**Exploring the stock data and building a portfolio by inference**

*Material for the class of International Financial Markets, prepared by dr Krzysztof Waśniewski*

# Introduction

This document presents a method of building an investment portfolio by inference from empirical data. I use an algorithm in the programming language called Python. For those of you who are familiar with it, those tips can be useful. Those, who don’t know and don’t want to learn Python, can just skip the algorithmic parts: all the essentials are provided in this document or in Excel spreadsheets which come as additional material.

# The market and the source empirical data

The market which we will be making our pick in is the Warsaw Stock Exchange, further designated as WSE (<https://www.gpw.pl/en-home> ). I focus on one type of security, in equities (or shares in equity).

On the WSE site, you can select a subsite, that of price archive: <https://www.gpw.pl/price-archive>.

Once you get there, select your instrument as ‘**Equities**’, and select a date. You get a download in Excel titled as: ‘\_YYYY-MM-DD\_akcje.xls’, where ‘akcje’ stands for stock AKA equities, e.g. ‘\_2024-04-22\_akcje.xls’ for April 22nd, 2024. Each such Excel spreadsheet is structured into columns, which are specified below with their English translation:

* PL: Data ⬄ ENG: Day
* PL: Nazwa ⬄ ENG: Name of the company
* PL: ISIN ⬄ ENG: Numerical identifier of the company
* PL: Waluta ⬄ ENG: Currency which the given equity is denominated in
* PL: Kurs otwarcia ⬄ ENG: Opening price of the day
* PL: Kurs max ⬄ ENG: Maximum price of the day
* PL: Kurs min ⬄ ENG: Minimum price of the day
* PL: Kurs zamknięcia ⬄ ENG: Closing price of the day
* PL: Zmiana ⬄ ENG: Change in price [%]
* PL: Wolumen ⬄ ENG: Volume of transactions in the day (number of shares traded)
* PL: Liczba Transakcji ⬄ ENG: Number of distinct transactions in the day
* PL: Obrót ⬄ ENG: Value of transactions in the day [Volume times average price]
* PL: Liczba otwartych pozycji ⬄ ENG: Number of open positions, i.e. the number of investors who hold that equity at the end of this specific day
* PL: Wartość otwartych pozycji ⬄ ENG: Value of open positions
* PL: Cena nominalna ⬄ ENG: Nominal price of the equity

In the analyses that follow, the compiled data covers trading sessions in the Warsaw Stock Exchange from November 27th, 2023, through April 24th, 2024.

## Compilation of the source data in Python

Compilation of data has been done with a code in Python (this is the part which the non-interested can just skip). I start by importing the libraries I will need:

>> import numpy as np

>> import pandas as pd

>> import os

>> import math

Then I set the directory where I want to store the source data downloaded from <https://www.gpw.pl/price-archive> and where I want to store transformations and analyses of that data:

>> os.chdir(‘Full path to the directory’)

In the next step I start loading the Excel files into my compiler and transforming them into Pandas DataFrames, as in the example below. The date column in the original file is a string of text and therefore I add a new column, named “Day”, in a numerical form:

>> Akcje\_2024\_04\_16=pd.DataFrame(pd.read\_excel('\_2024-04-16\_akcje.xls'))

>> Akcje\_2024\_04\_16.insert(0,"Day",20240416)

I repeat the same operation for all the Excel sheets from <https://www.gpw.pl/price-archive> which I want to include in my analyses. In other words, I create a separate DataFrame for each Excel file. That done, I concatenate those daily DataFrames into one DataFrame:

>> **GPW\_Stock** = pd.concat( [ Akcje\_2023\_12\_29, Akcje\_2023\_12\_08, Akcje\_2023\_11\_27, Akcje\_2023\_12\_11, Akcje\_2023\_12\_05, Akcje\_2023\_12\_22, Akcje\_2023\_12\_28, Akcje\_2023\_12\_04, Akcje\_2024\_01\_02, Akcje\_2023\_11\_28, Akcje\_2023\_12\_13, Akcje\_2023\_12\_07, Akcje\_2023\_12\_18, Akcje\_2023\_12\_20, Akcje\_2023\_12\_15, Akcje\_2023\_12\_01, Akcje\_2023\_11\_29, Akcje\_2023\_12\_06, Akcje\_2023\_12\_12, Akcje\_2023\_11\_30, Akcje\_2023\_12\_27, Akcje\_2023\_12\_19, Akcje\_2023\_12\_21, Akcje\_2023\_12\_14, Akcje\_2024\_01\_02, Akcje\_2024\_01\_03, Akcje\_2024\_01\_04, Akcje\_2024\_01\_05, Akcje\_2024\_01\_08,Akcje\_2024\_01\_09, Akcje\_2024\_01\_10, Akcje\_2024\_01\_11, Akcje\_2024\_01\_12, Akcje\_2024\_01\_15, Akcje\_2024\_01\_16, Akcje\_2024\_01\_17, Akcje\_2024\_01\_18, Akcje\_2024\_01\_19, Akcje\_2024\_01\_22, Akcje\_2024\_01\_23, Akcje\_2024\_01\_24, Akcje\_2024\_01\_25, Akcje\_2024\_01\_26, Akcje\_2024\_01\_29, Akcje\_2024\_01\_30, Akcje\_2024\_01\_31, Akcje\_2024\_02\_01, Akcje\_2024\_02\_02, Akcje\_2024\_02\_05, Akcje\_2024\_02\_06, Akcje\_2024\_02\_07, Akcje\_2024\_02\_08, Akcje\_2024\_02\_09, Akcje\_2024\_02\_12, Akcje\_2024\_02\_13, Akcje\_2024\_02\_14, Akcje\_2024\_02\_15, Akcje\_2024\_02\_16, Akcje\_2024\_02\_19, Akcje\_2024\_02\_20, Akcje\_2024\_02\_21, Akcje\_2024\_02\_22, Akcje\_2024\_02\_23, Akcje\_2024\_02\_26, Akcje\_2024\_02\_27, Akcje\_2024\_02\_28, Akcje\_2024\_02\_29, Akcje\_2024\_03\_01, Akcje\_2024\_03\_04, Akcje\_2024\_03\_05, Akcje\_2024\_03\_06, Akcje\_2024\_03\_07, Akcje\_2024\_03\_08, Akcje\_2024\_03\_11, Akcje\_2024\_03\_12, Akcje\_2024\_03\_13, Akcje\_2024\_03\_14, Akcje\_2024\_03\_15, Akcje\_2024\_03\_18, Akcje\_2024\_03\_19, Akcje\_2024\_03\_20, Akcje\_2024\_03\_21, Akcje\_2024\_03\_22, Akcje\_2024\_03\_25, Akcje\_2024\_03\_26, Akcje\_2024\_03\_27, Akcje\_2024\_03\_28, Akcje\_2024\_04\_02, Akcje\_2024\_04\_03, Akcje\_2024\_04\_04, Akcje\_2024\_04\_05, Akcje\_2024\_04\_08, Akcje\_2024\_04\_09, Akcje\_2024\_04\_10, Akcje\_2024\_04\_11, Akcje\_2024\_04\_12, Akcje\_2024\_04\_15, Akcje\_2024\_04\_16, Akcje\_2024\_04\_17, Akcje\_2024\_04\_18, Akcje\_2024\_04\_19, Akcje\_2024\_04\_22 ] )

I check the structure of the concatenated DataFrame:

>> GPW\_Stock.info()

… and I get the following feedback:

<class 'pandas.core.frame.DataFrame'>

Int64Index: 42788 entries, 0 to 410

Data columns (total 16 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Day 37044 non-null int64

1 Data 37044 non-null object

2 Nazwa 37044 non-null object

3 ISIN 37044 non-null object

4 Waluta 37044 non-null object

5 Kurs otwarcia 37044 non-null float64

6 Kurs max 37044 non-null float64

7 Kurs min 37044 non-null float64

8 Kurs zamknięcia 37044 non-null float64

9 Zmiana 37044 non-null float64

10 Wolumen 37044 non-null int64

11 Liczba Transakcji 37044 non-null int64

12 Obrót 37044 non-null float64

13 Liczba otwartych pozycji 37044 non-null int64

14 Wartość otwartych pozycji 37044 non-null int64

15 Cena nominalna 37044 non-null int64

dtypes: float64(6), int64(6), object(4)

memory usage: 5,5+ MB

Now, I get rid of the columns which I think I will not need in further analyses:

>> GPW\_Stock.drop ( [ 'Data', 'ISIN', 'Waluta', 'Cena nominalna', 'Liczba Transakcji', 'Obrót', 'Liczba otwartych pozycji', 'Wartość otwartych pozycji' ], axis=1, inplace=True)

Remark: I think I don’t need those columns. You can approach the thing differently.

When the “GPW\_Stock” DataFrame is reduced to the data I think I will need further, I rename columns in English, so as to make the resulting analyses presentable to an international audience:

>> GPW\_Stock.rename (columns ={'Nazwa': 'Name'},inplace=True)

>> GPW\_Stock.rename (columns ={'Kurs otwarcia': 'Opening price'},inplace=True)

>> GPW\_Stock.rename (columns ={'Kurs max': 'Maximum price'},inplace=True)

>> GPW\_Stock.rename (columns ={'Kurs min': 'Minimum price'},inplace=True)

>> GPW\_Stock.rename (columns ={'Kurs zamknięcia': 'Closing price'},inplace=True)

>> GPW\_Stock.rename (columns ={'Zmiana': 'Change in price'},inplace=True)

>> GPW\_Stock.rename (columns ={'Wolumen': 'Volume'},inplace=True)

Remark: After each modification of the DataFrame, such as the renaming of columns, it is a good idea to check the result with the command:

>> GPW\_Stock.info()

Now, I calculate the mean values for the columns I left in the DataFrame, and I transfer those means to an Excel spreadsheet:

>> GPW\_Stock.groupby('Name').mean('Change in price').to\_excel('GPW Stock mean values.xlsx')