Methodology, Modelling, and Consulting Skills
School of Mathematics
The University of Edinburgh
Year: 2022/2023

THE UNIVERSITY *of* EDINBURGH

# Computer Session 4: More Xpress techniques

This handout gives you more Xpress techniques to work with.

## 1 Solving quadratic models

In this workshop we are going to solve a classic Markowitz model. Suppose that we have a choice of ten investments, and that each investment has an associated return, denoted by $r_i \forall i \in \{1, \ldots, 10\}$. More information pertaining to the investments can be found in Table 1. Let $x_i$ denote the fraction of capital invested in investment $i \in \{1, \ldots, 10\}$ and suppose that we want to minimise the overall risk of our investments by minimising the objective function

$$\min z = \sum_{i=1}^{10} \sum_{j=1}^{10} Q_{i,j} x_i x_j \tag{1}$$

where $Q_{i,j}$ is the $(i, j)$-th entry of the covariance matrix

$$\boldsymbol{Q} = \begin{pmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 19 & -2 & 4 & 1 & 1 & 1 & 0.5 & 10 & 5 \\ 0 & -2 & 28 & 1 & 2 & 1 & 1 & 0 & -2 & -1 \\ 0 & 4 & 1 & 22 & 0 & 1 & 2 & 0 & 3 & 4 \\ 0 & 1 & 2 & 0 & 4 & -1.5 & -2 & -1 & 1 & 1 \\ 0 & 1 & 1 & 1 & -1.5 & 3.5 & 2 & 0.5 & 1 & 1.5 \\ 0 & 1 & 1 & 2 & -2 & 2 & 5 & 0.5 & 1 & 2.5 \\ 0 & 0.5 & 0 & 0 & -1 & 0.5 & 0.5 & 1 & 0.5 & 0.5 \\ 0 & 10 & -2 & 3 & 1 & 1 & 1 & 0.5 & 25 & 8 \\ 0 & 5 & -1 & 4 & 1 & 1.5 & 2.5 & 0.5 & 8 & 16 \end{pmatrix}.$$

Furthermore, suppose that out portfolio must adhere to the following constraints. We cannot invest more than 30% in a single investment

$$x_i \leq 0.3 \quad \forall i \in 1, \ldots, 10.$$

If investments 1, 2, 3 and 4 are all American investments then at least 50% of our portfolio must be American

$$\sum_{i=1}^{4} x_i \geq 0.5.$$

Table 1: The returns for the 10 different investments.

| Investment | Investment name | return (£million) |
|---|---|---|
| 1 | Treasury | 5 |
| 2 | Hardware | 17 |
| 3 | Theatre | 26 |
| 4 | Telecom | 12 |
| 5 | Brewery | 8 |
| 6 | Highways | 9 |
| 7 | Cars | 7 |
| 8 | Bank | 6 |
| 9 | Software | 31 |
| 10 | Electronics | 21 |

If investments 2, 3, 4, 9 and 10 are considered to be high risk investments then our portfolio should consist of no more 1/3 high risk investments

$$x_2 + x_3 + x_4 + x_9 + x_{10} \leq \frac{1}{3}.$$

The return we obtain from our investment should be at least £9 million

$$\sum_{i=1}^{10} r_i x_i \geq 9,$$

and finally we can not invest more than 100% of our income

$$\sum_{i=1}^{10} x_i \leq 1.$$

To tell Xpress that we are going to use quadratic programming we need to add another module:

```
uses "mmxprs", "mmquad"
```

Now, in the declarations block, we declare the covariance matrix, which is read later as other arrays in previous examples:

```
Q: array(asset,asset) of real
```

Since we have loaded the module `mmquad`, now we can use a quadratic expression in the objective function:

```
risk:= sum(a1,a2 in asset) Q(a1,a2)*frac(a1)*frac(a2)
```

## 2 Output formatting

We would like to align the information we print by columns.

For this, we are going to use the command `strfmt`. If we write `strfmt(mystring,a)`, then a spaces are reserved for the string `mystring`. The text is aligned to the right (if we want to align to the left, the second parameter must be `-a`). For example, if we write (assuming the proper sets and arrays are defined)

```
forall(a in asset) writeln(strfmt(asset_name(a),11),": ",
strfmt(100*getsol(frac(a)),2),"%.")
```

the output looks like this:

```
   Treasury: 30%.
   Hardware: 7.15389%.
    Theatre: 7.3826%.
    Telecom: 5.46351%.
    Brewery: 12.6549%.
   Highways: 5.91099%.
       Cars: 0.333552%.
       Bank: 29.9999%.
   Software: 1.10004%.
Electronics: 5.55215e-07%.
```

Notice that we have used `strfmt(100*getsol(frac(a)),2)` to write the solution in percentage format and with two digits (before the decimal point). But, if we wish to align the text better, we can, for example, use 5 digits, being 2 of them decimal. We write

```
forall(i in asset) writeln(strfmt(asset_name(i),11),": ",
strfmt(100*getsol(x(i)),5,2),"%.")
```

Now, the output is much better:

```
    Treasury: 30.00%.
    Hardware:  7.15%.
     Theatre:  7.38%.
     Telecom:  5.46%.
     Brewery: 12.65%.
    Highways:  5.91%.
        Cars:  0.33%.
    Bank: 30.00%.
    Software:  1.10%.
 Electronics:  0.00%.
```

Consult the documentation for `strfmt` for more details.

## 3 Exporting data to a file

Sometimes we need to save the solution to a file. For example, let us take a portfolio model as starting point and assume that we are asked to write the objective function and the values of the solution in a text file. This can be done as follows:

```
fopen("myqportfolio.txt",F_OUTPUT)
writeln("The optimal portfolio is:")
writeln
forall(i in asset) writeln(strfmt(asset_name(i),11),": ",
strfmt(100*getsol(x(i)),5,2),"%.")
fclose(F_OUTPUT)
```

The first line means that we create a new file name `myqportfolio.txt` where we will save our data. The last line closes the file. The lines in between are the same that we used before, but now there is one important difference: since we are writing to the file, no information is displayed on the screen.

### Exporting to a `csv` file

Sometimes it is useful to export data to a `csv` file that we can use with Excel (for example, to draw charts). In this case, each row that we want to have in Excel is a data row that we must save to our file. The columns are separated with commas ",". This can be done directly with the `mmsheet` module, but we show the low-level version that generalizes to other file formats.

For example, in order to save the solution of our portfolio model to a `csv` file, we write the following:

```
fopen("myqportfolio.csv",F_OUTPUT)
writeln("Asset,Fraction")
forall(i in asset) if(getsol(x(i))>0.001) then
writeln(asset_name(i),",",100*getsol(x(i))); end-if
fclose(F_OUTPUT)
```

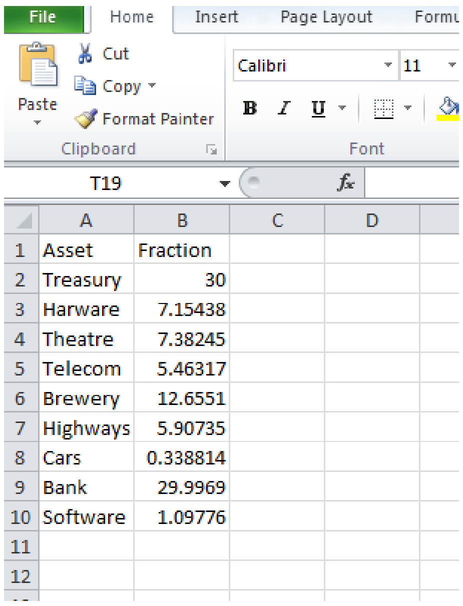
First we open a `csv` file with name `myqportfolio`. Then we write the column headers. After that, we write the rows with the information we are interested at and, finally, we close the file.

The file looks like this:

```
Asset,Fraction
Treasury,30
Harware,7.15438
Theatre,7.38245
Telecom,5.46317
Brewery,12.6551
Highways,5.90735
Cars,0.338814
```

```
Bank,29.9969
Software,1.09776
```

But when we open it with Excel, we can see the desired result in Figure 1.



| | A | B | C | D |
|---|---|---|---|---|
| 1 | Asset | Fraction | | |
| 2 | Treasury | 30 | | |
| 3 | Harware | 7.15438 | | |
| 4 | Theatre | 7.38245 | | |
| 5 | Telecom | 5.46317 | | |
| 6 | Brewery | 12.6551 | | |
| 7 | Highways | 5.90735 | | |
| 8 | Cars | 0.338814 | | |
| 9 | Bank | 29.9969 | | |
| 10 | Software | 1.09776 | | |
| 11 | | | | |
| 12 | | | | |

Figure 1: csv file.

## 4 Solving models repeatedly

If we change the minimum level of return in a portfolio model, it is very likely that the optimal minimum risk will change. If we are interested in studying the trend (i.e., how the risk changes when we increase the minimum return required), we will need to solve several times the same model where only one constraint changes (the constraint imposing a minimum return level). We are going to see how to do this and save the information in a `csv` file so that we can draw a scatter plot with Excel.

We are going to start by imposing a minimum return of 0 and then we will modify the constraint by increase this minimum return in jumps of 0.25. This is why we introduce a parameters block:

```
parameters
    jump = 0.25
    max_minimum_return = 15
end-parameters
```

Parameters are values that will not change during the code. Here, we have the jump described before and also the maximum return that we will use (that is, we will be repeating the model with 0.25, 0.50, 0.75, and so on until 15). We define `minimum_return: real` in the declarations block because this value does change in the different models.

Now, we do not include the constraint on the minimum return with all the other constraints because this constraint will change from model to model. Instead, we define a `repeat` loop. When we use `repeat` *mycode* `until` *mycondition*, we are running *mycode* until *mycondition* is true. We have to be careful and make sure that, eventually, this condition will be true. Otherwise, we will enter into an infinite loop.

First, we write:

```
minimum_return:=0
fopen("data_frontier.csv",F_OUTPUT)
writeln("Return,Risk")
```

We do the following steps:

1. Initialize `minimum_return` with value 0.

2. Create and open to write a file `data_frontier.csv`.

3. Write the headers of the columns.

```
repeat
    ! Minimum expected return
    ! This constraints is updated every we go to the start of the loop
    Min_return:= sum(i in asset) return(i)*x(i) >= minimum_return
    minimize(risk)
    if(getprobstat = XPRS_OPT) then
        writeln(minimum_return,",",getobjval)
    else
        writeln("Not solved to optimality with minimum_return of ",
        minimum_return)
    end-if
    minimum_return+=jump
until((getprobstat=XPRS_INF)  or (minimum_return>=max_minimum_return) )
```

Notice that we repeat the code inside the loop until one of these two things happens:

1. We solve an infeasible problem.

2. We have covered all the range of values for `minimum_return` that we wanted.

The reason why we stop when we find an infeasible problem is that any further problem (with a larger minimum return requirement) is going to be infeasible too. Therefore, there is no point in solving them.

Now, looking at the code inside the loop, notice that we have the constraint for the minimum return level and we give it the name `Min_return`. By writing it like this, every time we begin the loop, we redefine this constraint and, thus, the new right-hand side is used.

Next, we write the values we are interested at in case the solution is optimal (or throw a message of not optimality). Last, the value of `minimum_return` is updated by increasing it `jump` units.

Note also that we have written `minimum_return>=max_minimum_return`, that is, we use an inequality instead of an equality. The reason again is good practice because we are using real numbers. Depending on the precision of the computer, if we write an equality, the comparison might not be true and the stopping rule would not be met, which could lead to an infinity loop or, at least, longer than desired.

Finally, once we have the `csv` file, we can open it with Excel and draw a scatter plot to show the line of the tendency risk versus minimum return (the *efficient frontier*). You can see the outcome in Figure 2.

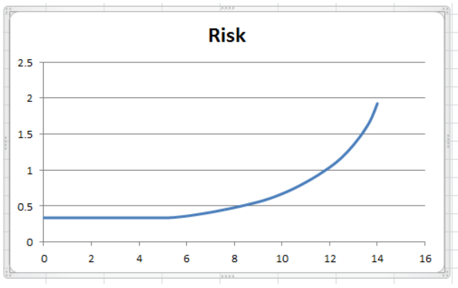


Figure 2: Risk trend.