# Spreadsheet Design Guidelines

## MTH785P – Programming for Business Analytics

| | |
|---|---|
| **Date** | 31/08/2017 |
| **Version** | 1.0 |
| **Status** | Final |

# Table of Contents

# 1 Introduction

## 1.1 Objectives

These guidelines outline best practice for the design of all spreadsheets during the Master's module MTH785P – Programming for Business Analytics

## 1.2 Scope

These guidelines cover an approach to spreadsheet design for spreadsheets that are developed and managed by students on the MTH785P – Programming for Business Analytics module in Microsoft Excel.

## 1.3 Ownership

The Course Leader for MTH785P – Programming for Business Analytics is responsible for the ownership of this document, feedback collation and periodic review.

# 2 Mandatory worksheets

## 2.1 Version

The first worksheet in any new spreadsheet should be simply called "Version". This will be saved in the Master template.

On creation:

- The version should be set to "0.1".
- The creation date should be entered in "dd/mm/yyyy" format with a date corresponding to the day the last change was made before being sent for review.
- A Description should be entered initially as "Created first version".
- Your full name should be entered in "Changed By".
- Once reviewed, the reviewer should enter their name in "Checked By"

## 2.2 Purpose

This worksheet should be the second in your new spreadsheet and should be called "Purpose". This worksheet can be loaded from the Master template

The purposes of this worksheet are:

- To outline the purpose of the spreadsheet
- To note any known limitations of the spreadsheet
- To note any known issues with the spreadsheet

This worksheet can be viewed as the model specification. It is important to lock down the requirements of the model and document them here before proceeding to the rest of the spreadsheet design.

## 2.3 Overview and Key

This worksheet should be the third in your new spreadsheet and should be called

" Overview and Key". This worksheet can be loaded from the Master template

It should contain an overview of all the worksheets, which should give a brief description of the worksheets' purpose.

The key is a useful method of quickly identifying spreadsheet elements consistently.

The key is colour coded for ease of use and the following colour coding convention has been adopted for MTH785P – Programming for Business Analytics:

| Colour | Item |
|---|---|
| | User Inputs |
| | Parameters |
| | Data |
| | Hard-coded values (including macro generated input) |
| | Intentionally blank cells |
| | Calculated Values |
| | Checks that have failed |
| | Descriptive worksheets and comments (no calculations) |
| | Outputs |

A coding convention for the purpose of worksheets has also been adopted as follows:

C = Calculations, D = Data, O = Outputs, P = Parameters, I = Inputs

## 2.4 Inputs and Parameters

These should be the fourth and fifth worksheets, respectively, in your new spreadsheet and should be called "I. Inputs" and "P. Parameters".

It is important to distinguish between inputs and parameters – in this context an "Input" is a response required from the user of the spreadsheet, which must be manually entered into a cell.

A "parameter" is a required piece of data for the correct configuration and functioning of the spreadsheet model. Parameters should include switches, which a user is restricted from changing.

We can illustrate the difference with a simple mortality rate look up example.

**User inputs**: Age, Smoker status

**Parameter**: AM92 (as this is the particular mortality table required)

For both parameters and user inputs it is good practice to define a meaningful name for the cell in which the value is going to be entered.

For example, if you are taking an input of an integer age define a named range for the cell called "intAge". Try to avoid over-complicated names – they should be descriptive but not cumbersome.

User inputs and parameters should be coded according to the table above. Appropriate validation should be added to cells to restrict the entry of invalid data.

## 2.5 Instructions and Control

If the spreadsheet you are designing contains VBA then an “Instructions and Control” worksheet must be added. This should be located towards the front of the worksheet, probably after the “Overview and Key” worksheet, but this depends on whether it requires feeds from Inputs or Parameters.

A sample “Instructions and Control” worksheet can be loaded from the Master template.

This worksheet should detail any Pre-requisites for the workbook in the first section.

In the second section it should contain instructions on how to populate any input fields required for the running of VBA code contained within the workbook.

This section should also contain sensibly labelled command buttons to run any macros.

# 3 Calculations

## 3.1 General

Spreadsheets, where possible, should be designed to operate with calculations set to automatic in order to reduce the risk of errors. This may require complex formulae / processes to be broken down into several “pieces”. If the spreadsheet does not operate with calculations turned to automatic then the reason why and the operation of the spreadsheet should be documented in the “Limitations” section.

Calculations in a worksheet should logically flow from left to right and / or top to bottom.

## 3.2 Formula construction

Any formula should be written with a view to making it as comprehensible as possible for the next person picking up the spreadsheet. Writing an intricate formula that is very time-consuming for somebody else to understand adds little value.

Never hardcode values as parameters to a formula. Always link them to either the “Inputs” or “Parameters” worksheet.

If a formula cannot be “dragged” then a comment should be added to say why and at what point the formula becomes consistent with those cells adjacent. For example a time 0 value may be obtained or calculated in a different way to those from time 1 forward.

## 3.3 Calculation Checks

These will be individual to the calculations being performed and will need to be tailored to suit the purpose.

The key criterion is that all checks should be easily seen and highlighted. The results of those checks should be readily apparent to the person reviewing / updating that worksheet.

Use the colour key to make checks apparent and link the results of checking to the overall sign-off.

## 3.4 Outputs

The outputs of any calculations produced by the spreadsheet should be either:

- Stored on an “Outputs” tab in the current spreadsheet
- Pushed to an external spreadsheet or flat file

In either instance the location of the outputs should be documented as well as the method to reproduce them. The documentation should also include evidence of testing and sign-off.

Access to edit the outputs should be restricted and the Output file(s) and / or worksheet should be read-only and password protected when not in development.

# 4 Other Workbook considerations

## 4.1 Spreadsheet Flow

The spreadsheet should be structured in such a way that the individual worksheets flow from left to right. A logical thought progression should bring you from start to finish this way.

As a simple example the below would be from left to right in a well-designed workbook:


The idea is that Calculations should read from “Inputs” and “Outputs” should read from “Calculations”. It is bad practice to have worksheets further to the right contain information pertinent to worksheets further to the left and this should be avoided.

This may result in additional planning prior to constructing the spreadsheet but it is worth the time.

In a situation where there are multiple calculations and multiple outputs (especially where some calculations follow on from a set of outputs) just try to structure in a sensible fashion – as a suggestion:


Finally - When saving the spreadsheet it should be saved with the first tab opened.

## 4.2 Spreadsheet Transparency

The spreadsheet should be as transparent and easy to follow for other users / developers as possible. This means:

- Minimising use of hidden rows / columns
- Minimising use of hidden worksheets
- Not colouring an input to a cell the same colour as the cell to conceal its contents

If you are using hidden elements in a spreadsheet, ensure that the location of these is apparent and documented so they can easily be reviewed and audited.

## 4.3 Hardcoding

“Hardcoding” means typing a value into a formula in Excel rather than have the formula refer to a cell value. The simple statement here is **NEVER DO IT**. An example of hardcoding is:

Min(MyValue, **100**)

A simple way around this would be to have a parameter called MaxValue on the “P. Parameters” tab and set its value to 100. Then have the formula read:

Min(MyValue, **MaxValue**) Hardcoding is one of the worst practices in any form of coding, including spreadsheet development, there are many reasons:

- It is very difficult to audit
- Often the hardcoded figure is not updated when it should be something different (for example updating a hardcoded tax rate after a budget)
- It is the most commonly used method to “adjust” numbers to make the changes difficult to find / understand and therefore a significant source of fraud risk
- A hardcoded number seldom comes with a proper explanation of what it is doing there. If the figure is placed as an input or parameter then it invites comment and explanation.
- It creates legacy problems as future developers of the application / model will not know about the hardcoded values

## 4.4 External linking

“External linking” is where you have a direct link from within a spreadsheet to another source and is often necessary within complicated models.

If a link is within a calculation or embedded somewhere deep within a spreadsheet it can cause problems and should be avoided:

The reasons are similar to the hardcoding of values (though slightly less so) – it also creates the additional problems of:

- Slowing down load times in the spreadsheet
- Where the external link is invalid (for various reasons – it has moved, been archived, the user does not have access to that particular folder) it can cause Excel to spam message boxes for each instance of a reference to the link, which must be clicked through to access the spreadsheet
- It creates security concerns as the linked spreadsheet is not within the control environment being created
- You can’t easily be sure that every workbook in the linked chain was updated in the right order

If you find you need to reference data / calculations from another spreadsheet then this should be managed from either the “I.Inputs” or “P.Parameters” sections, depending on whether the user needs to change the location of external input. The external spreadsheet link should be listed here, together with an appropriate label / description.

If large quantities of data need to be imported then this should be managed through a Visual Basic for Applications solution.

Validation of the existence of the external workbook / worksheet would then take place just once, prior to any processing. Correct error handling will reduce processing time and errors opening / finding the external data.

## 4.5 Printing

Spreadsheets should be formatted with the view that each tab on the spreadsheet is likely to, at some point, be printed. The following should be performed for each worksheet:

- A print area should be defined and checked in print preview to ensure a print-out will be legible.
- A Header and Footer should be added – these should comprise of the full filename and file path and also the date (so it is possible to check the version of the spreadsheet printed as well as the date it was printed).
- In the case of worksheets that span multiple rows / columns please use the “Print Titles” functionality to repeat labelling rows and columns on each sheet so they can be easily read and understood.
- One sample print of the whole workbook should be performed post-development to ensure it is legible.

# 5 Testing

The testing of any spreadsheet model will naturally be dependent on the purpose of the spreadsheet.

Testing performed should be capable of being evidenced if required so consider keeping a separate testing folder to hold copies of test results.

If the spreadsheet is to be used as the main model for business purposes then a formal test pack should be designed and used.

Testing should consist of:

- **Functional testing**
- **Regression testing**
- **User acceptance testing**
- **System integration testing**

## 5.1 Checks

The last worksheet in your spreadsheet should be a “Checks” worksheet. This will contain evidence of:

- Compliance with all guidelines in this document
- All error checking completed and passed
- Links to results of testing
- Sign-off
- This can be used to evidence TAS compliance.

For large workbooks it may be more practical to keep the Checks spreadsheet towards the front so that a reviewer or auditor can easily access it. This should be the only exception to the “Spreadsheet Flows” rule if this is adopted.

# 6 Review

Once the spreadsheet is completed to your satisfaction you should review prior to submission. The spreadsheet should be capable of being understood and reviewed as a standalone piece of work with no outside documentation / explanation.

The review should consist of:

- Version control correct
- Satisfactory explanation of the purpose of the spreadsheet and instructions for use
- Adherence to the colour code
- Adherence to the spreadsheet design principles in this document
- Adherence to VBA coding standards
- Correctness & Accuracy of spreadsheet and results
- Within File Size Limits

Once you are happy that the spreadsheet is fit for purpose it should submitted.

In the review sections on each worksheet the reviewer should ensure they are happy that the function of that worksheet is as intended, compliant with these guidelines and the overarching principles, and correct.

Once they are happy with each worksheet and the review has been completed there should be an automatic check on the “Checks” tab that identifies and notifies the review is completed.