# Problem Solving by Computer MATH36032
# Coursework 3

For this project, we were given a data file *bananas.csv* (containing average weekly wholesale prices of bananas by country of origin), and aksed to process the file using MATLAB to answer a series of questions. The first few lines of the file are shown in Figure 1. The column *Origin* contains countries or regions of origin of the bananas. The *Date* column contains the date and in the *Price* column we have the cost in pounds sterling for a kilogram of bananas.

| | 1<br>Origin | 2<br>Date | 3<br>Price | 4<br>Units |
|---|---|---|---|---|
| 1 | belize | 2021-01-15 | 0.8600 | '£/kg' |
| 2 | costa_rica | 2021-01-15 | 0.8700 | '£/kg' |
| 3 | dominican... | 2021-01-15 | 0.5500 | '£/kg' |
| 4 | ecuador | 2021-01-15 | 0.8300 | '£/kg' |
| 5 | guatemala | 2021-01-15 | 0.7700 | '£/kg' |
| 6 | acp_banan... | 2021-01-15 | 0.8000 | '£/kg' |
| 7 | dollar_ban... | 2021-01-15 | 0.8500 | '£/kg' |
| 8 | all_bananas | 2021-01-15 | 0.8400 | '£/kg' |

Figure 1: The first few lines of the *bananas.csv* file.

Before I begin, I must explain to the reader that I wrote this code in a MATLAB live file. This choice was a matter of personal preference as one could just as easily write the answer to each question in an individual script file. The difference in this case is that in the live file I can call upon variables created in previous sections easily, without having to write them all out again each time. Importantly though, if one wants to run the code I have written here on their own machine, they need to write it all out and run it in the order it appears here, as each section relies on the ones before it. Now, onto the questions:

First, I needed to produce a list of the distinct entries in the *Origin* column.

To begin with, I imported the data into MATLAB, using the `readtable` function. This creates a table variable of the file *bananas.csv*, which we can now manipulate.

For the next part it's important to understand the `categorical` data type. "`categorical` is a data type that assigns values to a finite set of discrete categories" [1], in this case the distinct entries in the *Origin* column.

I used the `categorical` function on the *Origin* column of the table to convert it into a categorical array. Then I applied the `categories` function to the newly converted column. This function can only be applied to categorical arrays, and outputs the distinct categories found within the specified array. So, in our case, it outputs a list of the distinct entires in the *Origin* column. Here's how it looks in the script, with the output in Figure 2:

```
dat = readtable("bananas.csv");  % imports the table
dat.Origin = categorical(dat.Origin);
Country = categories(dat.Origin)
Country'  % formats the output
```

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 'acp_bananas' | 'all_bananas' | 'belize' | 'brazil' | 'cameroon' | 'colombia' | 'costa_rica' |

| 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|
| 'dollar_bananas' | 'dominican_republic' | 'ecuador' | 'eu_bananas' | 'ghana' | 'guadeloupe' |

| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|
| 'guatemala' | 'honduras' | 'ivory_coast' | 'jamaica' | 'malaysia' | 'martinique' | 'mexico' |

| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|
| 'nicaragua' | 'panama' | 'somalia' | 'st_vincent' | 'surinam' | 'venezuela' | 'windward_isles' |

Figure 2: The distinct entries in the *Origin* column (edited to fit onto the page).

Secondly, I was to find the three countries from which the average price was lowest, and the three from which it was the highest, over the last 5 years of data for each country.

The way I went about finding these countries was by generating a list containing the average for each country over their last 5 years, then picking out the bottom and top 3. To do this, a `for` loop over the elements of the cell array `Country` from the last question seemed like the best idea. I'll now go over how I constructed the loop.

On each iteration of the loop, I extract the sub-table containing only the data for the respective country, and sort this new table by date, with the most recent date at the top. This helps, as I want the last 5 years of data for each country, but the most recent date is different for each. This allows me to find the most recent date with ease - it's now the first entry in the *Date* column of each sub-table.

For the next part of the loop, it's important to understand `datetime` arrays and timetables. "`datetime` arrays represent points in time" [2], for example, `2020-03-26` (the date of the first lockdown in the UK due to COVID-19). I can perform arithmetic on `datetime` values: if `t` is the `datetime` value now, then `t+hours(3)` is the `datetime` value in 3 hours. "`timetable` is a type of table that associates a time with each row. ... In addition [to tables], timetables provide time-specific functions to align, combine, and perform calculations with time-stamped data ... The *row times* of a timetable are `datetime` ... values that label the rows. You can index into a timetable by row time ..." [3].

By converting this table into a timetable with the `table2timetable` function, I can now extract table data from a certain time range with the `timerange` function. I select the sub-table containing data from the last 5 years by choosing the range to be, quite simply, 'most recent date minus 5 years', to 'most recent date'. I've added the function parameter `'closed'` here: by default, the `timerange` function takes values from the half-open interval [Start Date, End Date). By adding this parameter I'm telling the function to take values from the closed interval [Start Date, End Date] (I've done this because there are a few cases in the data file where a country has one most recent data point followed by 5 or more years of no data. In this case the average over the last 5 years is just the price value for that most recent date, so I've made sure to keep it when selecting data from a specific time range).

Finally, I take the average of the price column for this sub-table, and append the value onto the list of averages. I chose the list of averages to be a cell array as, unlike standard arrays, entries can be appended onto the end in this way. This completes the content of the `for` loop.

Now I can input my list of countries, and their averages, into the `table` function which will output them in a table which I can manipulate. I sort the *Average Price* column in ascending order and, finally, output the top and bottom 3 rows of the table. Here's how it looks in the script, and I've shown the output in Figure 3:

```
av = {};

for i = 1:27

    T = dat(dat.Origin == Country{i},:);  % isolates data from each country
    T = sortrows(T,2,'descend');  % sort by date, most recent at the top
    m_r = T{1,'Date'};  % datetime value of most recent date

    T = table2timetable(T);  % switch to timetable in order to select by ...
       timerange
    T = T(timerange(m_r-years(5),m_r,'closed'),:);  % isolates the last 5 ...
       years of data
    av{i} = mean(T{:,'Price'});  % updates list of averages with each ...
       country's average
end

Average_Price = av';  % so dimensions match in table function below
Units = cell(27,1);
Units(:) = {'£/kg'};  % units column
A = table(Country,Average_Price);  % tabulates the data for the averages
A = sortrows(A,2,'descend');  % sorts by average price, ascending

top_3 = A(1:3,:)
% a bit of formatting, the bottom 3 are shown in ascending order
bottom_3 = sortrows(A(25:27,:),2,'ascend')
```

| | Country | Average_Price | Units |
|---|---|---|---|
| 1 | 'somalia' | 0.8300 | '£/kg' |
| 2 | 'panama' | 0.7890 | '£/kg' |
| 3 | 'costa_rica' | 0.7690 | '£/kg' |

(a)

| | Country | Average_Price | Units |
|---|---|---|---|
| 1 | 'windward_isles' | 0.4703 | '£/kg' |
| 2 | 'surinam' | 0.5071 | '£/kg' |
| 3 | 'venezuela' | 0.5227 | '£/kg' |

(b)

Figure 3: (a) The three countries from which the average price was highest, and (b) the three from which it was the lowest, over the last 5 years of data for each country.

Thirdly, I was asked to produce a plot comparing the variation of the prices of bananas with countries of origin: Costa Rica, Windward Isles and Ecuador. Also, I was to include on that plot the data from the country 'All Bananas', which is an average of the collective data for each date.

My approach to creating this plot was to extract the sub-tables containing the data of each of the four countries, then plot the price values on the y-axis against time on the x-axis. The code for this question is very simple, following exactly the steps I have just described, and then adding a few details: line colours, legend and axis labels. I could have generated the sub-tables using some sort of loop but in this case the number of countries I'm comparing is small enough that either method would have taken the same amount of work. For a plot comparing a larger number of countries a loop would probably be a more effective way of generating the plot. Here's how my code looks in the script, with the output shown in Figure 4:

```
T1 = dat(dat.Origin == 'costa_rica',:);  % sub-tables with the data for each ...
    specified country
T2 = dat(dat.Origin == 'windward_isles',:);
T3 = dat(dat.Origin == 'ecuador',:);
T4 = dat(dat.Origin == 'all_bananas',:);

p = plot(T1.Date,T1.Price,'r',...
    T2.Date,T2.Price,...
    T3.Date,T3.Price,'b',...
    T4.Date,T4.Price,'m');

p(2).Color = [0.4660 0.6740 0.1880];  % special line color

lg = legend('Costa Rica','Windward Isles',...
    'Ecuador','All Bananas','Location','northwest');  % legend

xlabel('Date');  % axis labels
ylabel('Price (£/kg)');
```

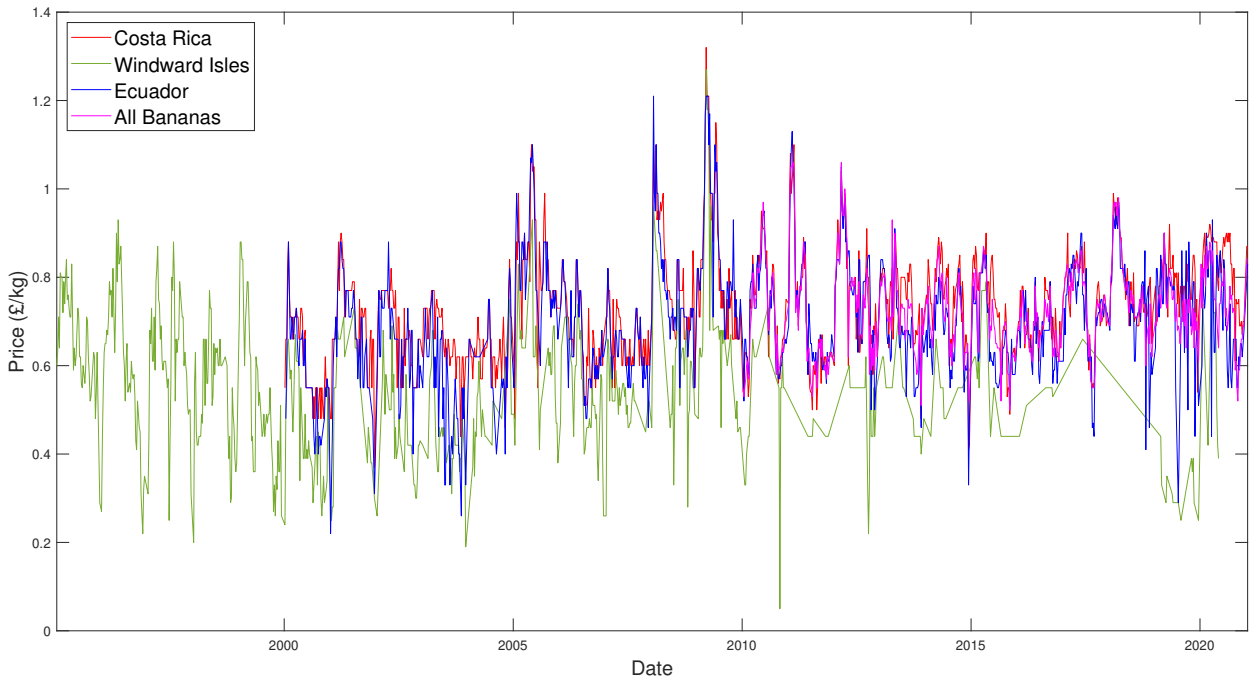


Figure 4: Plot comparing the variation of the prices of bananas with countries of origin.

Finally, I was asked to produce a plot comparing the variation of the prices of bananas with countries of origin as before, over the time period from 2016-01-01 to 2020-12-31, and to comment on any seasonal trends.

Producing this plot was very simple as it's already shown on the previous plot. It was just a case of reducing the x-axis of the previous plot to the specified date range, using the `xlim` function. Here's how that looks in the script, with the output shown in Figure 5:

```
xlim([datetime(2016,1,1), datetime(2020,12,31)])  % same plot, x-axis limits ...
    shortened to the required timespan
lg.Location = 'southwest';  % formatting the legend
```

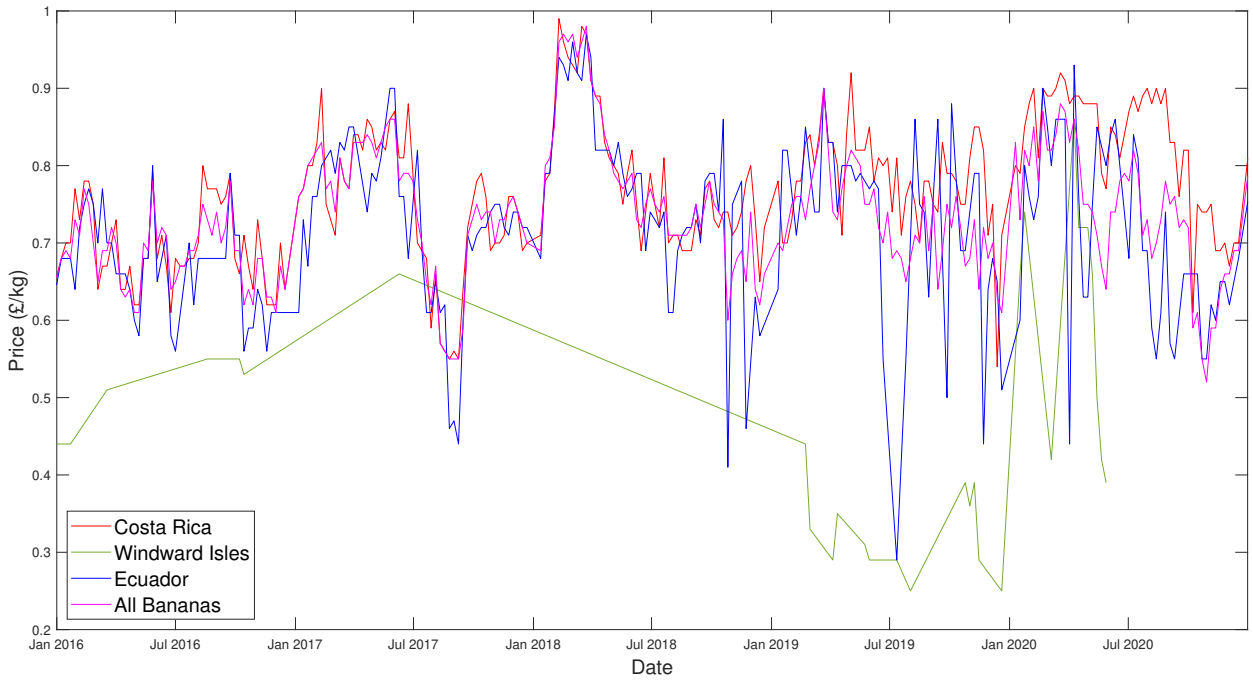


Figure 5: Plot comparing the variation of the prices of bananas with countries of origin over the time period from 2016-01-01 to 2020-12-31.

When looking for seasonal trends, I looked at the above plot. At first glance I thought there might be some trends as the price seems to fluctuate a fair amount. I decided to investigate further. I wanted to know what time of the year bananas grew, as if there is a seasonal trend, it would surely be affected by this factor. However I found that "Unlike other fruit like apples which have a growing season, bananas are available all year round." [4]. This leads me to believe there should be a constant supply of bananas throughout the year, so if there is any seasonal trend, this factor would not contribute to it. Next I decided to check the average retail price of bananas in the UK. If there was any seasonal trend there, it might suggest that there is a seasonal variation in banana supply/cost that I could find in my own data. The Office for National Statistics has a data set, RPI: Ave price - Bananas, per kg [5]. The data over the relevant time frame is shown in Figure 6. There appears to be no variation from month to month whatsoever. Together, this information leads me to conclude that there are no seasonal trends in my data, despite first appearances.

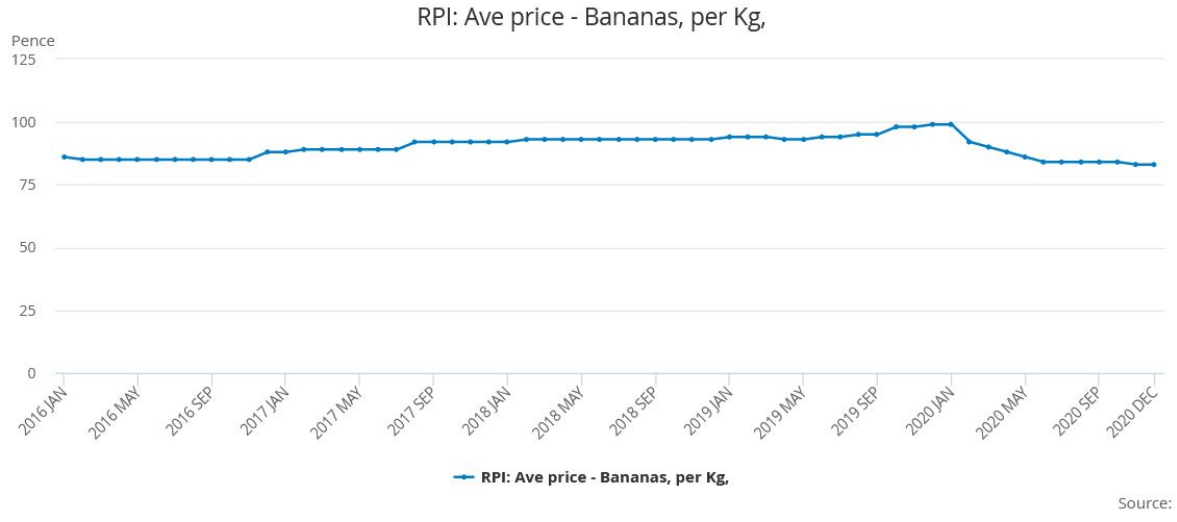


Figure 6: Average retail price/kg bananas, per month, over the last 5 years.

## References

[1] MathWorks (2013). *Array that contains values assigned to categories.* Available at: https://uk.mathworks.com/help/matlab/ref/categorical.html?searchHighlight=categorical&s_tid=srchtitle (Accessed: 12 May 2021).

[2] MathWorks (2014). *Arrays that represent points in time.* Available at: https://uk.mathworks.com/help/matlab/ref/datetime.html?s_tid=doc_ta (Accessed: 12 May 2021).

[3] MathWorks (2016). *Timetable array with time-stamped rows and variables of different types.* Available at: https://uk.mathworks.com/help/matlab/ref/timetable.html?s_tid=doc_ta (Accessed: 12 May 2021).

[4] BananaLink (n.d.). *All About Bananas.* Available at: https://www.bananalink.org.uk/all-about-bananas/ (Accessed: 9 May 2021).

[5] Office for National Statistics (2021). *RPI: Ave price - Bananas, per Kg.* Available at: https://www.ons.gov.uk/economy/inflationandpriceindices/timeseries/czmv/mm23 (Accessed: 9 May 2021).