M6 Excel Topics

Topics in Insurance, Risk, and Finance ¹

Professor Benjamin Avanzi²



18 July 2024 10:14

 $^{^{1}}$ References: Slager and Slager (2020) and Katz (2023) | \rightarrow <u>latest slides</u> $^{1/46}$ 2 Contributions from Ho Ming Lee and William Ho are gratefully acknowledged.



- 1 Introduction and assumed knowledge
- 2 Data wrangling and Exploratory Data Analysis ("EDA")
- Oynamic arrays
- Oynamic references
- 5 General etiquette, auditing, and tools
- Simulation, Macros and VBA
- Next steps



- Introduction and assumed knowledge
- Data wrangling and Exploratory Data Analysis ("EDA")
- Oynamic arrays
- 4 Dynamic references
- 5 General etiquette, auditing, and tools
- 6 Simulation, Macros and VBA
- Next steps



- Introduction and assumed knowledge
 - Why Excel?
 - Issues with Excel
 - Assumed knowledge
 - Some keyboard shortcuts



Why Excel?

- Excel is great to spread data and calculations in a tabular form, and have a visual overview.
- Most financial modelling is done in Excel (at least initially).
- In actuarial work, many more advanced codes (in R, Python, C++, C#, VBA, ...) often starts with someone playing around in Excel, and once proof of concept is approved, this moves to proper coding.
- It is assessed in CM2-B (!).



- 1 Introduction and assumed knowledge
 - Why Excel?
 - Issues with Excel
 - Assumed knowledge
 - Some keyboard shortcuts



Issues with Excel

Excel is notoriously problematic in certain areas:

- Lack of transparency one can't see the formulas unless you click in a cell.
- Similarly, you can't see the formulas and understand where the numbers come from unless you are in Excel (a problem for reports, presentations etc).
- Tiny mistakes (the wrong letter, the wrong number in cell reference, forgetting a \$) can have huge consequences in the end (Note that the job of Excel auditor actually exists!).
- Lack of good documentation capability (as opposed to code); this
 makes collaboration and audit difficult, and creates an operational risk
 (e.g. spreadsheet developer leaves).
- Lack of rigour in the construction of a model (input, assumptions, intermediary calculations, output).

The following improved over recent years, but are still downsides of Excel as compared to some coded languages:

- Can't handle (seriously) large data sets.
- Lack of good and flexible data cleaning and manipulation capabilities.
- Some operations are just a lot easier to perform with code (e.g. flip a vector around, sum over a diagonal, . . .).

I know there are counter arguments for all of those, but this presupposes you know what the solutions are. You'll learn some of those here!



- Introduction and assumed knowledge
 - Why Excel?
 - Issues with Excel
 - Assumed knowledge
 - Some keyboard shortcuts



Assumed knowledge

See prerequisite knowledge on the website. Some extracts:

- Autofill: Chapter 3, p. 105-116, and p. 298-301
- Named ranges and constants: Chapter 7, page 312-332
- Absolute and mixed cell references (\$): Chapter 7, pages 332-342
- New Excel 2019 functions (IFS, MAXIFS, MINIFS): Chapter 8, pages 381-398
- Formula Auditing: Chapter 9, p. 436-439
- Paste special (incl, e.g. Transpose): Chapter 11, pages 518-530

Page references are for Slager and Slager (2020), see link here.

Also, see tab Assumed Knowledge in the module 6 lectures spreadsheet.



- 1 Introduction and assumed knowledge
 - Why Excel?
 - Issues with Excel
 - Assumed knowledge
 - Some keyboard shortcuts



Some keyboard shortcuts

There are many keyboard shortcuts that can help you work with Excel efficiently. The "Excel Wizards" hardly ever use their mouse. You are advised to try and learn as many of those shortcuts as you can!

Examples of useful shortcuts are (e.g. for Windows):

- F4: Cycles through combinations of absolute and relative references
- ctrl + shift + arrow key: Extend the selection of cells to the last non-blank cell towards the specified direction
- ctrl + PageDown/PageUp: Move to the next/previous sheet
- shift + F11: Insert new worksheet

See Microsoft - Keyboard shortcuts in Excel for list of all keyboard shortcuts.



- 1 Introduction and assumed knowledge
- Data wrangling and Exploratory Data Analysis ("EDA")
- Oynamic arrays
- 4 Dynamic references
- 5 General etiquette, auditing, and tools
- 6 Simulation, Macros and VBA
- Next steps



- 2
 - Data wrangling and Exploratory Data Analysis ("EDA")
 - Data Wrangling
 - Pivot tables
 - Pivot charts



Data Wrangling

- Sometimes we need to work on data from an external source.
- The data could be messy to work with.
 - There could be non-printable characters such as tabs (\t), new lines (\n) in the original data.
 - Check List of ASCII values for ASCII codes for non-printable characters.
 - We may want to process the strings to retrieve information.
 - Data formats are inconsistent.
- It is important to clean our data before analysis!



Some data cleaning steps

- Editing Texts
 - CLEAN(), TRIM(), and SUBSTITUTE() to deal with any non-printable characters.
 - Merging and splitting columns with functions like LEFT(), RIGHT() and MID() and SEARCH(). Wildcard characters? and * are also useful.
- Removing duplicate rows: Data → Data Tools → Remove Duplicates.
- Converting table into "machine-readable" formats.
 - An example in Kolokolov, 2023 (Chapter 1, P.2-9).
 - Macros would be useful here!
- There are much more functions and steps can be used to clean your data - see Microsoft - Top ten ways to clean your data.



- 2
 - Data wrangling and Exploratory Data Analysis ("EDA")
 - Data Wrangling
 - Pivot tables
 - Pivot charts



Pivot tables

- Pivot tables are often considered as very difficult to master, but they are not that difficult to start with.
- Example (in module 6 spreadsheet): FIFA WWC
 - Insert / Pivot Table.
 - See how you can use variables as filters, rows, or columns. Move them around.
 - See how columns can display other things than Sum, such as Count, Average, Max, Min, or Product
- Note there are recommended Pivot Tables (automated recommendation within Excel); in the case of FIFA WWC it is not very helpful.
- Reference: Chapter 15 of Slager and Slager (2020).



- 2
 - Data wrangling and Exploratory Data Analysis ("EDA")
 - Data Wrangling
 - Pivot tables
 - Pivot charts



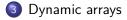
Pivot charts

- A pivot chart can be created from the Pivot Table but also directly from the data.
- A major difference with start charts is that you it will be somewhat "interactive" there will be buttons you can use to alter the chart.
- Example (in module 6 spreadsheet): FIFA WWC
 - Insert / Pivot Chart;
 - In the example I changed the style of graph to "Combo" to allow for the two different scales;
 - I also included a "slicer" (click on chart / insert slicer), in order to easily filter by squad.
- Reference: Chapter 15 of Slager and Slager (2020).



- Introduction and assumed knowledge
- 2 Data wrangling and Exploratory Data Analysis ("EDA")
- Opposition of the second of
- 4 Dynamic references
- General etiquette, auditing, and tools
- 6 Simulation, Macros and VBA
- Next steps





- Spilling
- Think in vectors
- The # sign
- New formulas
- Lookup functions



Spilling

- One advantage of programs like R are the easy use and manipulation of vectors.
- Excel can do similar things, and the vectors are called arrays. This is new (post Office 365), and is a bit of a game changer.
- Before Office 365, Excel was incapable of depositing results beyond just 1 cell. This is called "spilling".
- Note Excell will need the required space to spill.
- Main reference is Katz (2023) we'll only introduce this here.



Example

- Here we introduce array formulas.
 - If you calculate the sum of an array you'll get a single number.
 - The result of an array formula (such as LEN()), when you input an array, will give you an array.
- See module 6 spreadsheet:
 - LEN() gives an array.
 - You could then get the sum without having to put the array anywhere: SUM(LEN(B3:B6)).
 - Note that when I wrote the above, it became SUM(LEN(B3#)) automatically - more on that later.
- SUM(LEN(B3:B6)) will work in any version of Excel because it requires only one cell to output, but not LEN(B3:B6) as it requires several cells ("spilling").
- Note you can spill named ranges, too!



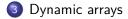
- Oynamic arrays
 - Spilling
 - Think in vectors
 - The # sign
 - New formulas
 - Lookup functions



Think in vectors

- Once you understand you can create vectors and either display or manipulate them, Excel becomes a lot more powerful.
- You can also use arrays in arguments of known formulas such as VLOOKUP(),
 - For instance VLOOKUP(.,.,{2,5}) will return value from the 2nd and 5th columns row-wise.
 - If you use VLOOKUP(.,.,{2;5}) (with the semicolon) they will display columnwise.
 - See examples in module 6 spreadsheet.





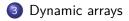
- Spilling
- Think in vectors
- The # sign
- New formulas
- Lookup functions



The # sign

- The # sign when added to a reference to a cell where a dynamic array is written will duplicate that array (and spilled results).
- It is shorter, but also it will dynamically change the size of the array
 - This can be desired or not.
 - See example in module 6 spreadsheet.





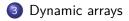
- Spilling
- Think in vectors
- The # sign
- New formulas
- Lookup functions



New formulas

- There are a number of new formulas which were available in coding languages like R for a long time, which can be useful, and which are now available in Excel
 - for instance SEQUENCE(), UNIQUE(), FILTER(), RANDARRAY()...
- Some of those are exemplified in module 6 spreadsheet.
- You are encouraged to browse through. They really bring data handling in Excel a little closer to coded languages such as R.





- Spilling
- Think in vectors
- The # sign
- New formulas
- Lookup functions



Lookup functions

- Lookup functions are essential for Excel users.
- Useful when you need to search within a single row or column to find a corresponding value in the same position in a second row or column.
- Multiple lookup functions are available.
- All of them works with arrays of lookup values!



VLOOKUP()

- Available for all versions of Excel, so it is reliable when sharing spreadsheets.
- Some drawbacks of VLOOKUP():
 - Lookup Column Must Be the First Column
 - Can be confusing when specifying the return column
- HLOOKUP() is the horizontal version of VLOOKUP().
- Check this video to see how it works.



XLOOKUP()

- An upgraded version for VLOOKUP() released in 2020 not available in Excel 2016 and Excel 2019.
 - Robust lookup function: You can search any direction.
 - Can return multiple arrays as well.
 - The lookup array does not have to be the first column/row.
 - Different search modes available: from first to last, last to first, etc.
 - An if not found argument to allow combinations with functions like IFERROR, IFNA.
- Watch this video to see how it works.



MATCH(), XMATCH()

- Both functions work similarly by finding the index of your lookup value within the lookup array.
- XMATCH() is more robust by providing new match mode and search mode.
- By combining with INDEX(), it returns the value instead of the index inside the lookup array.

Check module 6 spreadsheet for comparisons between these lookup functions.

Reference: Murray, 2022 (Chapter 7, 11)



- Introduction and assumed knowledge
- ② Data wrangling and Exploratory Data Analysis ("EDA")
- Opposition of the second of
- 4 Dynamic references
- 5 General etiquette, auditing, and tools
- Simulation, Macros and VBA
- Next steps



Dynamic references

- Sometimes we have to work with references that changes position or size over time.
- For instance, we need to adjust our range of references for each development period when modelling the development effects.
- This is tedious to do manually if we have a large dataset!
- Dynamic Referencing will be useful here.



- Oynamic references
 - Some functions for dynamic referencing
 - How to use them for triangles?



OFFSET()

- Allows you to move your initial references to any direction by any number of cells.
- You can select the height and width of the returned range.

Month	Profit	No offset	Move down the reference	by 6 rows		
Jan	624		Move right by 1 column			
Feb	390					
Mar	1000	=OFFSET(B2:B7,0,0)	=OFFSET(B2:B7,6,1)			
Apr	250	Month	109	Move down by 6 rows		
May	439	Jan	530	Move right by 1 column		
Jun	109	Feb	123	Select the first 2 rows only		
Jul	530	Mar	5345	=OFFSET(B2:B7,6,1,2)		
Aug	123	Apr	312	109		
Sep	5345	May	98	530		
Oct	312					
Nov	98					
Dec	672					



INDIRECT()

- Goes to the address specified by the reference text you entered.
 - e.g. INDIRECT(B2) goes to the cell A2 if cell B2 contains the text "A2".
- Your reference text can be a table, a named range, etc.

	Α	В	С	D	Е	F	G	Н	I
1									
2		Month	Profit		Address?		Address	of your range	e?
3		Jan	624		В3		В3	C9	
4		Feb	390						
5		Mar	1000		=INDIRECT	T(E3)	=INDIRE	CT(G3):INDIR	ECT(H3)
6		Apr	250		Jan		Jan	624	
7		May	439				Feb	390	
8		Jun	109				Mar	1000	
9		Jul	530				Apr	250	
10		Aug	123				May	439	
11		Sep	5345				Jun	109	
12		Oct	312				Jul	530	
13		Nov	98						
23/464		Dec	672						

ADDRESS()

- Gives you the address of your specified columns and rows.
- For instance, you will get A1 if you type in the formula ADDRESS(1,1).
- specifies the column.

 This allows you to obtain dynamic addresses by changing your specifi

The first argument specifies the row, and the second argument

- This allows you to obtain dynamic addresses by changing your specified columns and rows based on some index columns/rows.
- By combining with INDIRECT(), you can retrieve the data contained in your dynamic range.



Some drawbacks

- Both OFFSET() and INDIRECT() are volatile functions.
- Volatile functions are functions in which the value can change even if none of the function's arguments change.
- Excel recalculates them even if you changed a cell without these functions!
- Can be computationally inefficient if the spreadsheet relies on them heavily.

See reference tab in module 6 spreadsheet for demonstrations.

Reference: Murray, 2022 (Chapter 11)



- Dynamic references
 - Some functions for dynamic referencing
 - How to use them for triangles?



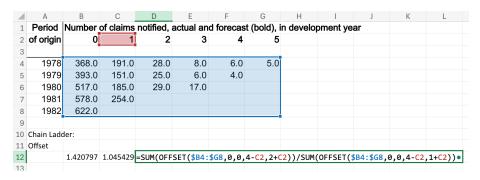
Manual method

	А	В	С	D	Е	F	G	Н	I	
1	Period	Number	of claims	notified,	actual an	d forecas	t (bold), i	n develo	pment ye	ar
2	of origin	0	1	2	3	4	5			
3										
4	1978	368.0	191.0	28.0	8.0	6.0	5.0			
5	1979	393.0	151.0	25.0	6.0	4.0				
6	1980	517.0	185.0	29.0	17.0					
7	1981	578.0	254.0							
8	1982	622.0								
9										
10	Chain Ladder:									
11	Manual Method									
12		1.420797	1.045429	=SUM(B4:	D6)/SUM(B4:C6) •				

- This method requires you to change your reference ranges manually each development period.



Offset method



1. First select the whole triangle (OFFSET(\$B4:\$G8,0,0)). 2. Select the range for summation using offset (The last two arguments in OFFSET). 3. As development period increase, dynamically change the references by editing the height and width of the offset using the development period.



Indirect Address method

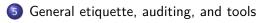
Α	В	С	D	E	F	G	Н	1	J	K	L	M	N	
Period	Number	of claims	notified,	actual an	d forecas	t (bold),	in develo	pment ye	ear					
of origin	0	1	2	3	4	5								
_														
1978	368.0	191.0	28.0	8.0	6.0	5.0								
1979	393.0	151.0	25.0	6.0	4.0									
1980	517.0	185.0	29.0	17.0										
1981	578.0	254.0												
1982	622.0													
Chain Lado	der:													
ndirect ad	ldress													
	1.420797	1.045429	1.012111	1.010084	1.008319									
	Period of origin 1978 1979 1980 1981 1982 Chain Lado	Period of origin Number 0 1978 368.0 1979 393.0 1980 517.0 1981 578.0 1982 622.0 Chain Ladder:	Period Number of claims of origin 1978 368.0 191.0 1979 393.0 151.0 1980 517.0 185.0 1981 578.0 254.0 1982 622.0 Chain Ladder: Indirect address	Period of origin Number of claims notified, of origin 2 1978 368.0 191.0 28.0 1979 393.0 151.0 25.0 1980 517.0 185.0 29.0 1981 578.0 254.0 254.0 1982 622.0 622.0 622.0	Period of origin Number of claims notified, actual an of origin 2 3 1978 368.0 191.0 28.0 8.0 1979 393.0 151.0 25.0 6.0 1980 517.0 185.0 29.0 17.0 1981 578.0 254.0 25.0 25.0 Chain Ladder: Indirect address 100.0 150.0 100.0	Period of origin Number of claims notified, actual and forecase of origin 2 3 4 1978 368.0 191.0 28.0 8.0 6.0 1979 393.0 151.0 25.0 6.0 4.0 1980 517.0 185.0 29.0 17.0 1981 578.0 254.0 1982 622.0 Chain Ladder:	Number of claims notified, actual and forecast (bold), of origin	Number of claims notified, actual and forecast (bold), in develor of origin	Number of claims notified, actual and forecast (bold), in development year of origin Q	Number of claims notified, actual and forecast (bold), in development year of origin 0	Number of claims notified, actual and forecast (bold), in development year of origin Q	Number of claims notified, actual and forecast (bold), in development year of origin 0	Number of claims notified, actual and forecast (bold), in development year of origin 0	Number of claims notified, actual and forecast (bold), in development year of origin 0

1. Select the top left corner of the triangle first (INDIRECT(ADDRESS(4,2))). 2. Dynamically change the reference by editing the location of bottom right corner of the reference (INDIRECT(ADDRESS(7-C2, 3+C2)))).



- Introduction and assumed knowledge
- Data wrangling and Exploratory Data Analysis ("EDA")
- Oynamic arrays
- 4 Dynamic references
- 5 General etiquette, auditing, and tools
- Simulation, Macros and VBA
- Next steps





- Etiquette
- Principles
- Tools for auditing



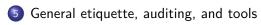
Etiquette

You should build your spreadsheet with (at least) the following **objectives** in mind:

- so as to minimise chances or error (accuracy);
- so as to minimise unnecessary calculations (efficiency);
- so as to make the structure as clear as possible (transparency);
- so as to make updates, changes and extensions possible and easy (extendibility);
- so as to allow someone else to use it easily (user friendliness);
- so as to allow someone else to verify it easily (auditability).

Those are, of course, interconnected. We could add more (for instance, automation of data input via an API, automation of communication objects such as charts, etc...).





- Etiquette
- Principles
- Tools for auditing



Principles

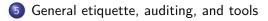
There is no single way to achieve the objectives above, but there are a number of **principles** that one could list, and that will contribute to meeting those objectives:

- Have a separate tab that collects all your assumptions that are valid for the whole spreadsheet (1, 3, 4, 5, 6).
 - Include some explanations about the source/justification of those assumptions.
 - Give your assumptions names (for instance, the technical rate of interest to calculate life insurance could be called techint or similar, for ease of later reference, and to make formulas more easily readable).
- Also include your data sets in separate tabs (1, 3, 4, 5, 6).
 - Name your data (including columns and/or rows if possible; e.g. FIFA WWC in the spreadsheet; see also "Named ranges and constants": Chapter 7, page 312-332).



- Consider colouring / contouring input and output differently (3, 5)
 - This is not always advisable, but if you have a large model with relatively few input (e.g. purchase price and interest rate for a property mortgage schedule) and/or few outputs (e.g. NPV) then this achieves many of the objectives.
- Use named ranges and variables as much as possible (1, 3, 5, 6), unless the variable is going to be used only once.
- Use more advanced formulas if shorter, and avoid too much nesting (1, 3, 4, 5, 6).





- Etiquette
- Principles
- Tools for auditing



Dependents and precendents

Use of dependents / precedents, e.g.:

- go to tab Dependents Precedents PPCI;
- click on one of the outstanding loss projected amounts;
- make sure the formula tool tab is live;
- Oclick on the "trace precedents" sequentially.

This will highlight dependence of each cell to previous cells, sequentially, with arrows.

This is useful for

- understanding the structure of a spreadsheet;
- check formulas (audit);
- debug issues.

(Formula Auditing: Chapter 9, p. 436-439)



Show formulas

- The "Formulas" tool tab should have a "Show formula" tile. This will replace all numeric values by formulas.
 - This is helpful to check what numbers are hard coded, and which are results of calculations. Together with dependents, it helps seeing if everything is as dynamic as it should.
- If you want a formula to be shown all the time start with an apostrophe ', and the formula will show as text.

(Formula Auditing: Chapter 9, p. 436-439)



- 1 Introduction and assumed knowledge
- Data wrangling and Exploratory Data Analysis ("EDA"
- Openamic arrays
- 4 Dynamic references
- 5 General etiquette, auditing, and tools
- Simulation, Macros and VBA
- Next steps



- Simulation, Macros and VBA
 - Simulation
 - Macros
 - VBA



Why simulation?

- Suppose we have some model with given inputs.
- Inputs can be uncertain.
 - Weather, arrival speed of claims, interest rates,...
- We want to derive the distributions of our output.
 - We want to know the statistical properties (e.g. mean, SD) of our output.
 - Sometimes it can be complicated to derive the distribution of the output explicitly.
- By simulating the uncertain inputs, we can generate random observations from our model.
 - These generations will result in an empirical distribution of the output.
 - We can then use this distribution to estimate moments, quantiles, ranges, and probabilities.
- Simulation is widely used in the work by actuaries.



Generating random numbers 1

- RAND() generates random numbers from a uniform distribution on (0,1).
- RANDARRAY() can generate an array of random numbers with specified dimension.
 - You can specify the range of numbers.
- You can also specify whether the generations are discrete or continuous.
- Excel recalculate these functions whenever there are changes in any cells.
- What if we want to generate numbers that are not uniformly distributed?



Generating random numbers 2

- Inverse functions can be useful to generate numbers under different distributions.
 - Use NORM.S.INV(RAND()) to generate numbers with standard normal distribution.
 - In general, to generate random numbers with specified distributions, we can apply their inverse functions to RAND().
- Some inverse functions available in Excel:
 - BETA.INV() for Beta distribution
 - LOGNORM.INV() for lognormal distribution
- To see more inverse/probability functions in Excel, go to Formulas \rightarrow More Functions \rightarrow Statistical.



Discrete random variables

- We can also simulate outcomes from discrete distributions.
- Suppose we have a random variable with three outcomes: A with p=0.5, B with p=0.3, and C with p=0.2.
- First, generate a random number x with RAND().
 - If $x \le 0.5$, the outcome is A.
 - If $0.5 < x \le (0.5 + 0.3) = 0.8$, the outcome is B
 - Otherwise, the outcome is C.
- Why does it work? Inverse transformation.
- More theoretical formulation in Module 9.

See module 6 spreadsheet for illustrations.



- 6 Simulation, Macros and VBA
 - Simulation
 - Macros
 - VBA



What is a macro?

- Sometimes we may need to do a set of actions or tasks repeatedly.
 - e.g. cleaning data, formatting cells, creating tables.
 - This is repetitive and very time-consuming to do cell-by-cell!
- A macro can store this set of actions and run whenever you need it.
 - You can also make minor edits to the recorded actions.



Recording a macro

- **1** A macro can be recorded by going to Developer tab \rightarrow Record Macro.
 - \bullet To turn on the Developer tab, check Quick start: Create a macro
- A dialog box will be opened. You can rename your macro and select a shortcut key for the macro.
- Olick "OK" to create your macro. Excel will now start recording your actions.
- Olick "Stop Recording" to save your macro.
- Note: You can select Use Relative References when recording a macro. In that way, you can run your macro relative to your currently selected cell, instead of a fixed range.



Running a macro

- There are multiple ways to run your recorded macros.
- Running macros with your assigned shortcut.
- Running macros through the Developer tab.
 - Go to Developer → Macro. A list of your recorded macros should come up.
 - Click on the sheet and cells you want to run a macro with. Click "Run" to run it.
- Running macros with buttons.
 - **1** Go to Developer \rightarrow Form Controls \rightarrow Insert \rightarrow Button.
 - Click on the cell you want to insert the button, the Assign Macro box should come up.
 - Choose your macro and click "OK". Now you can run your macro by clicking the button.
- There are more ways to run macros. See P.886-894 in Slager and Slager (2020).



- 6 Simulation, Macros and VBA
 - Simulation
 - Macros
 - VBA



What is VBA?

- VBA stands for Visual Basic for Applications.
- It is a programming language used by Excel (and all Microsoft Office apps).
- Your recorded macros are stored as VBA code.
- You can create your own macros by writing VBA code.
 - They can be assigned in the same way as a recorded one.
- VBA is needed as some actions such as conditioning and looping cannot be recorded.
- Excel writes bad VBA code from recorded macros.



Editing your macros

- You can modify your recorded macros with Visual Basic Editor (VBE).
- Open VBE with Alt + F8 or go to Developer \rightarrow Code \rightarrow Macros \rightarrow select the macro you want to edit \rightarrow Edit.
- Some minor edits you can make to your macros:
 - Switch between reference types.
 - Add/change different types of formatting.
 - Edit cell selections.



Conditions

- By using conditions in our VBA code, we can run different actions according to the given condition, just like IF in Excel.
- But there's more we can do with VBA!
- Some simple examples of conditioning:
 - Change cell colour based on cell value.
 - Display different message boxes based on the input to input box.
 - Display different texts in cells based on the value in your active cell.
- You can do nested if with if-else statements.
- Logical operators are also available for multiple conditions.
 - And, Or, Xor, Not



Loops

- Useful when you want to automate iterative operations.
- There are two types of loops in VBA:
 - For loops are used when you know the number of iterations.
 - While loops are used when you want to stop based on some conditions.
 - Just a general rule. They are interchangeable.
- Some examples:
 - Format cells until reaching the end of a row/column.
 - Fill every *n* cell with some numbers.
 - Keep showing input boxes and do operations until entering specific input.



- Introduction and assumed knowledge
- 2 Data wrangling and Exploratory Data Analysis ("EDA")
- Openie arrays
- 4 Dynamic references
- 5 General etiquette, auditing, and tools
- Simulation, Macros and VBA
- Next steps



Next steps

- All of Katz (2023) is relevant, but take it as a cook book for the assignment. You can go as far as you wish.
- Chapters 16, 17, 18, and 19 of Slager and Slager (2020) are out of scope.
- However, macros and VBA (which is Chapter 19) are essential components of Excel
 - I strongly encourage you to get started. Start by recording a macro, then play around with the code.
 - VBA allows more efficient calculations via compiled code, and is a powerful addition to Excel.
- Some recent development in understanding spreadsheets with Large Language Models: SpreadsheetLLM: Encoding Spreadsheets for Large Language Models.
- Fun fact: the 2021 Excel World Champion is an actuary: Andrew Ngai, now Director at Taylor Fry.

References I

Katz, A. I. 2023. *Up up and Array! Dynamic Array Formulas for Excel 365 and Beyond.* Apress.

Slager, D., and A. Slager. 2020. Essential Excel 2019. 2nd ed. Apress.

