Processing/Wrangling)	YES
ULO2 (Data Discovery/Extraction)	YES
ULO3 (Requirement Analysis/Data Sources)	YES
ULO4 (Exploratory Data Analysis)	YES
ULO5 (Data Privacy and Ethics)	