**Sample Text**

**(and Proofreaders)**

***by Paul Beverley***

Version 20.09.19

*(At the end is a* **Changes Log** *to show what has changed since you last downloaded a copy.)*

***Thank you!***

I’m very much aware that any gift, talent or skill that I might have comes from God. And I’m very grateful to Him! (See also Appendix 10.) And I’m very grateful, too, to my wife Sue for caring for me and being patient with me.

I also want to thank my professional colleagues in the UK’s Society for Editors and Proofreaders (www.sfep.org.uk) for inspiring me and encouraging me to create the enclosed macros.

So, thanks to everyone for giving me such joy in life!

***Thank me?***

If you find this book helpful and would like to say thank you, please make a gift to a charity of your choice, in your own country, and preferably one that aims to alleviate human suffering in some way. Thanks! But if you prefer to make a donation to me – or through me – I have a PayPal account with my email address paul@archivepub.co.uk.

**Finding the macros**: Note that the macros are stored separately, in the second volume of the book: *The Macros*. The name of the macro is given at the end of each macro description. You can use that name to search through *The Macros* for the particular macro you want.

### **Other notes:**

### **Printing the book?** – This is intended to be an electronic document, but if you do decide to print it, you can improve the printout page coverage: either (a) delete the two comments in the section headed ‘Add a Comment’, or (b) go to Review toolbar, look at Track Changes, and select ‘Final’, and not ‘Final: Show Markup’.

## Mac users start here, please

(**New problem**: (Sept 2018) I think it’s Word 2016, but people are reporting errors: the command Application.Resize is not available. They are only cosmetic touches to the macros, so if such an error occurs, just disable the command by adding an apostrophe at the beginning of the line: ' Application.Resize and the line will turn green.)

I don’t own a Mac (though I did for about 12 years, starting in the days of the longtime MacPlus with its 9″ B&W screen!), but I’ve done my very best to make my macros Mac-compatible. A few points to note:

1) None of these macros will work with **Word 2008** because it can’t run macros written in Visual Basic.

2) **Word 2004** does have Visual Basic, but it does not have the macro command Replace(). This is a powerful text-manipulating function that I use a lot, so if you want to use my macros, you will find yourself very frustrated unless you upgrade at least to 2011. I can only apologise.

3) With the possible exception of the MultiFile\_ macros, all of the macros *should* work with **Word 2011 and later**. If you find any that cause problems, please let me know (I’ve had no longtime error reports since about 2015/16).

Unfortunately, in the Mac version of Visual Basic, the file handling has some issues with some versions of the Mac operating system, so I can’t absolutely guarantee that my MultiFile\_ macros will work without problems. Mac users who have used them say that it seems to work OK provided that the names of your files are relative short; best to keep them less than 18 characters long.

4) One macro, DocAlyse, can sometime say ‘Compile error: procedure too large’. If you get this error, try using the macro, DocAlyseForMac. Even if that gives the same error, there is a solution; please email me and ask for a copy of DocAlyseForMac\_2014.

# Introduction

**What is this document?**

This is a freely distributable set of computer tools (macros) for use with Word, programmed by and for editors and proofreaders. *(Writers and author will also benefit from using some of these macros because they have to read and edit their own texts to try to make them more consistent.)* If this book is useful to you, please use it. If you think that others will find it useful, please pass it on. However, the copyright remains with the author, Paul Beverley, and if you want updated versions of these macros, they are always available from my website at: http://www.archivepub.co.uk/TheBook

**Disclaimer**: Many parts of this book have not been edited or proofread, and I can’t guarantee that the macros won’t go wrong, but I’ll do my best to correct anything that you bring to my attention. I keep updating the book, often on a daily or weekly basis, so as a whole it’s a ‘work in progress’. Thanks for your tolerance.

*Paul Beverley, paul@archivepub.co.uk*

## Who is Paul Beverley?

After spending over 20 years writing, editing and publishing using Mac and Acorn computers, I became a freelance technical proofreader in 2005. Later I began doing editing, using Microsoft Word on a Windows PC. Through active CPD, I became an ***Advanced Member of the UK’s Society for Editors and Proofreaders***, and was then awarded a ***Licentiateship of the City & Guilds Institute in Editorial Skills***.

On the old Acron computers, I had developed a number of editing tools, so when I moved to MS Word I missed them, but I soon discovered macros – and they have allowed me to develop even more sophisticated tools than I had used previously. So I thought that other editors might like to benefit from my development work – no point in reinventing the wheel.

Is this pure altruism on my part? Am I keen to enable my rival freelance editors to improve their work rate and increase the consistency of their Acorn output? Well, yes, I do find it satisfying to discover that others are benefitting from my efforts – who wouldn’t?! But I also hope that, by making these macros freely available, I might increase my range of contacts around the world (these macros can be used in other language groups – they aren’t just restricted to English language users).

### Now with added training ...

In fact, since I first released this book in January 2010, I have started running training workshops. If you want to get a group together and invite me, I’d be happy to come and do a demonstration, to whet your appetite, or I could run a workshop where you install macros on your laptops and I would then provide help, instruction and encouragement.

Unlike the book, that training would not be free, but in most cases, I only charge for my expenses. There is not normally a tuition fee. The thing is I’m partially retired, and doing training gets me out of my home-office, and I really enjoy making new friends. I’m happy to travel (almost) anywhere in the world – you ‘only’ have to pay my expenses, and I’m happy to travel to new countries.

## Quick start

If you already know about macros and just want to see what this book has to offer, please go straight to ‘My twelve favourite macros’.

**General macro warning**: One or two of the more complex macros can take a long time to run, so if nothing seems to be happening, please be patient. The worst thing you can do is to click on the screen to try to see if anything is happening. Word sometimes crashes when you do that.

## Non-programmers start here

*(This video might help: https://youtu.be/pN8SO6E8dLg)*

If you already realise the value of Word macros but you feel nervous about using them because “I’m not a programmer”, let me try to reassure you on a number of issues.

### What is a (Word) macro?

It’s a computer program which, in general terms, can ‘do things with words’. As with computer programs generally, macros can be small programs to do very simple jobs, or they can be long complicated ones that do very sophisticated tasks.

### Why use macros?

Since ‘doing things with words’ is what an editor does, maybe some of the tasks that we do manually could, to some extent, be automated by using macros. This could

– allow us to complete each job more quickly

– help us to produce a more accurate and consistent end result

– allow us to spend more time doing the interesting things – i.e. engaging with the text – and less time doing the boring repetitive jobs.

### Who should use macros?

Generally, macros are of most use to those who do on-screen editing, but anyone who has to edit Word files – for whatever reason – could benefit from using macros.

Proofreaders too can benefit greatly from using some of the macros, as explained in the section ‘My six favourite macros (as a proofreader)’.

### But I’m not a programmer!

That’s not a problem. You don’t need to be a programmer. This book offers a huge range of different macros – written specifically for editors – so to get started, you don’t need to learn how to program your own. As you gain confidence, you can start by making changes to my macros – Jack Lyon’s *Macro Cookbook* (ISBN: 9781434103321) is essential reading for that.

But to use the macros in this book, you only need to learn how to load macros into Word – full instructions are given below. Once the macros are loaded, you just use them.

### How do I run a macro?

There are three ways in which a macro can be run:

a. Open the Macros menu and select it from the list of macros, and click Run.

b. Add an icon to the toolbars at the top of the Word screen – one icon for each macro.

c. Use keyboard shortcuts – one for each macro.

How to set up (b) and (c) is explained below.

### What jobs can macros do?

That’s a bit like asking, ‘What jobs can woodworking tools do?’ The answer is that there are many different tools and they do many very different jobs. It takes time and effort to learn how to use the different woodworking tools, and so it is with macros. I just hope that you find it as enjoyable, profitable and satisfying as I do.

### But aren’t macros dangerous?

Yes, they are very dangerous! All carpenters know that circular saws and other power tools are *extremely* dangerous. They have to use the right tool for any particular job, and they have to use it in the right way. If they spoil a piece of wood by using a power tool, it’s not the fault of the power tool! But with experience, you will be able to use macros more and more effectively.

If you misuse macros, you can produce poor quality text, but macros can’t damage your computer in any way.

Also, you should know that the macros cannot attach themselves to your clients’ files. Your clients will not know whether you did the job entirely by hand, or whether you were ‘macro-assisted’.

# Introduction to macros

*(* ***My first macro – Part 1*** *https://youtu.be/hi4QCQy1QWg and* ***Part 2****: https://youtu.be/KFOVs3qBomY )*

To use the macros in this book you do need to know what they look like because, to use a macro, you first have to electronically copy it (Ctrl-C) out of this book and then paste it (Ctrl-V) into your computer.

Here’s a very simple macro – it’s just a list of instructions plus a beginning and an end:

**Sub\_Swap Characters()**

Selection.MoveEnd 1

Selection.Cut

Selection.MoveLeft 1

Selection.Paste

End Sub

I have highlighted the beginning and the end of the macro in turquoise – the word ‘Sub’ is short for subroutine, which is another name for a macro or computer program. Each macro has to have a unique name, highlighted here in green. (You must ***not*** have two macros with the same name; if you do, Word will generate the error, say, ‘Ambiguous name detected: FRedit’. If so, that means you’ve got two copies of, in this case, FRedit, and you need to delete one of them.)

These macros are created in a computer language called Visual Basic for Applications (VBA), and they can be run from within Word. There are VBA macros that work with other Microsoft applications such as MS Excel, but we can ignore those here. To us, a ‘macro’ means an MS Word macro.

To take advantage of what the macros in this book can do, you need to know where to store them (within the VBA application) and how to run them from within Word.

## Storing your macros

Macros can be stored in various places in your computer, but the simplest place is within Word’s Normal template. This is the most convenient place because they then are available for use with any file(s) that you are working on.

(You might hear people saying that it is dangerous to store macros in the Normal template. It is true that there ***were*** once problems with doing so, but that was back in the days of Word 97 and 2003. As far as I’m aware, this hasn’t been a problem since Word 2007 onwards.)

The macros are stored inside the Normal template, one after the other, in a Visual Basic file called *Normal.New\_Macros*. Here’s part of my *Normal.NewMacros* file to show you what the macros look like when viewed in VBA (please don’t worry about the content of these macros, or what they do – just note the way that they are stored):

........................................................................................................................................................................

Sub\_SubscriptSwitch()

' F4

Selection.Font.Subscript = Not Selection.Font.Subscript

End Sub

........................................................................................................................................................................

Sub\_Superscript\_Switch()

' F5

Selection.Font.Superscript = Not Selection.Font.Superscript

End Sub

........................................................................................................................................................................

Sub\_DeleteIt()

' Version

Selection.Delete

End Sub

........................................................................................................................................................................

Sub\_Mu()

' Version

Selection.TypeText Text:=ChrW(956)

End Sub

........................................................................................................................................................................

Sub\_FontRemove()

' Version

On Error GoTo ReportIt

Selection.Font.Reset

Exit Sub

ReportIt:

beep

End Sub

........................................................................................................................................................................

There are just three important things to understand here:

• The macros are ***all stored together in a single file***, but they don’t have to be in any particular order – they are run from Word by name.

• It’s a single file, so ***you can select all the macros (Ctrl-A) and copy them (Ctrl-C)***. You can then paste them somewhere else, perhaps in a Word file, as a way of keeping a backup copy.

• ***When adding or removing macros be very careful not to break that*** ***repeated*** ***pattern of*** Sub\_ ... End Sub\_which I’ve again highlighted to make it stand out.

**Tip**: Below, I explain about how to add a macro, but rather than just reading it in theory, why not choose a particular macro and actually install it. You could try the transpose characters macro that I gave as an example above.

Sub\_TransposeChars()

Selection.MoveRight 1, Extend:=wdExtend

Selection.Cut

Selection.MoveLeft 1

Selection.Paste

End Sub

It transposes adjacent characters, say from ‘Pual’ to ‘Paul’ – you just put the cursor between the ‘u’ and the ‘a’ and run the macro.

## Adding macros

Macros can be added simply by copying them from this book and pasting them into *Normal.NewMacros*. So, in this book, make sure that you select the complete macro, from

Sub\_Something( )

to

End Sub

and press Ctrl-C to copy it. Then run VBA (see below), decide where to put the new macro and press Ctrl-V to paste it in. I tend to put new macros down at the bottom of the file, but it really doesn’t matter because Word calls them by name.

(N.B. This book is arranged as two files: this file has the descriptions of the macros, and the other file, ‘TheMacros’, has the actual macro listings.)

The difficult thing is knowing how to open *Normal.NewMacros* in VBA – it is different on different computer systems!

## Installing a macro from scratch

*(See also video: My First Macro – Part 1 (6:04): https://youtu.be/hi4QCQy1QWg)*

*(See also video: My First Macro – Part 2 (5:45): https://youtu.be/KFOVs3qBomY)*

*(See also video: Macro Starter Pack (5:42): https://youtu.be/IeMnmtJT2Ys)*

Macros can be added simply by copying them from an electronic book, from a website or from an email, and pasting them into a program called Visual Basic for Applications (VBA), where they will be stored in the ‘Normal template’, as it’s called. I’ll try to explain in a number of small steps.

**Step 1: Copy the macro**

Wherever the macro comes from, you first have to select it and then do a Ctrl-C to copy it. (*On a* *Mac, that’s Command-C,* ⌘*-C.*)

However, you do need to make very sure that you select the *complete* macro, i.e. from the

Sub\_SomethingOrOther( )

down to and including the

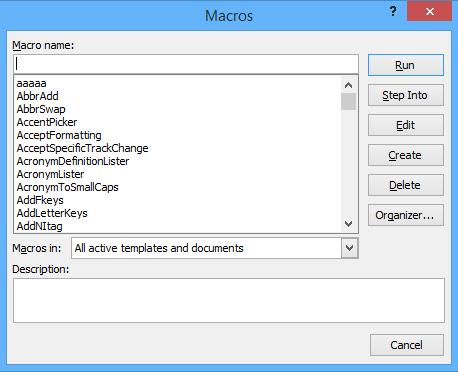
End Sub

before you press Ctrl-C (*Mac:* ⌘*-C*) to copy it.

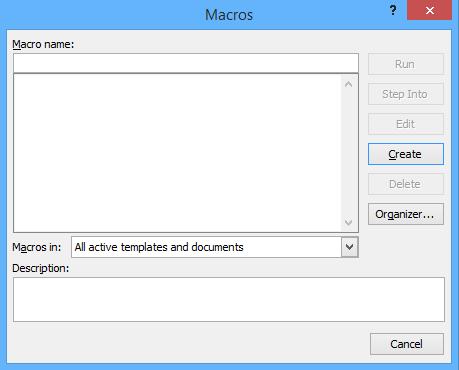
**Step 2: Open VBA**

VBA is a separate application that works alongside Word. The computer programs in VBA are called macros. These macros can be used from within Word without VBA actually being on screen. However, to install your macros in the first place, you have to open VBA, as follows:

Click on Alt-F8 (*Option-F8 on a Mac*), and it should open the Macros window. (If not, on Word 2003/4 you can use the menu: Tools–Macro–Macros, or on 2007 onwards View–Macros.)



In the middle of this window is a list of all the macros that are currently installed in your computer*.* But of course, if no-one has yet put any macros in your computer, the list will have no items in it:



Now, in the top box (Macro name:) type the single word Dummy, and click the Create icon (fourth down on the right). VBA will now open, showing:

Sub\_Dummy()

'

' Dummy Macro

'

'

End Sub

**Step 3: Paste in your new macro**

Select the whole of the Dummy Macro, from the Sub\_Dummy() line up to and including the End Sub\_line, and then click Ctrl-V (*Mac:* ⌘*-V*). This will put your new macro in place of the Dummy macro.

**Step 4: Close VBA**

You can usually do this with Alt-Q (⌘*-Q*), but you can also do it by clicking in the top right ‘X’ (Close) icon – but notice that there’s another ‘X’ just below it, so click in the very top icon.

**Step 5: Running your new macro**

To run your new macro, use Alt-F8 (*Option-F8*), to open the Macros window again (or Tools–Macro–Macros or View–Macros). Look in the list of macros for the one you want, click on it and then click Run (top right button).

Once you are familiar with this, there’s a simpler way to add a macro: Press Alt-F8 to open the Macros window, click on any one of the macro names, click on the Edit button, then you can paste in the new macro. However, you must be very careful where you put the new macro! You must be careful not paste the new macro inside one of the existing macros; it must go ***after*** and End Sub, and ***before*** the next Sub\_SomethingOrOther. Probably the safest is to use Ctrl-End to go to the very end of the VBA macros, and paste the new macro in there.

## Running the macros

There are basically three ways you can run macros:

1. from the Macros dialogue box

2. by adding an icon to the toolbar at the top of the Word screen

3. by pressing a particular key combination.

I use the dialogue box for those macros that I use very rarely, but I never use icons (2). I run 99% of my macros from keystrokes because it’s so much faster than using icons. Once you get more than a small number of icons for macros, it just becomes impractical to use icons.

“But I can’t remember keystrokes!” OK, let me ask you a question: do you have to remember where the gears are in your car? If you do something often enough, it becomes automatic. What’s more, there’s a pattern to the gears, which helps. So I suggest that you make a ‘pattern’ for your keystrokes. Use keys that have some significance to you, and/or use various key combinations with the F-keys, and put a strip of card with the macro names written on it. For very frequently used keystrokes, I suggest that you use a key combination that you can press just with your left hand, and/or keys on the numeric keypad with your right hand – the numeric pad means that it’s less far for you to move your hand away from the mouse.

**Tip**: Use the *CustomKeys* macro to call up the Customize Keyboard dialogue box so that it’s quick and easy to change your keystroke allocations. Then the trick is as follows: When you try to use a macro that you don’t often use, press the key combination that you *think* it might be. Then if that’s *not* the right one, change the keystroke allocation to that keystroke. The point is that, for you, that is a more intuitive choice of key combination.

## Allocating a keystroke (Word 2013 and 2010)

*(Video: https://youtu.be/XXs6z-QhzPw)*

1. Right-click on a blank part of the ribbon and click on ‘Customize the Ribbon’.

2. Underneath the left-hand column, below the scrollable window, it says ‘Keyboard shortcuts:’ Click the ‘Customize’ button next to it.

3. In the Customize Keyboard window that appears, in the left-hand list (Categories), find ‘Macros’ (pressing ‘m’ on the keyboard, twice, will get you there quickly).

4. The right-hand list becomes (not surprisingly) ‘Macros’. Select your chosen macro name.

5. Click in ‘Press new shortcut key’ and do just that: press the keyboard shortcut that you want to associate with this macro.

6. To the left of that box is a ‘Current keys’ box. This box shows whether that macro already has a keystroke assigned to it. Also, immediately under that box is a line telling you whether the keystroke you pressed is currently assigned to something else – another macro or a Word command or a special character – or whether it is currently ‘Unassigned’.

7. If you’re happy that you want this keystroke to be uniquely linked to your selected macro then click the ‘Assign’ button.

## Allocating a keystroke (Word 2011 – Mac)

1. On the Tools menu, click ‘Customize Keyboard’.

2. In the Customize Keyboard window that appears, in the left-hand list (Categories), find ‘Macros’ (pressing ‘m’ on the keyboard, twice, will get you there quickly).

3. The right-hand list becomes (not surprisingly) ‘Macros’. Select your chosen macro name.

4. Click in ‘Press new shortcut key’ and do just that: press the keyboard shortcut that you want to associate with this macro.

5. Above that box is a ‘Current keys’ box. This box shows whether that macro already has a keystroke assigned to it. Also, immediately under the ‘new keyboard shortcut’ box is a line telling you whether the keystroke you pressed is currently assigned to something else - another macro or a Word command or a special character – or whether it is currently ‘Unassigned’.

6. If you’re happy that you want this keystroke to be uniquely linked to your selected macro then click the ‘Assign’ button.

## Allocating a keystroke (Word 2007)

1. Click the ‘Customize Quick Access Toolbar’ menu – the little down-arrow at the right-hand end of the Quick Access Toolbar (QAT).

2. Choose ‘More Commands’.

3. At ‘Keyboard Shortcuts’ at the bottom of the box, click ‘Customize’.

4. In the left-hand list (Categories), select ‘Macros’ (pressing ‘m’ on the keyboard, twice, will get you there quickly).

5. Click in ‘Press new shortcut key’ and do just that: press the keyboard shortcut that you want to associate with this macro.

6. To the left of that box is a ‘Current keys’ box. This will show whether that macro already has a keystroke assigned to it. Also, immediately under that box is a line telling you whether the keystroke you selected is currently assigned to something else – another macro or a Word command or a special character – or whether it is ‘Unassigned’.

7. If you’re happy that you want this keystroke to be uniquely linked to your selected macro then click the ‘Assign’ button.

8. Then ‘Close’, and your keystroke is ready to use.

## Allocating a keystroke (Word 2002/3)

1. Open the ‘Tools–Customize’ tab.

2. Click ‘Keyboard’.

3. Then, in the left-hand list (Categories), select ‘Macros’ (pressing ‘m’ on the keyboard, twice, will get you there quickly).

4. The right-hand list becomes (not surprisingly) ‘Macros’.

5. Select the macro name.

6. Click in the ‘Press new shortcut key’ box.

7. Press the keystroke you want to use.

8. Just below that box, a line will appear saying ‘Currently assigned to:’ and, hopefully, ‘[unassigned]’.

9. If it is already assigned to another function within Word, you’ll have to decide if it is a function that you would want to use via a keystroke and, if so, choose a different keystroke.

10. Click ‘Assign’.

11. Then ‘Close’, and your keystroke is ready to use.

### **Tip** – using the Customize keyboard dialog

### This dialogue box is also useful where

a. you can’t remember which keystroke you have used for a given macro

b. you can’t remember the macro name for a keystroke that you already use.

For (b), just click in the ‘Press new shortcut key’ box, press the relevant keystroke and look in the ‘Currently assigned to:’ line.

## Adding icons (Word 2007/2010)

1. Right-click on a blank space on the screen’s toolbar. This brings up the Quick Access Toolbar (QAT)

2. From the QAT, click the ‘Customize Quick Access Toolbar’ menu.

3. Choose ‘More Commands’.

4. (In Word 2010 only, you also now need to click on ‘Customize Ribbon’.)

5. In the ‘Choose Commands From’ list, select ‘Macros’.

6. Select ‘Normal.NewMacros.<Macro\_Name>’ from the list below.

7. Click ‘Add’.

8. The macro will appear at the bottom of the QAT list on the right-hand side.

9. Click ‘Modify’ (under the QAT list).

10. Choose a symbol for your macro.

11. In ‘Display Name’, shorten ‘Normal.NewMacros.MyNewMacro’ down to ‘MacroName’.

12. Click OK.

13. Use the up/down arrows on the left of the QAT list to move your MacroName symbol to the desired location on the QAT.

14. Click OK.

## Adding icons (Word 2002/3)

1. Open the ‘Tools–>Customise’ tab.

2. Select the ‘Commands’ tab.

3. In the left-hand list (Categories), select ‘Macros’.

4. In the right-hand list find ‘Normal.NewMacros.<MacroName>’.

5. Drag it up to the toolbar.

6. The cursor will have an ‘x’ in it, but it will turn into a ‘+’ when you are over a bit of the bar where you are permitted to drop it.

If you want to customise the appearance of the macro icon, do the following:

1. With the ‘Customize’ box open, right-click on your macro.

2. Click ‘Default Style’.

3. Right-click again.

4. Choose either an existing icon from ‘Change Button Image’.

5. Or create your own button from ‘Edit Button Image’.

## Updating macros

If you have a macro that you are already using, and you hear that there’s a more up-to-date version, how do you make the upgrade? The important thing to remember is ***not*** to delete the whole of the macro, from

Sub\_Something( )

to

End Sub

If you just delete the old version of the macro, the associated icon and/or keystroke will be lost and you will have to set it up again.

Instead, just delete the ‘meat’ of the macro, leaving the Sub\_and End Sub\_lines, for example:

Sub\_Citehecker( )

End Sub

Then copy the ‘meat’ of the new macro, i.e. not the Sub\_and End Sub\_lines, and paste it in the space that you have left for it. (Don’t worry about there being extra blank lines – they are totally irrelevant to the working of the macro.)

## What happens when things go wrong?

*(Video: https://youtu.be/AY6B-IkLEN8)*

First, I’ll give you a general description of what to do to report an error to me, and then I’ll list a few errors that sometimes occur, giving you a suggestion of the possible cause.

### A general suggestion

If you get error with some of the macros – especially all the ...***Alyse*** macros – it may be worth creating a text-only version of your file and then running the macro on that copy.

One way is to use [Ctrl-A, Ctrl-C, create a new file, Paste as Pure Text], but to be sure you’re getting **all** the text – including what’s in the foot/endnotes and text boxes – you can use *CopyTextWithSomeFeatures*.

### How to respond to – and report to me – an error

Sometimes, when you try to run a macro, it generates an error, and Visual Basic (VBA) asks you what you want to do, offering you:

**End, Debug, Help.**

Ironically, the least helpful of these is to click ‘Help’. Don’t bother.

If you just want to give up altogether and ignore the idea of using the macro, click on End.

To find out what went wrong – perhaps so that you can report the error to me – the first thing to do is to make a note of how VBA describes the error. Here’s an example:

*Runtime error ‘5174’:*

*This file could not be found.*

MS Word won’t let you copy and paste the error message, but you could perhaps go over to your email software and start to compose me an email, typing in this error message.

Next, click on Debug. Debugging is a technique that programmers use to try to work out what has gone wrong with a program. This will take you into VBA with one of the lines of the macro highlighted in yellow, maybe looking something like this:

If gottaList = False Then

Documents.Open dirName & listName

Else

listDoc.Activate

End If

Make a note of the line so that you can report it to me. However, this time, you *can* do it by selecting a bit of the macro, either side of the yellow line, copying it, and then pasting it into a Word file (or your email), where it will appear as ordinary text. But please tell me exactly which line was actually highlighted in yellow – this is important if you want me to correct the problem.

Next, you have to stop the debugging process, or ‘reset’ VBA. You do this by clicking the Reset button on VBA’s top tool bar. Look for the set of three icons – as on an AV device: Play, Pause and Stop. The ‘Reset’ button is the square block, as used for ‘Stop’ on an AV player.

Send me that information, and I’ll see what I can work out.

(If you **don’t** stop the debugging process and simply go back into Word, all will **seem** to be OK. However, when you later try to run another macro, it will generate the error: ‘Can’t execute in break mode’. You then have to click ‘OK’, select the VBA window and click the ‘Reset’ icon, as mentioned above.)

### Some possible errors and their possible cause

**“Variable not defined”** – Search in VBA amongst your macros, and see if there’s a line saying Option Explicit. If so, put an apostrophe in front of it, to disable it. This won’t harm the operation of any of the other macros.

**“The Find What text contains a Pattern Match expression which is not valid.”** – There are various reasons for this. In general, just report it to me, as above, but if you’re using any of the ...Alyse macros, such as DocAlyse, then it’s likely to be a problem with what’s called the ‘list separator’ used in the operating system of your computer. This is especially likely if you’ve got a computer set up for mainland Europe. Here are my standard instructions:

The ‘list separator’ used within Word needs to be a comma, not a semicolon.

However, this is not a Word option, rather it’s an operating system option.

So, on Windows 7, 8.1 and 10, it is in the Control Panel under ‘Clock Language and Region’ and then ‘Region’ and then ‘Additional settings’ (which is a button near the bottom of the Region window). In Additional settings, the fourth from the bottom is ‘List separator’. Change it to a comma and click OK.

**“Compile error:** **procedure too long”** – Let me guess... you’re trying to use DocAlyse, right? And you’re using a Mac? If you get this error, try using the macro, DocAlyseForMac. Even if that gives the same error, there is a solution; please email me and ask for a copy of DocAlyseForMac\_2014.

# My twelve favourite macros (as an editor)

Here’s a list of the twelve Word tools (macros or groups of macros) that save me most time as an editor and enable me to produce a better quality of work. However, all editors work in different ways, so there may be other different macros that you find more useful than these. The aim of this list is just to give you a feel of the sort of macros that are available.

1) ***FRedit*** is the biggest timesaver. Unfortunately, it uses a concept that is new to many editors: scripted find and replace. It sounds complicated, but it isn’t. However, within this book I have only provided a brief introduction to the concept, because *FRedit* has its own set of instructions, plus a library of tools for you to use for a range of different jobs. (http://www.archivepub.co.uk/documents/FRedit.zip)

2) ***HyphenAlyse*** and ***DocAlyse*** give me valuable information to help me to prepare my stylesheet for a job. They tell me what conventions the author has used (more or less consistently). This information helps me to decide what conventions to use for punctuation and spelling etc. Because I do this *before* I start reading, it saves me a lot of time.

3) ***SpellingErrorLister*** produces an alphabetic list of all the different words in the document that Word’s spelling checker ***thinks*** are spelling errors. You can decide which are or are not spelling errors. You can then use *SpellingErrorHighlighter* to highlight some of the words for your attention as you edit, or it can change the spelling errors for you automatically.

If I also run ***ProperNounAlyse***, the computer will produce a list of pairs of proper nouns that look as if they might be variant spellings of one another, e.g. Beverly/Beverley.

4) ***IStoIZ*** and ***IZtoIS*** change and/or highlight all the words in a file that need switching to whichever convention your client wants. (This is only applies to English language documents.)

5) **Highlighting** macros – There are several macros for applying highlights of different colours, (selectively) removing highlights, and searching for text that is highlighted in different colours.

6) ***InstantFindDown(Up)*** – If you want to look at the previous or next occurrence of a word or phrase, InstantFind will take you straight to it – with one single click. The macro also loads this word/phrase into the Find box, so that you can use Word’s own Ctrl-PageUp and Ctrl-PageDown to go through the various occurrences of this text. And the other very powerful find macro (***FindSamePlace***) is where you want to compare the text in two documents. You select some text in one document, and the macro switches to the other file, goes up to the top of the document and finds the first occurrence of this text.

7) **Text editing** macros – This refers to macros for various text editing actions, as you actually read the text. For example, one macro will change the next number from numerals into words (and another one changes words to numerals). There are dozens of the text editing macros, so decide which editing actions you use most often, and find a macro for each of them. You’ll find them in the section: ‘Editing: Text Change’.

8) **Scripted word switching** – *MultiSwitch*, *WordSwitch* and *CharacterSwitch* are three very powerful and, more importantly, flexible ways of editing the text. I won’t bother explaining here; just have a look at the three sections following the heading: ‘Common Word/Phrase Switch’.

9) ***CitationLister* and *CitationListChecker*** – With this pair of macros, I first create a list of all the citations of references that occur in the text, and then the second macro tries to pair up the citations with the references within the list. I can then see if there are any citations that don’t have a corresponding reference in the list, or any references in the list that are not cited in the text. (Often the reference/citation ***is*** there, but there’s a spelling error or a mistake in the date etc.)

10) ***CommentAddMenu* and *CommentCopier*** – Select some text, and *CommentAdd* copies it, creates a new comment for an author query, adds ‘AQ:’ and pastes the text inside quotes, ready for you to type in your query. Or *CommentAddMenu* does the same sort of thing, but offers you a menu of different standard comments you might want to use (you can obviously edit this menu of comments according to your own style). Then CommentCopier copies all the comments in the file, puts them into a separate file and adds an ‘Answer:’ line in between each query and the next, ready for the author to type in a response. It also creates a ‘Context’ file, a compilation of all paragraphs that contain one or more comments.

11) ***WhatChar*** – For example, you come to something that *looks* like a degree symbol, but you suspect that it might not be. *WhatChar* checks the ANSI code (a degree is 176), but it also spells out in words what the character actually is. So, for example, it tells you what each of the following, highly confusable, characters (printed here in Century Gothic, to illustrate the problem) are: l|I1°ºvbvb. They are: lowercase letter-L, vertical bar, uppercase letter-I and the number one, then a proper degree symbol, a masculine ordinal (as used in Nº) and a superscripted lowercase letter-O.

12) ***CountPhrase*** allows you to select a word or phrase and it tells you how often this occurs in the text. This helps you to maintain consistency because, for example, you can very quickly check if something is spelt in either of two variant ways. But it also does both case-sensitive and case-insensitive counts, so you can see if it is capitalised differently in different parts of the document. (Also, the macro, ***HyphenSpaceWordCount***, counts the number of occurrences of, say, cow-bell, cowbell and cow bell.)

# My six favourite macros (as a proofreader)

‘*But I’m not an editor – I just do proofreading*’, you say. Nevertheless, you too can gain both speed and consistency through the use of certain of the macros in this book. Personally, I would *never* accept a proofreading job without also being given the text in electronic format (most commonly in PDF format).

To gain advantage from macros, you first need to copy and paste the text out of the PDF file(s) and into Word. You can, of course, search for things in PDF files, but once the text is in a Word file, you can use the following macros:

1) ***HyphenAlyse*** and ***DocAlyse*** give me valuable information to help me to prepare my stylesheet for a job. They tell me what conventions the author has used (more or less consistently). This information helps me to decide what conventions to use for punctuation and spelling etc. Because I do this *before* I start reading, it saves me a lot of time.

2) ***SpellingErrorLister*** produces an alphabetic list of all the different words in the document that Word’s spelling checker ***thinks*** are spelling errors. You can decide which are or are not spelling errors. You can then use *SpellingErrorHighlighter* to highlight some of the words for your attention as you edit, or it can change the spelling errors for you automatically.

If I also run ***ProperNounAlyse***, the computer will produce a list of pairs of proper nouns that look as if they might be variant spellings of one another, e.g. Beverly/Beverley.

3) ***IStoIZ*** and ***IZtoIS*** changes and/or highlights all the words in a file that need switching to whichever convention your client wants. (This is only applies to English language documents.)

4) ***WhatChar*** – For example, you come to something that *looks* like a degree symbol, but you suspect that it might not be. *WhatChar* checks the ANSI code (a degree is 176), but it also spells out in words what the character actually is. So, for example, it tells you what each of the following, highly confusable, characters (printed here in Century Gothic, to illustrate the problem) are: l|I1°º. They are: lowercase letter-L, vertical bar, uppercase letter-I and the number one, then a proper degree symbol, a masculine ordinal (as used in Nº) and a superscripted lowercase letter-O.

5) ***CountPhrase*** allows you to select a word or phrase and it tells you how often this occurs in the text. This helps you to maintain consistency because, for example, you can very quickly check if something is spelt in either of two variant ways. But it also does both case-sensitive and case-insensitive counts, so you can see if it is capitalised differently in different parts of the document. (Also, the macro, ***HyphenSpaceWordCount***, counts the number of occurrences of, say, cow-bell, cowbell and cow bell.)

6) ***InstantFindDown(Up)*** – If you want to look at the previous or next occurrence of a word or phrase, InstantFind will take you straight to it – with one single click. The macro also loads this word/phrase into the Find box, so that you can use Word’s own Ctrl-PageUp and Ctrl-PageDown to go through the various occurrences of this text.

# Preparing for a (book) job – proofreading

Here’s what I do as I start a new book job, if I’m proofreading – I’ve recorded it here just as a suggestion as to what I find useful.

|  |  |
| --- | --- |
| **Action** | **Result** |
| Read the brief and/or style guide (if provided) and fill in as much as possible of the stylesheet | Some items decided on stylesheet (see Appendix 7) |
| If the book is in separate files, create an AllWords file using ***MultiFile\_Text*** | All the words (inc. footnotes and text from textboxes), but no images, in one file |
| Run ***DocAlyse*** | Stylesheet with more decisions, including some items in the word list, e.g. co(-)operate, learn(t/ed) etc |
| If no decision on UK/US English, run ***UKUScount*** | The numbers of UK and US English words, and hence a language decision |
| If no decision on is/iz, run ***IZIScount*** | The numbers of -is- and -iz- words used, and hence an is/iz decision |
| Run ***SpellingErrorLister*** and ***SpellingErrorHighlighter*** | Actual spelling errors highlighted; or a list of spelling errors and corrected words for use with *FRedit* |
| Run ***HyphenAlyse*** | Frequencies of all hyphenated words and of all words with certain prefixes (anti-, non-, post-, pre- etc) |
| Run ***ProperNounAlyse*** | A list of possibly misspelt proper nouns, including frequencies |
| For academic jobs, run ***CitationLister*** and***CitationListChecker*** | List of referencing problems |

# Preparing for a (book) job – editing

*(See also video: Book editing using macros (13:53): https://youtu.be/WSfXidPGC1A)*

Here’s what I do as I start a new book job, if I’m editing. This is clearly a lot more complicated and detailed. I keep trying to refine this ‘recipe’ each time I edit a job, but it’s far from perfect.

|  |  |
| --- | --- |
| **Action** | **Result** |
| Read the brief and/or style guide (if provided) and fill in as much as possible of the stylesheet | Some items decided on stylesheet (see Appendix 5) |
| If the book is in separate files, create an AllWords file using *MultiFile\_Text* | All the words (inc. footnotes and text from textboxes), but no images, in one file |
| Run *DocAlyse* | Stylesheet with more decisions, including some items in the word list, e.g. co(-)operate, learn(t/ed) etc |
| If no decision on UK/US English, run *UKUScount* | The numbers of UK and US English words, and hence a language decision |
| If no decision on is/iz, run *IZIScount* | The numbers of -is- and -iz- words used, and hence an is/iz decision |
| Run *SpellingErrorLister* | List of apparent spelling errors |
| Read through spelling error list and use *SpellingSuggest* to add alternates, e.g. mesage|message, but colour or highlight words that need checking when I do the actual read through  Copy and paste any ‘suspect items’ in the proper noun section of the error list into a separate document for later use, e.g.  Macmullan  MacMullan | Spelling errors needing to be changed  Spelling queries to be highlighted  List of a few possible proper noun errors |
| Copy spelling error list to the end of the *FRedit* list and run *FReditListProcess* once with words in the spelling error list (any case) and once with proper nouns (case sensitive) | *FRedit* items for spelling |
| Run *HyphenAlyse* | Frequencies of all hyphenated words and of all words with certain prefixes (anti-, non-, post-, pre- etc) |
| Use *HyphenationToFRedit* to add items to the *FRedit* list that will correct hyphenation (and remember to record hyphenation decisions in the Words List at the end of the style sheet) | *FRedit* items to correct hyphenation  Updated Words List |
| Run *ProperNounAlyse* | A list of possibly mis-spelt proper nouns, including frequencies |
| Check through proper noun list and try to resolve any conflicts with names of different spelling, using *InstantFindUp* or *FindSamePlace* to jump around and look at the context, and/or *GoogleFetch* to check names on the internet | Items added to *FRedit* list to correct (or highlight, if not sure) proper noun errors |
| If not sure on some names, query with author and get them correct, and *only then*... | More items for *FRedit* list |
| For academic jobs, run *CitationLister* and *CitationListChecker* to check the references and... | List of referencing issues to check via the internet or query with the author |
| ...use *AuthorDateFormatter* to sort formatting of names, initials and dates | Improved formatting of references list |

# What can macros do for you?

There are very many different things that macros can do, so I have divided them up into sections to try to make it easier for you to find the macro(s) that you want for any given job.

### Textual analysis – preparing your stylesheet

There are only a few macros under this heading, but they are some of the most powerful, for both proofreaders and editors. Their purpose is to help you to assess the script before starting work on it. The aim is to help you make decisions about spelling, hyphenation, punctuation styles etc *before* you start to read. This can save you a lot of time. (Editors may like to run some of these macros again on the finished files to pick up any remaining inconsistencies.)

### Pre-editing tools

If you are editing a text, there can be a lot of changes to be made to the file before you actually start reading, and many of these involve repetitive tasks – just the sort of thing that computers are good at. The most powerful tool here, *FRedit*, provides ‘scripted find and replace’, a concept that is new to some editors, for which there’s only a brief introduction in this book. *FRedit* has its own, separate documentation. This macro can be very useful even if is used very simply, but it can also do some extremely time-saving tasks if you are willing to learn to use its more powerful aspects. The *FRedit* package comes with a library of tools that other people have developed. This is especially helpful because many of these special tools use wildcard find and replace.

Other macros in this section do various editing jobs on: tables, frames, textboxes, footnotes and endnotes, bookmarks, comments and styles. For example, there are macros that pull all the tables and/or figures out into a separate file, and a macro that creates a list of all the acronyms in a file etc, etc.

### Editing: text change

As you are reading through the text, you do lots of minor editing actions: adding a comma, hyphenating two words, switching the order of two words, changing numerals 1–9 (or 10) into words etc. Using these macros can speed up the editing process but, more importantly, they enable you to make those minor changes without taking your attention off the meaning of the text that you are reading.

### Editing: information

These macros provide useful bits of information about the piece of text you are working on.

### Editing: highlighting

Coloured highlighting can provide another set of tools to aid the editor: you (or a macro) can use different colours to highlight different things. These macros allow you to add highlights of whatever colour, and then to move around the text, looking at the text in the different colours. Also, you can get rid of the highlighting, either in a given area of text, or selectively by colour; you can remove, say, all the green highlighting while leaving all the rest of the highlighting intact.

### Editing: navigation

When working with text, you want to be able to move around the text, quickly and easily, looking at various bits, checking them and changing them. So, by using macros, you can jump instantly to, say, another heading of the same type, to another occurrence of the selected text, to another comment, to the same place in a different file – plus a whole load of other ways of jumping around the text.

### Editing: comment handling

Word’s comment facility can be useful for making notes for yourself or others, and macros can help with adding comments. It can also collate the comments afterwards to pass on to the author or the typesetter or the client – especially useful if it’s a MultiFile\_ job.

### Other tools

This final section is just a miscellany of macros – ones that didn’t fit into any other category.

## Macro Menu – an Overview of Available Macros

The aim of this section is to whet your appetite by offering a macro smorgasbord. The problem with this book is that there are far too many macros, so how do you know what’s available that you might find useful for your own particular style of editing? Here I take different topics and simply state what’s available related to that topic (i.e. some macros appear in multiple sections below). If something catches your fancy, place your cursor in the macro name (in italic) and *InstantFindDown* to search for the details.

# Textual Analysis

(Preparing your brief or style guide)

When you are assessing a script ready to start editing or proofreading, there are decisions to be made about spelling, hyphenation, punctuation styles etc. Some of these *might* have been specified for you by the client, but in any case there will be others where it is best to find out what the author has done most often (though not consistently!) and run with that. I wrote these macros having had the experience of making a style decision based on chapter 1 of a book, only to find that, in chapters 2 to 20, the exact opposite convention had been used!

Making as many style decisions *before* starting to read can potentially save you a lot of time, so in this section we look at the main macros to help in that process.

The first macro, *DocAlyse* (**doc**ument an**alyse**), very quickly gives you a feel for a lot of the spelling and punctuation conventions.

Then there are a whole range of spelling analysis macros, including UK/US, -is-/-iz-, general spelling errors and variations in use of accents (see the overview in that subsection). Also included is *ProperNounAlyse* to list any similar-looking proper nouns (showing the frequency of each) so that you are alerted to possible misspellings.

Finally, there’s *HyphenAlyse*, which gives you a frequency list of all the words in the book that are (or might be) hyphenated.

If your book is in multiple files, you’ll also need to use the macro *MultiFile\_Text* to scrape the text out of all the files to present to the analysis macros.

## DocAlyse

The aim of this macro is to help you to assess a Word document by counting the number of times the author uses various spelling, punctuation and formatting conventions.

To do this, *DocAlyse* creates a copy of the currently open Word file and generates a list such as this:

(My explanatory comments are added in italic but it’s the macro that has made all the zero item non-bold .)

*(one to nine, 10 upwards or one to ten, 11 upwards)*

**ten 35**

**10 20**

*(commas in thousands)*

**nnnn 21**

**n,nnn 3**

n nnn –

**serial comma 2**

**no serial comma 30**

**single quote 90**

**double quote 2**

**etc 19**

**etc. 1**

**et al 1**

**et al. 9**

et al (italic) –

**i.e. 2**

**ie 1**

**e.g. 1**

**eg 1**

*(different formats for initials)*

J. L. B. Matekoni –

J.L.B. Matekoni –

**J L B Matekoni 17**

**JLB Matekoni 9**

p/pp. 123 –

**p/pp.123 13**

**p/pp 123 1**

**p/pp123 4**

**UK spelling (approx.) 37**

**US spelling (approx.) 1**

**-is- (approx.) 102**

**-iz- (approx.) 4**

**data singular 2**

**data plural 2**

*(alternative past participle spelling)*

-rnt -elt –

**-rned -elled 10**

**fig 1**

**figure 8**

**Figure 1**

Chapter 0

**chapter 2**

**spaced units (3 mm) 3**

unspaced units (3mm) –

**focus... 6**

focuss... –

co-oper... –

**cooper... 2**

**diacritics 3**

N.B. The ‘JLB Matekoni’ option can get exaggerated in number because it will pick up things like: ‘the US Department of Energy’ and think that it’s a person with un-full-stopped initials.

I’ve added the wildcard Find sequence to some of the results, e.g. JLB Matekoni, so if you see only a couple of examples of one item and want to find them, you can copy and paste the relevant wildcard find into Word’s Find box.

If you don’t want these codes to show, then at the beginning of the macro, change the line to:

showWild = False

Thiers Halliwell has sent me a set of medical abbreviations to add to *DocAlyse*. There are rather a lot, and if you don’t want them, it’ll slow down the operation of the macro, so I’ve taken the unusual step of putting it as a separate piece of code, which medics will need to copy and paste into the middle of the macro in the space indicated.

Some extra items have been added to DocAlyse, as follows.

It now counts OK, ok, Ok, okay – OK?!

For percentages, it provides:

**unspaced, e.g. 9% 2**

**spaced, e.g. 9 % 3**

**9 per cent 5**

**9 percent 3**

**nine per cent 1**

**nine percent 1**

Then there’s edition/editor(s):

**ed 2**

**eds 3**

**edn 4**

**ed. 4**

**eds. 2**

**edn. 3**

For feet and inches (or minutes and seconds):

**feet (straight) 9' 3**

**inches (straight) 9" 2**

**single prime: 9′ 6**

**double prime: 9″ 4**

It already did a count of ‘proper’ ellipses against trios of full stops (periods), either spaced or unspaced, but I’ve now added (for proper ellipses only) a count of how they are spaced: before, after, both or neither.

This same count of spacing is also made for solidus (forward slash), em dash, en dash and hyphen, although I don’t, of course, count unspaced hyphens.

**Sub\_DocAlyse()**

N.B. This next ‘macro’ is not a macro in its own right; it’s the extra bit to be inserted inside *DocAlyse* proper at the place indicated:

**Sub\_DocAlyseMedBits()**

## Analyse Different Formatting of Centuries

*(Video: https://youtu.be/P-6VdmT2BbE)*

When I asked for ways in which DocAlyse could be extended, someone asked if it could count the different ways of formatting centuries, so I asked for a list of the different formats. This is what he sent me!

C19

C19th

C19th

Nineteenth Century

nineteenth century

19th Century

19th century

19th Century

19th century

XIXth Century

XIXth century

XIXth Century

XIXth century

To count all these formats would be far too difficult within DocAlyse, so I wrote a separate macro. It generates a table, like this:

|  |  |
| --- | --- |
| C19 | 2 |
| C19th | 1 |
| C19th | 0 |
| Nineteenth Century | 1 |
| nineteenth century | 1 |
| 19th Century | 3 |
| 19th century | 3 |
| 19th Century | 0 |
| 19th century | 0 |
| XIXth Century | 0 |
| XIXth century | 0 |
| XIXth Century | 0 |
| XIXth century | 0 |

**Sub\_CenturyAlyse()**

## Highlight Possible Errors with A/An

This macro tries to highlight all errors of the form: a orange, an pear, an university, a hour, a HTML, an UFO etc and also: a O, an P, a H, a S, an U.

Words like ‘university’ and ‘hour’ are simply exceptions, so I’ve dealt with them by listing them all at the beginning of the macro:

OKwithA = ",europe,european,once,one,uniform,uniformly,unified"

OKwithA = OKwithA & ",unique,uniquely,unit,unitarian,united,"

OKwithA = OKwithA & ",university,union,united,universe,"

OKwithA = OKwithA & ",universal,universally,unilateral,unilaterally,"

OKwithA = OKwithA & ",useful,usefully,useless,uselessly,user,"

OKwithA = OKwithA & ",usual,usually,,utility,utilities,utilitarian,"

OKwithA = OKwithA & ",utilization,utilisation,"

OKwithAn = ",hour,hourly,honest,honestly,honor,honour,honorary,"

OKwithAn = OKwithAn & ",honorarium,honorific,"

If you want to add others (e.g. if your client insists on ‘an hotel’!) then just add them to the list, being sure to keep the pattern of punctuation.

For acronyms, it highlights, say ‘a SME’, but with some acronyms, it depends whether you sound the letters or sound it as a word: ‘an RFI’ cf. ‘a ROM and a RAM’. So ‘a RFI’ is obviously an error, but ‘an RFI’, ‘an ROM’ and ‘an RAM’ ***might*** be wrong, so the macro highlights them in a less obvious colour (light grey).

**Sub\_AAnAlyse()**

## Reveal Formatting and Special Characters

When preparing to start a job, it can be helpful to get a feel of some of the formatting features that the author has used, and some of the special characters too. I tried to show some of these using *DocAlyse*, but it’s a bit of a blunt instrument, so this macro allows you to check the document more ‘surgically’.

It works by using highlighting to make the various features visible (so you might like to work on a copy of the file under test, rather than the original). It has a range of different features that you can highlight, and each time you run the macro, it removes all the existing highlighting and highlights the selected feature. Here’s the menu.

Once you’ve used one of these options to highlight something, you can use *FindHighlightDown* and *FindHighlightUp* to look through the text and see what’s what.

However, especially with the final four options, the macro might have highlighted things that you’re **not** interested in. If you know a little bit about macros, you might be able to tailor the macro accordingly (adjust to taste).

Another approach is to unhighlight just those characters, which then makes it easier to look more selectively through the remainder. For this, the *HighlightSame* macro is your friend, as follows.

You work your way through the highlights with *FindHighlightDown*, and then, when you decide one character is no longer of interest, run *HighlightSame*, and it will unhighlight all those characters throughout the text. (Make sure you use the 12/8/15 version or later because the newer version does unhighlighting as well as highlighting.)

Here’s some more detail on what the different options do. If there are other things that you want to highlight, please let me know.

## lling Analysis

**Overview** – There are a lot of spelling-related macros available, so I thought a quick overview would be helpful, to set the scene.

*UKUSCount* – Has the author predominantly used UK or US spelling?

*IZISCount* – For UK spelling, has the author predominantly used -is- or -iz- spellings?

(For editing, you can use *IStoIZ* and *IZtoIS* to implement your decision.)

*SpellingErrorLister* – Generates an alphabetic list of all the different spelling ‘errors’ (according to MS Word).

*SpellingErrorHighlighter* – Adds various highlights to words in the list that *are* (or could be) spelling errors.

(For editing, *SpellingSuggest*, *FReditCopy*, *FReditSame* and *FReditListProcess*, *SpellingListProcess* can

be useful*.*)

*ProperNounAlyse* – Alerts you to possible proper noun misspellings, showing their frequency.

(For editing, *FReditCopyPlus* can be useful*.*)

*HyphenAlyse* – Shows the frequency of word pairs in hyphenated, two-word and single-word form.

(For editing, *HyphenationToFRedit* can be useful*.*)

*WordPairAlyse* – Shows the frequency of word pairs that are never hyphenated (e.g. can not/cnnot).

*AccentAlyse* – Compares words that use the same letters, but with different accents.

*AccentedWordCollector* – Collects all the accented words in a text

*AAnAlyse* – Highlights a/an errors: a orange, an pear, an university, a hour, a HTML, an UFO, a O, an P, a H, an U.

*DocAlyse* – Counts past participles (-t/-ed), among(st), C/chapter, et al., i.e., etc, focus(s), benefit(t), Fig(ure), Eq(n).

Stretching the definitions of ‘spelling’ and ‘analysis’ a bit...

*AlphabeticOrderChecker* – Finds any suspicious non-alphabetism.

*AlphaOrderChecker* – Creates an alpha-sorted version of selected text showing changes

*AlphabeticOrderByLine* – Finds any suspicious non-alphabetism

*DuplicatedWordsHighlight* – Highlights things like ‘the the’ and ‘and and’ (easy to miss across two lines of text).

*DuplicatedWordsFind* – Jumps to the next duplicated word pair: ‘the the’ and ‘and and’ etc.

*ing* – Changes ‘splodge’ to ‘splodging’ or vice versa.

*DictionaryFetch, GoogleFetch, OUPFetch, PubMedFetch, ThesaurusFetch, WikiFetch* – Looks up the current word on the relevant website.

*LanguageSetUS*, *LanguageSetUK* – Sets language for whole document.

## Count IS/IZ spellings

This macro combines the basic ideas of the two *IStoIZ* and *IZtoIS* macros in order to count, fairly accurately, the numbers of -is- and -iz- type spellings. It can also, optionally, highlight them. You need to have the IS and IZ exceptions files set up in the same way as for *IStoIZ* and *IZtoIS* – for instructions see ‘IZ to IS Spelling and Vice Versa’.

**Sub\_IZIScount()**

## Count UK/US spellings

This macro counts how many words there are in UK spelling that are errors in US spelling and vice versa in order to give an indication of which spelling convention has mainly been used and how consistently.

If you want a quicker (less accurate) assessment, you can get it only to check those words that are equal to or longer than a certain number of letters. The example below of minimum word length, counts and timings for a 66,000-word book gives you an indication.

N.B. The macro will ***not*** count any words that have the strikethrough attribute applied, so you can ‘blank off’ the references list and any long quotations. (To strikethrough all quotations, you can use the *QuotationMarker* macro.)

*Chars* *UK* *US* *Mins*

3 23 52 4.8

4 23 48 4.8

5 18 48 3.6

6 17 47 2.8

7 17 43 2.3

8 17 32 1.9

**Sub\_UKUScount()**

## Highlight UK/US spellings

In a text that is set to UK spelling, this macro highlights all the ‘errors’ that are in fact just US spellings, according to Word’s spellchecker, and vice versa. So in a UK English file it will highlight ‘color’ and ‘center’, and in a US English file, it highlights ‘colour’ and ‘centre’.

However, MS Word’s spellchecker thinks that ‘practicing’ and ‘licencing’ are correct UK English spellings, so the macro also highlight those. If you know of any other words that MS Word gets wrong, please let me know and I’ll add them to the macro.

**Sub\_UKUShighlight()**

## Spellchecking for Proofreading and Editing

Whether you’re editing or proofreading, the first macro to use is *SpellingErrorLister*. For proofreading, you can then use *SpellingErrorHighlighter* to highlight all the spelling errors so that you can check that you haven’t missed any.

If you’re editing, you can use various tools to take the spelling error list and implement the necessary changes in the file(s) of the document you’re working on. The most obvious tool is *FRedit*, but there are a number of others that could prove useful. These are covered in the ‘Pre-editing Tools’ section of this book.

## Spellchecking for Proofreading

*(Video: https://youtu.be/6F\_yT1MIW\_Q)*

For proofreading, probably all you want to do is to ***highlight*** all the possible spelling errors. Then, after you’ve read the text, you can go back and compare with the highlighted file, to make sure that you haven’t missed any of the spelling errors (a salutary exercise, I find!).

Word’s spelling checker can put a wiggly line under all the ‘spelling errors’, but these are the ones that ***it*** thinks are errors. You know that, in your field of work, say ‘anticommutation’ and ‘bimagnon’ are perfectly acceptable, so you don’t want them highlighted; and equally, you don’t want lots of proper nouns highlighted. You can solve this problem by using macros: *SpellingErrorLister* and *SpellingErrorHighlighter*.

But before you can use macros to check the spelling, if your text is provided as a PDF file, you will first have to convert it into a Word file. There are many different ways of doing this (as a quick search on the web will reveal) but because it’s just the spelling we’re interested in, it may well be sufficient to simply copy the entire PDF (Ctrl-C) and paste it into a new Word file (Ctrl-V) – or maybe use PasteAsPureText.

**Listing the errors** – *SpellingErrorLister* creates a complete alphabetic list of all the different ‘spelling errors’ (according to Word’s spell-checker) that occur anywhere in a text.

(On a big file, this can take quite a few minutes, maybe 6–12 minutes for a 100,000-word book. The status bar should show you the progress, and if you do want to stop the macro running, you should be able to do so by pressing Ctrl-Break, and then select End, as opposed to Debug. However, Word does sometimes ignore Ctrl-Break on a hard-working macro. But I can almost guarantee that if you click hopefully on the screen, Word will crash! In other words **do not click on the screen**.)

(Unfortunately, my new Lenovo laptop doesn’t have a Break key, so there’s no way I can stop a macro mid-program. Argh! But it’s OK – I just use an external keyboard.)

The list that *SpellingErrorLister* creates might start something like this (a UK English file):

| Spelling Errors

acteylene

adjoint

analyze

analyzed

castilated

cill

clearnd

contractural

crainage

cranage

crosswall

crosswalls

develpments

Clearly, some of these *are* spelling errors (acteylene, analyze(d) etc), while others might be specialist words (adjoint, cranage) that are perfectly correct. Only you, the intelligent human, know which is which, so your job now is to highlight (in any colour you like – say green) the actual errors, or those that might be an error depending on the context (say light grey):

| Spelling Errors

acteylene

adjoint

analyze

analyzed

castilated

cill

clearnd

contractural

crainage

cranage

crosswall

crosswalls

develpments

**Highlighting the errors** – If you run *SpellingErrorHighlighter*, it will look for a list headed ‘| Spelling Errors’ and then work its way down the list, highlighting all those words in the main text, in the same colours that you have used.

In fact, *SpellingErrorLister* creates the list of ‘errors’ in two parts, the second one starting, for example:

Abrusci

Adlung

Agranovich

Altand

Altland

Ambrosch

Appl

ASI

Athanasopoulos

Azumi

Baldo

Bao

BARFORD

Barford

Bässler

These are probably proper nouns and won’t need highlighting, so listing them separately means that you don’t need to look quite so intently through them, when trying to spot the words that ***are*** spelling errors.

N.B. You can use this macro on any Word file, and it will, in fact (a) check and highlight all the text including footnotes and endnotes – but not textboxes – and (b) ignore (i.e. not highlight) any text, such as reference lists, that are struck through, ~~like this~~.

**Highlighting errors in textbox text** – If highlighting spelling errors in textboxes is important, then you can (a) first use *BoxTextIntoBody* to copy the textbox text into the main body of the text, (b) first use *MultiFile\_Text* – you just give it a ‘list’ of the single file you want to work on, or (c) do the highlighting by using *FRedit* – simply put ‘| Textboxes = yes’ at the beginning of the *FRedit* list.

**Technical details** – Some PDF to Word conversions will give you: Bässler, Brédas, Itô etc. The macro will correct these to: Bässler, Brédas, Itô. This conversion is set up at the beginning of the macro, so hopefully if you get any more different ones, you’ll be able to work out how to add them to the list:

myFind = "á,é,ä,ë,ö,ü,ô"

myReplace = "á,é,ä,ë,ö,ü,ô"

The macro is already aware of ligatures: ff, fi, fl, ffi and ffl, and will change them to separate letters. (That said, later versions of Word – certainly Word 2010 – recognise ‘conflict’ as correctly spelt, even though it here uses an fl ligature.)

**Sub\_SpellingErrorLister()**

**Sub\_SpellingErrorHighlighter()**

## Checking for Misspelt Proper Nouns

*(Video: https://youtu.be/JOTUvQAu-uo and https://youtu.be/PB0hXA\_1tRo)*

This macro makes a list of all the proper nouns (well, words with an initial capital) that appear in the text, and shows the frequency with which each occurs. It then goes through them all and uses a whole range of different tests, in order to find pairs (or groups) of words that might possibly be alternative spellings of the same name.

If the macro finds, say, *Beverley* and *Beverly* then alphabetically those are going to be next to one another in the list but if it finds, say, *Barnhan* and *Byrnham* then they would be further apart. So I’ve used random colours and different attributes, like underline and strikethrough for the pairs so that you can more easily spot which word it thinks might be a corruption of which other word.

So if you see a word and can’t see its matching pair immediately on screen, memorise the attributes (e.g. red highlight and bold text) and scroll further down. However, if it’s in the Os, say, you don’t need to go into the Ps because the macro only compares words within each alphabetic section by the initial letter. So, no, it would ***not*** find Allsworth and Ullsworth, sorry!

So the macro produces some useless information, but if you look through the list, you’ll hopefully be able to pick out a few gems, such as:

Brosseau . . . 3

Brousseau . . . 2

LeJeune . . . 4

Lejeune . . . 1

Shiroishi . . . 2

Shirioshi . . . 1

Zenderlend . . . 1

Zenderland . . . 3

I’m sure that I would never have spotted Shiroishi and Shirioshi at opposite ends of a 100,000-word book, and I find that clients and authors are impressed when you notice such things – but, of course, they don’t know that it was actually *the computer* that picked them up, not you!

**Long files** – If you have to test a file of more than about 250–300k words, the only thing you have to beware of is **Don’t touch the mouse!** It sounds silly, but I have discovered that moving the mouse over the active windows while a macro is running can cause the macro to paste the text it’s working on into the wrong file! So when I run a long analysis, I move the mouse to the far right, away from the working windows.

N.B. As with many of my macros, if you apply the strikethrough attribute to a section of text, the macro does not include it in this analysis.

**Sub\_ProperNounAlyse()**

## Frequency List of Full Names

*(Video: https://youtu.be/PB0hXA\_1tRo)*

This macro is mainly aimed at checking for inconsistency in people’s full names, including those with full first names, and those with initials (or both), but it is useful for any multiple-word proper nouns.

It provides the list of names twice: once sorted on first name, then sorted on last name.

|  |  |
| --- | --- |
| A Pninian | 1 |
| Al Cook | 6 |
| Alexander III | 1 |
| Alexander Petrovich Kukolnikov | 1 |
| Alexandra Smith | 1 |
| Alissa Zinovievna Rosenbaum | 1 |
| Allan Pryce-Jones | 3 |

etc...

|  |  |
| --- | --- |
| Adamovich, Georgii | 1 |
| Ahvnue, Cleef | 1 |
| Aims, Divergent Literary | 1 |
| Amis, Kingsley | 1 |
| Answers, AR | 8 |
| Answers, Ayn Rand | 2 |
| Aykhenvald, Yuri | 1 |

etc...

From my experience, this macro is most useful if you have run *ProperNounAlyse* first, and corrected any misspelling of proper nouns generally. This then focuses the attention of this macro into ***combinations*** of correctly spelt names.

When you run the macro, it gives you the option to includes names with initials or not.

As with many of my macros, if you apply the strikethrough attribute to a section of text, the macro does not include it in the analysis.

*ProperNounAlyse* used to have the facility for double names, but it was really slow, so I’ve taken it out. In any case, as I’ve said, it’s more effective if you correct the proper nouns first.

**Sub\_FullNameAlyse()**

## Find and Count Repeated Phrases

*(Video: https://youtu.be/PB0hXA\_1tRo)*

The CatchPhrase macro scans your document to find if any phrases are repeated and, if so, it checks how many times that phrase occurs. You end up with lists such as (searching for five-word phrases):

is a measure of the .... 3

this is a case of .... 4

of protons and neutrons in .... 2

it can be shown that .... 5

weighing in at a few .... 2

gold and silver and platinum .... 2

along with the rest of .... 2

However, you can specify that unless a phrase occurs more than a certain number of times, you don’t want to know, so if you had specified 5(4), it would search for five-word phrases and only report those that occur four or more times:

this is a case of .... 4

it can be shown that .... 5

The same macro can also be used for spotting any sections of text that have been accidentally repeated. For this, you just specify, say, 25. This means that if any section of 25 or more words is repeated, it will tell you. You can then use any of my search macros to get back and check the context of those two occurrences.

So, to specify what searches you want to be done on your text, you give it a list such as:

25, 6(4), 5(8)

This means is that the macro will first do a test for 25 words, reporting any and all it finds. Then it will look for six-word and five-word phrases, but it will only report those that occur a minimum of four times or eight times, respectively.

To add to the flexibility, I’ve set up the macro so that it gives you a menu of three different sampling regimes:

a = 25, 6(4), 5(8)

b = 6(3), 5(8), 4(10)

c = 7(3), 6(5), 5(10), 4(15)

and you can select a, b or c (but these regimes are set up at the beginning of the macro, so you can adjust them to taste).

Also, when you run the macro, instead of a, b or c, you can type in 8(4) or whatever, and it will just do that one test and stop.

Note that there’s not a lot of point in searching for really long phrases because Word can only search a maximum of 255 characters, so that’s about 35 words. Anyway, if there is a 20- or 25-word phrase repeated, you know immediately that it needs checking in context!

As you can imagine, checking the whole of a document can take a fair bit of time, so to help make the search faster, it creates a copy of the text – just the words plus hyphens, commas and apostrophes (which it replaces with temporary text-codes) and all the words are changed to lowercase.

It can take quite a while to create this text-only version of your file, so it’s worth saving it, in case you want to do some more tests later. Also, it’s probably worth doing some trial runs on subsections of the text, just to get an idea of how long it’s going to take – for a big text, you can always run it overnight.

So it’s worth trying relatively short sections of text first – say, 5000 or 10,000 words – to get a feeling of how fast (or slow!) the macro is. The following speed test results will give you an indication:

10,000 – 80 s

25,000 – 6 min

50,000 – 19 min

100,000 – 60 min

So as you can see, it’s increasing by a bit less than the square of the number of words:

25 −> 50 19/6 = 3.2×

50 −> 100 60/19 = 3.2×

i.e. each doubling of the number of words takes a little less than four times as long.

But when the macro runs, along the status bar, it reports what’s going on and, once it has found its first repeated phrase, it then tells you what time it thinks it will finish that run (e.g. ETA: 10:15 – “It’ll be finished about quarter past 10”).

But note that the ETA it gives you is for that particular number of words in the phrase, not the overall finish time for your whole set of tests. And it’s important to note that the more repeated phrases it finds, the longer it will take, so searching for shorter phrases takes longer, as these results show:

30 words 20 min

10 words 25 min

6 words 42 min

This is simply because, for shorter phrases, it’s going to find lots more of them. (However, I have since improved the counting speed, so the difference between searching for longer phrases and shorter phrases will be less.)

There is also now an option to run a quick test, to estimate the likely time it will take to do one run, i.e. the 20 minute speed in the above list.

**For the speed freaks among you:** To get the highest possible speed, you need to have as little as possible of the text showing on the screen. So, first you can reduce the size of the window that contains your file. It can go down very small, but it’s best to make it long and thin, so you can see the macro’s progress reports on the status bar.

But you can get up to 20% extra speed by letting the macro make the window totally invisible! To do this, at the beginning of the macro, you set goExtraFast = True. But because the Word windows are invisible, you can’t see the progress indication, so to use this option, it’s worth opening the VBA window and clicking Ctrl-G. This opens the Immediate Mode window, in which the macro’s progress is also reported.

The three control buttons at the middle top of the VBA window – start, pause and stop (like those on a DVD player) – can be used to control the macro. However, if you stop in the middle of a run – perhaps because it’s taking too long – you’re left with an invisible version of Word. Never fear! Simply run CatchPhrase again; it will see that Word is invisible, switch it back to visible again, and then stop – it won’t do another actual run of the tests.

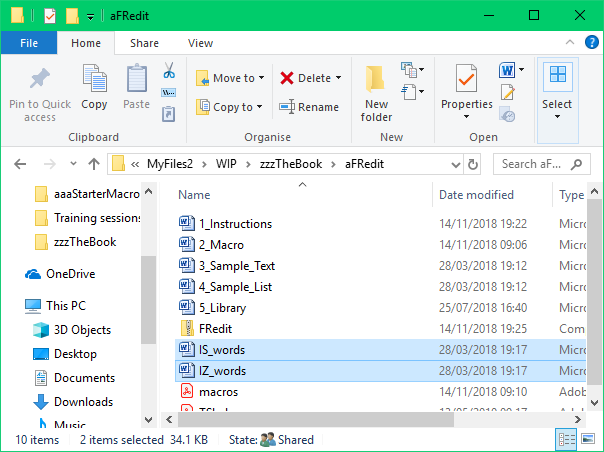
**Sub\_CatchPhrase()**

## IZ to IS Spelling and Vice Versa

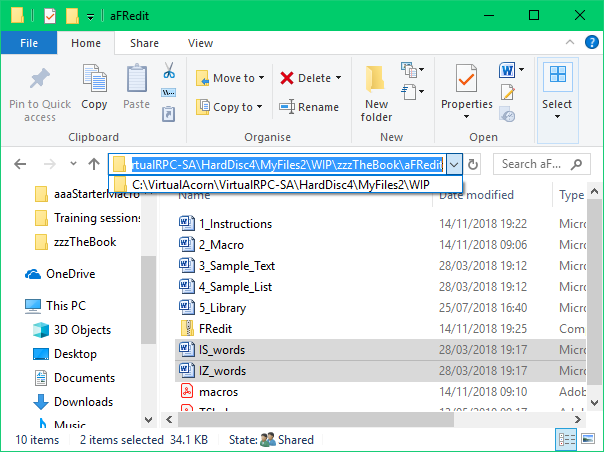
*(See video: https://youtu.be/SXmAJrUCZ\_I)*

The following two macros allow you either to highlight words that need changing or to actually change them. When you run the macro, it asks which you want to do.

The lists of exceptions need to be held in files (one for each macro) called ‘IS\_words’ and ‘IZ\_words’. You can set up each macro so that it automatically loads the relevant file from your hard disc, but the macro needs to know where on your disc to find it. You therefore have to put the full filename of the exception file into each macro. To do this, navigate to the folder (directory) where these two files are held:



If you click on the down menu arrow to the right of the line showing the string of folder names, the full path name appears:



Click Ctrl-C to copy this and add it into the line at the beginning of the macro. Suppose mine is:

C:\VirtualAcorn\VirtualRPC-SA\HardDisc4\MyFiles2\WIP

So now the line at the beginning of the macro:

myFile = "C:\Documents and Settings\Paul\My Documents\IS\_words"

has to be changed to (shaded so you can see what I’ve added):

myFile = "C:\VirtualAcorn\VirtualRPC-SA\HardDisc4\MyFiles2\WIP\IS\_words"

(or whatever it is on your computer).

The two lists of exceptions are among the IS/IZ macros in the TheMacros file.

If you discover other words that are exceptions, please email them to me so that I can update these as central lists. I have dated the lists so that you can check if you’ve got the latest version. I’ve put a yellow highlight on the proper nouns, because they may look a little funny; the macro requires the words to be in lower case.

N.B. You don’t need words like ‘disabled’ and ‘misapprehension’ in the list (and there are a ***lot*** of them!) because the macro ignores ‘isa’ if it’s too near the beginning of the word.

The *IStoIZ* macro takes account of the fact that, in ***UK*** English, analyse, catalyse, paralyse and hydrolyse keep the ‘ys’ form, but not in ***US*** English. It senses what the main language of the text is, and acts accordingly.

You can select the highlight colour at the beginning of the macro:

highlight\_Colour = wdTurquoise

And/or you can select the font colour at the beginning of the macro:

text\_Colour = wdTurquoise

If you don’t want the is/iz words changing in certain parts of the file (e.g. quotations and/or references lists) you can ‘protect’ the text (a) by using the strikethrough font feature (this is the same feature as is used with *FRedit*) and/or (b) by specifying the style names using the line near the beginning of the macro: nonoStyles = "Display Quote,References List", so just include your particular style name(s) in between the quotes.

You don’t have to have the is/iz words both track-changed ***and*** highlighted. However, if you ***do*** want them both track-changed and highlighted, change the option line to:

bothTCandHighlight = True

The text of ‘IZ\_words’ file is hidden in among the macros, pretending to be a macro!

**Sub\_IZ\_words()**

Similarly for the ‘IS\_words’ file:

**Sub\_ISwords()**

The actual macros are:

**Sub\_IZtoIS()**

**Sub\_IStoIZ()**

## Spellings with Varying Accents

*(N.B. If you run this macro and get the error ‘String parameter too long’, then that’s probably because you haven’t read these instructions properly! See italic phrase below. But also see the following macro, in case that helps.)*

This macro is to draw your attention to any words where, in different parts of the document, it is spelt with different accents, e.g. facade/façade. *You need to have already created a word-frequency list*, for example using TextStat:

abbrev 500

abc 2

abc-dq 1

abc-frame 2

Abdel-Rahman 1

Abdo 1

Abdullah 1

ability 12

able 23

Abnormal 1

abnormal 6

The macro looks through this list and gives you any that have different accents, e.g.

Ängquist . . . . . . . . . . 1

Angquist . . . . . . . . . . 2

Belanger . . . . . . . . . . 1

Bélanger . . . . . . . . . .17

CIGRÉ . . . . . . . . . . . .1

CIGRE . . . . . . . . . . . 47

Dennetiere . . . . . . . . . 1

Dennetière . . . . . . . . . 8

Dube . . . . . . . . . . . . 4

Dubé . . . . . . . . . . . . 4

edition . . . . . . . . . . .1

Édition . . . . . . . . . . .1

facade . . . . . . . . . . . 1

façade . . . . . . . . . . . 3

Gerin-Lajoie . . . . . . . . 1

Gérin-Lajoie . . . . . . . . 2

Saint-Germain-des-Prés . . . . . . . . 1

Saint-Germain-des-Près . . . . . . . . 1

Saint-Malô . . . . . . . . 1

Saint-Malo . . . . . . . . 7

If you want it to take account of any additional accented characters, then add them to this line at the beginning of the macro:

**Sub\_AccentAlyse()**

## List All Words Containing Accents

Maybe before you run the AccentAlyse macro above, you might like to have a list of all the words in your text that contain at least one accent. This macro creates a list of all the words that contain accents, multiple words occurring in the list multiple times, so you can see any commonly used words.

**Sub\_AccentedWordCollector()**

## Highlighting Duplicated Words

Word’s spellchecker (certainly in recent versions of Word) will throw up occurrences such as “the the”, asking if you want to correct it. So, if you want to draw such things to your attention, you can use this macro to highlight them.

One editor asked if this could be extended to two words and then three words, e.g. “he said he said” or “as it were, as it were.” So I’ve added that to the macro (it ignores punctuation between the repeated phrases).

N.B. This macro is based on a very ingenious find and replace worked out by Douglas Vipond of Canada. Many thanks, Doug!

**Sub\_DuplicatedWordsHighlight()**

And as the highlighting is done by F&R, you can do it instead with *FRedit*.

An alternative solution to this problem is a macro that jumps to the next duplicated word, so that you can look at it and, if necessary edit it there and then. However, you need to have one macro for each of the one-word, two-word and three-word cases.

**Sub\_DuplicatedWordsFind()**

**Sub\_DuplicatedWordsFind2()**

**Sub\_DuplicatedWordsFind3()**

## Page Numbering PDFs

I’ve listed this macro here because it’s prime for use with SpellAlyse, and is ideal for use when proofreading.

The idea is that you copy and paste the text out of a set of PDFs, making sure you paste them into your Word file in the correct order. Once this file has been SpellAlysed, it’s an excellent aid to spotting spelling errors, page by page. It’s also useful if, at some stage, you have to search through for all occurrences of something and want to know which page each occurrence is on so that you can make changes to the paper version.

The problem is that the text you’ve scraped out of the PDFs, while it is in page order, the page numbers are often not easy to spot. The *PDFpager* macro(s) will (a) make the page numbers stand out – larger fonts and some chevrons: **>>56<<** – but also it will add a dotted line to indicate the page break before each one.

This simple macro will only work if the page number is against the left-hand margin. Before running the macro, place pairs of chevrons – as above – on the very first number in the file (even if it starts not from 1 but from something higher), and on the very final number in the file.

N.B. I’ve called it ‘Simple’ because you may have a PDF where the page number is harder to identify. If so, do get in touch, and we’ll see if we can find a way of doing the same sort of thing with your file.

*Ha ha ha!* I’ve just realised, after all that, as the page numbering is so simple on this first PDF I’ve worked on – each page number on a line of its own – I can achieve ***exactly*** the same effect with one line, using *FRedit*:

That said, the next one is more of a challenge: the numbering on odd and even pages is different:

**2 Preliminary remarks**

**Preliminary remarks 3**

4 Preliminary remarks

For this one I’ve done PDFpagerOddEven (below).

**Sub\_PDFpagerSimple()**

**Sub\_PDFpagerOddEven()**

## Word and Phrase Frequency

Put a line of full stops at the end of a file, and follow it with a list of words or phrases, and this macro will look through the file, counting them all for you, so that you end up with, for example:

..................

it can be shown that 6

my husband and I 3

at this point in time 7

or perhaps you might use it for ...

..................

co-ordinate 5

coordinate 1

co-operate 4

cooperate 3

[It occurs to me that it might be helpful to be able to select *either* a ‘whole word’ count or one which would look for, say, ‘co-ord’ or ‘coord’ as part of a word. If anyone wants that as an option, just ask and I’ll see if I can implement it. PB.]

The first line of the macro is used so that you can decide whether or not to count the words case-sensitively or not.

(Just to explain, the ‘Hide/Restore the hyphens’ sections are necessary because VBA’s ‘MatchWholeWord’ property treats ‘co-ordinate’ as two separate words, and you can’t use ‘<’ and ‘>’ because that requires wildcards, and case-insensitivity doesn’t work with wildcards! The trick is to change them all to, for example, ‘co-ordinate’, create the list and then change them back again.)

**Sub\_PhraseCount()**

## Checking for (and Counting) Duplicated Sentences

*(Video: https://youtu.be/xkxuZwF9oIY)*

*(Do watch the video; you’ll get the idea how this macro works much more quickly than me trying to explain it in words!)*

This macro checks all the sentences in a document, to find out if any are (exactly) duplicated and, if so, it counts how many times they occur. It lists the duplicated sentences in a separate document.

(This version replaces an earlier version that was so slow as to be unusable on anything longer than a couple of thousand words.)

Before you run this macro, I **strongly** suggest you first run *CopyTextWithSomeFeatures*. This will generate a copy of your document, ensuring that all the text, including that in the foot(end)notes and textboxes is included in the check that this macro is about to run.

When you then run this macro on the copied text, it looks through the sentences alphabetically. This is how it manages to be so much faster than the earlier version: it only has to compare the “A blank blank blank” sentences with the other sentences starting with “A”, and the “Blank blank blank” sentences with the other sentences starting with “B”, etc.

Not only does it produce a list of duplicated sentences, showing to frequency of each, but it also creates a FRedit list that you can use to highlight the occurrences of the duplicated sentences in the original text.

At the beginning of the macro,

minWords = 10

sets the minimum length of “sentence” that the macro bothers to check. Otherwise short section headings, and column headings in tables can end up being reported as “duplicated sentences”.

**Sub\_DuplicateSentenceCount()**

## Visualise Where Specific Words/Phrases are Used

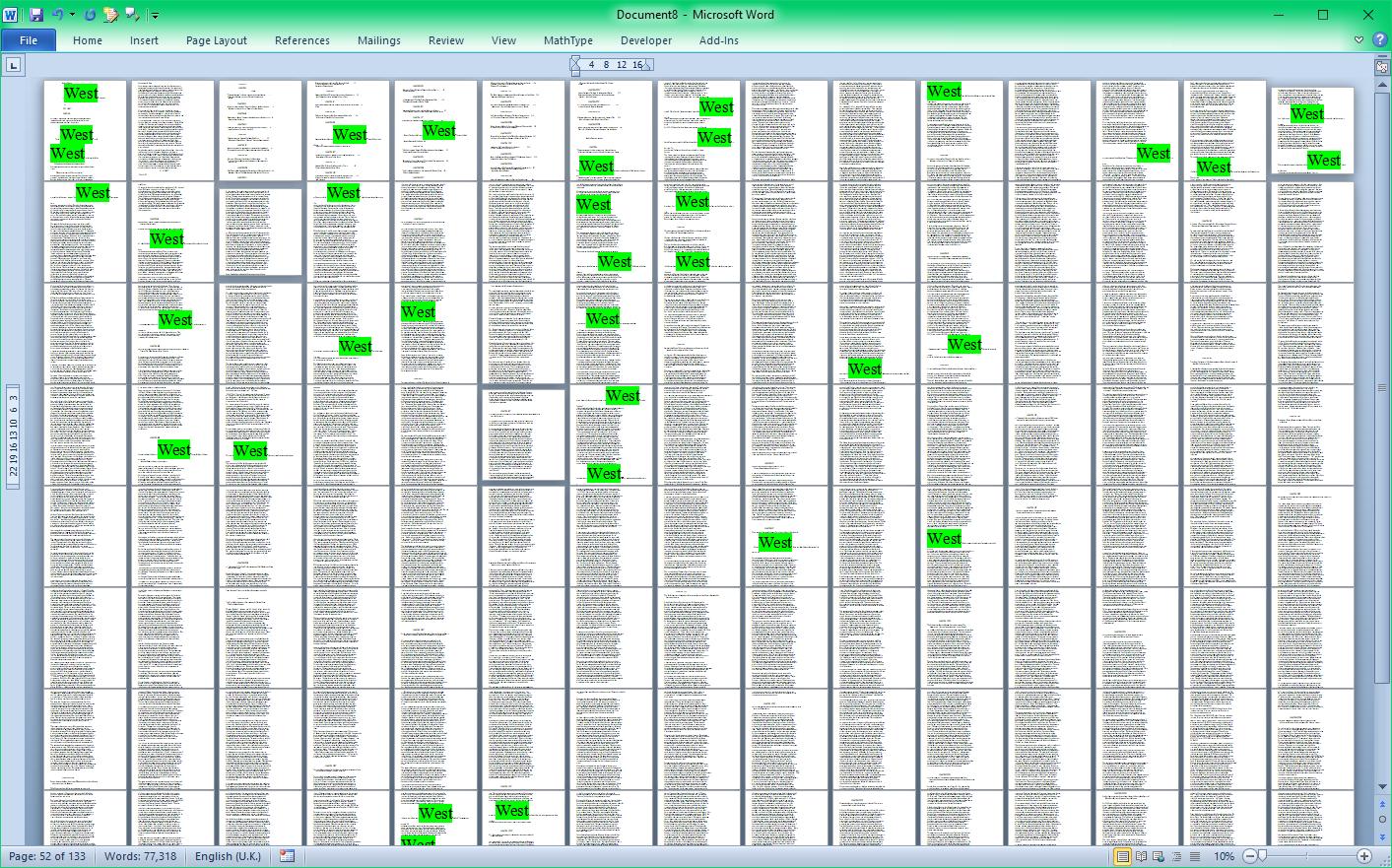
*(Video: https://youtu.be/2PG7n5MCMCo)*

The idea of this macro is to give you a quick way to see where and how often a given word or phrase occurs within a document. For example, it might be the name of a character in a novel or a legal phrase in a legal document, or the name of a chemical in a technical document.

The macro creates a copy of the document under test, finds the test word/phrase and increases its point size, massively, to make it stand out. Then it reduces the zoom size of the document view, so that you can see several pages at one go – but obviously you can zoom the document view in and out further, according to taste.

So open the document in question and run the macro. If no word is selected, it will ask you to type in your word; if a word is selected, that’s the word it will search for.

The macro then creates a copy of the original, in which it will highlight and enlarge occurrences of that word and it will reduce the zoom size of the window, so you can see lots of pages at one go:



If your word has some uppercase characters in it, such as a name, the macro allows you to search case sensitively (or not). So above, ‘West’ was the word, and I said I wanted it case sensitive, so it ignored ‘west’.

Also, you can use the wildcard symbols ‘<’ and ‘>’, so if you want, say, to avoid ‘Western’, I could use ‘West>’. And you can use Word’s other wildcard F&R codes, like ^p and ^t or ^=, ^+, etc.

If you’ve already run the macro once – i.e. it has created a copy of the original file – it will continue to use that copy, and you can just keep graphing different words/phrases.

Other options, which should hopefully be self-explanatory, are:

myFontSize = 120

myZoomSize = 10

myHighlight = wdBrightGreen

' myHighlight = wdNoHighlight

**Sub\_WordGraph()**

## Checking Hypenation of Word Pairs

*(Video:* ***New****: https://youtu.be/LHWUVKgU-hs –* ***Old****: https://youtu.be/olyCyDzCDe8)*

(N.B. This macro is complementary to WordPairAlyse, which is worth checking out, if you’re a consistency-freak, like me!)

This macro analyses the text to find how often word pairs are hyphenated, as two words or as a single word, for example: ‘run-off’, ‘run off’ and ‘runoff’, and it also now picks up words separated by an en dash, e.g. ‘blue–green’.

(N.B. If you are a *FRedit* user then look at the macro *HyphenationToFRedit* in the ‘Quicker Creation of *FRedit* Lists’ section below – it will save you a lot of time!)

Here’s a sample output:

|  |  |  |  |
| --- | --- | --- | --- |
| above-mentioned . . 2 |  | abovementioned . . 3 |  |
| afore-represented . . 1 |  |  |  |
| all-band . . 1 | all band . . 1 |  |  |
| art-methods . . 1 |  |  |  |
| attention-based . . 12 | attention based . . 19 |  |  |
| attention-guided . . 3 | attention guided . . 1 |  |  |
| attention-modulated . . 25 |  |  |  |
| band-pass . . 18 | band pass . . 1 |  |  |
| bell-like . . 1 |  |  |  |
| bell-shaped . . 6 | bell shaped . . 2 |  |  |
| between-coefficient . . 1 |  |  |  |
| binary-classification . . 1 |  |  |  |
| bit-stream . . 17 |  | bitstream . . 2 |  |
| block-based . . 28 | block based . . 1 |  |  |
| block-wise . . 1 |  |  |  |
| blue-green . . 2 |  |  | blue–green . . 4 |
| bold-faced . . 1 |  |  |  |
| bottom-left . . 1 | bottom left . . 1 |  |  |
| closely-packed . . 5 | closely packed . . 3 |  |  |
| contrast-based . . 1 | contrast based . . 4 |  |  |
| corner-based . . 1 |  |  |  |
|  |  |  |  |

As you can see, as well as counting the hyphenated word (and that includes triple and quadruple words – e.g. the much-over-used expression, ‘state-of-the-art’).

Any item that only occurs as one type of word pair is unlikely to be an inconsistency, so they are coloured light grey to help to draw attention away to the more important word pairs.

The macro also flags up (in red) any word pairs that occur in two form that are more likely to be inconsistencies, e.g. ‘co-axial’ and ‘coaxial’, which clearly need to be made consistent. Word pairs that occur only in columns 1 and 2 are probably OK: you could have, ‘With contrast based on iris size, you have to use a contrast-based assessment.’

One exception to this is, for example: ‘closely-packed’ and ‘closely packed’. Since many editors consider the hyphen to be superfluous with ‘-ly’ adverbs, these word pairs have also been coloured in red.

The macro also counts (and displays in blue) all the words starting with certain specific prefixes, whether or not they are appear in hyphenated form, for example:

|  |  |  |  |
| --- | --- | --- | --- |
| non-ambiguity . . 1 |  |  |  |
| non-attentional . . 2 |  |  |  |
| non-equal . . 1 |  |  |  |
| non-explicit . . 1 |  |  |  |
| non-gaussian . . 1 |  |  |  |
| non-head-mounted . . 1 |  |  |  |
| non-homogeneous . . 1 |  | nonhomogeneous . . 1 |  |
| non-ideal . . 1 |  |  |  |
| non-interest . . 1 |  |  |  |
| non-linear . . 24 |  | nonlinear . . 2 |  |
| non-linearly . . 3 |  |  |  |
| non-object . . 2 |  |  |  |
| non-oscillation . . 1 |  |  |  |
| non-overlapping . . 2 |  |  |  |
| non-parametric . . 1 |  | nonparametric . . 2 |  |
| non-reference . . 4 |  |  |  |
| non-roi . . 2 |  |  |  |
| non-uniform . . 1 |  |  |  |
| non-zero . . 4 |  |  |  |

To specify which prefixes you want checking, you can use the line:

myList = "anti,cross,eigen,hyper,inter,meta,mid,multi," \_

& "non,over,post,pre,pseudo,quasi,semi,sub,super"

In non-technical work, you might not need them all, so you could delete some of them from the list, although it won’t actually make a lot of difference to the overall speed.

This macro is a bit slow on large complicated files. I ran it on a 213,000-word file that had a lot of hyphenated words and it took 54 minutes, coming up with 2154(!) different hyphenated or prefixed words. Other timings: an 111k file with 1447 items took 20 mins, and 70k file with 520 items took just 5 mins. This is with a eight-year-old desktop computer that wasn’t top-of-the-range when I bought it.

As supplied, the macro includes numbers in its search, so that it will find, say, ‘2D-based’ or ‘9-mm’ or ‘non-90-degree’. You can speed up the macro up slightly, if you tell it not to include numbers, by changing the line:

include Numbers = True

to False.

The other option is to display the results table whether or without lines. With lines, it looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
| bell-shaped . . 6 | bell shaped . . 2 |  |  |
| between-coefficient . . 1 |  |  |  |
| binary-classification . . 1 |  |  |  |
| bit-stream . . 17 |  | bitstream . . 2 |  |
| block-based . . 28 | block based . . 1 |  |  |
| block-wise . . 1 |  |  |  |
| blue-green . . 2 |  |  | blue–green . . 4 |

which I find more difficult to read. This is set with:

deleteTableBorders = True

**Hint**: On a big file, the status bar should show you the progress, and if you do want to stop the macro from running, you should be able to do so by pressing Ctrl-Break. However, Word does sometimes ignore Ctrl-Break on a hard-working macro.

**Sub\_HyphenAlyse()**

## Transferring Words from HyphenAlyse to a Stylesheet

Once you’ve run *HyphenAlyse* and looked through the author’s usage of hyphenation, you’ll be making decisions about which words will be hyphenated and which not. These decisions would normally be recorded in a word list as part of a stylesheet, and this macro speeds up the production of that word list.

Obviously, I’ve done it to create a word list in the format I use. If you don’t like it, maybe I can alter the macro to suit your taste. My lists look like this:

auto<word> – NONE are hyphenated

chincap

chincup

cooperat...

coordinat...

cross bite

hyper<word> – NONE are hyphenated

focused

infra<word> – NONE are hyphenated

intra<word> – NONE are hyphenated except…

intra-arch

inter<word> – NONE are hyphenated

mesiodistal

meta-analysis

mid<word> – NONE are hyphenated

multi<word> – NONE are hyphenated

non<word> – ALL are hyphenated

overjet

post<word> – NONE are hyphenated except…

post-pubertal

post-retention

post-surgical

post-treatment

pre<word> – NONE are hyphenated except…

pre-adolescent

pre-phase

pre-treatment

semi<word> – ALL are hyphenated

sub<word> – NONE are hyphenated except…

sub-gingival

super<word> – NONE are hyphenated except…

supra-eruption

while (not whilst)

So, to create the hyphenation items of this list, look through the hyphenation list, marking it up according to your decisions, using highlighting for the ‘rules’ and either underline or strikethrough (whichever you prefer) for the exceptions:

1) For no hyphenation of a particular prefix, highlight just the prefix, e.g. autorotation . . . . 2

2) For hyphenation of a particular prefix, make sure you highlight the hyphen: non-linear . . . 14

3) For the exceptions, apply an underline (or strikethrough) to some part (any part) of the cell: intra-arch . . . . 6

4) For any other word that you need to mention specifically in the list, highlight ***at least*** the whole word: chincap . . . 9 (i.e it’s OK to use double-click, which also highlights the following space.)

N.B. You only need to highlight ***one*** of the words that starts with, say, ‘auto’, not all of them!

**Hint**: I find yellow highlighting doesn’t show up very well on text, so I’ve set up my *HighlightPlus* macro to use bright green as my colour of choice.

**Sub\_HyphenationToStylesheet()**

## Checking Word Pairs that are Not Hyphenated

*(Video: https://youtu.be/LHWUVKgU-hs)*

If, for example, your text uses ‘web site’ and ‘website’, but does not use ‘web-site’, then *HyphenAlyse* will not detect it. Unfortunately, it needs a lot more computing time to check every single pair of words in the whole text, to work out whether that word might also occur somewhere as a single word – just think how many different word pairs there are in this paragraph: , etc., all of which need to be checked.

Here’s the output for the first job that I tried this new macro on:

Before you run this macro, I strongly suggest you first run *CopyTextWithSomeFeatures*. This will generate a copy of your document, including all the text in any foot(end)notes and textboxes.

Here’s a list of timings for different sized files on my desktop computer, to give you some idea of how long you’ll have to wait:

6.7 kwords 1 min

22 kwords 4 min

86 kwords 38 min

159 kwords 91 min

In order to speed up the process, I’ve made it so that the macro can ignore a set of common words that (we assume) won’t form part of compound words. For example, when checking absolutely all the words, to scan the 86k file, the macro took 92 minutes (2.4 times as long).

However, if you decide that you don’t want to risk missing any compound words, you can tell the macro **not** to delete any words. You do this by changing mySet = 1, to mySet = 0, at the beginning of the macro.

Alternatively, you can be more selective which words are ignored, by creating your own list (or editing the existing list) and selecting mySet = 2 or mySet = 3.

Having used it a number of times, I noticed that it wasn’t finding any (or many) word pair inconsistencies after about 20–30% of the way through the book. this is because the most common word pairs, being numerous, will be spotted quite quickly – but you have to keep checking because there might be other, more obscure word pair inconsistencies.

But then, if they are obscure with, say only one occurrence each of the two-word and one-word forms, maybe it’s not worth waiting. So I have added a feature where it does, say the first 30% and then asks if you want to carry on for a further, say 10% – you can play around with this if you want to. When you say you don’t want to continue, it sorts the results it has found into alphabetic order. This is set up at the beginning of the macro with:

pausePercent = 30

stepPercent = 10

If you don’t want the macro to pause (because you’ll have to be in attendance to click it to continue) then set:

pausePercent = 100

i.e. don’t pause until you’ve done 100% of the job, i.e. don’t pause!

If you’ve already started the macro, but forgotten to open VBA to see the progress report, you can just press Ctrl-Break, and it will stop the macro running (temporarily). If you then click ‘Debug’, it will open the VBA window, then click Ctrl-G will open the Immediate Window. If you now click F5 (or click the Run Sub\_icon – a triangular ‘Play’ icon), it will continue running, and you can watch its progress.

You get a display something like this:

## sure Average Sentence Length

Someone asked me for a macro that would not only measure the average length of sentences in terms of words, but also give the standard deviation.

Writing it was an interesting job because it revealed that Word’s own ‘readability statistics’ which purport to give a figure of average sentence length are a little questionable. In case you are interested, this macro has an option at the beginning which allows you to decide if you want to see what Word thinks. It currently does not show the stats. I’m sure you can work out what to do to make it show the stats.

More interestingly, it also does two versions of the average sentence length. The first takes the text as a whole. The second version first deletes all paragraphs that don’t have a full stop at the end. The idea is that this will delete headings and items in lists. Both sets of figures are given by the macro, so you can pay attention to whichever is the more meaningful in a given situation.

This latest version also does a frequency distribution showing the numbers of sentences in each of a range of lengths, e.g.:

1 to 3 = 7

4 to 6 = 9

7 to 9 = 4

10 to 12 = 11

13 to 15 = 8

16 to 18 = 5

19 to 21 = 7

22 to 24 = 4

25 to 27 = 1

28 to 30 = 6

31 to 33 = 2

34 to 36 = 4

37 to 39 = 0

40 to 42 = 0

43 to 45 = 2

You can choose your range of lengths using the myStep value at the start of the macro. The list above is myStep = 3, so I’m sure you can work out how to change it to other values.

(Watch out for a version which also does the same sort of thing with word length.)

**Sub\_ SentenceAlyse()**

## Highlight Over-Long Sentences

If you want to draw attention to all the very long sentences in a text, this macro sets two lengths and highlights sentences of these lengths or more in yellow (medium) and red (megalong).

**Sub\_ LongSentenceHighlighter()**

## Count Uppercase and Lowercase Characters

This is more for fun than anything else, but when I had One of Those Jobs where the Author thinks that Every Important Word has to have an Initial Cap, I wondered just how many words I had decapit(alis)ated. As I say, just a bit of fun to run this before and after.

**Sub\_ CountCase()**

## Count Words that Are Highlighted

If you place the cursor in an area of text in a certain highlight colour, it collects and counts all the text in that colour.

If you place the cursor in an area of text that is ***not*** highlighted, it collects and counts all the text in ***any*** colour.

**Sub\_ CountWordsInHighlightColour()**

## Copy Paragraphs that Contain Highlighted (and Coloured) Text

This macro creates a new document consisting of a copy of each and every paragraph that contains some highlighted (or font-coloured) text. You can choose to double-space the paragraphs.

**Sub\_ CopyHighlightedText()**

## Highlighting Words Not in Vocabulary List

The aim of this macro is to check through a text and highlight any ‘too difficult’ words. The definition of ‘too difficult’ is more than a certain length (set by ignoreLength = 3) and not included in either the list of ‘easy’ words at the beginning of the macro, or in a word list file.

To run the macro, you must have two and only two files open in Word: the text file being tested and the word list file. The latter is defined by having the words ‘Word list’ at the top, e.g.

**Word list**

macro

probably

patiently

patient

happening

etc

If you don’t want the easy-words feature, just use

easyWords = ""

And if you don’t need the ‘minimum word length’ feature, set

ignoreLength = 0

**Sub\_ TooDifficultWordHighlighter()**

## Point Out Repetitious Use of Words

The idea here is that you are looking for the same word (or derivatives) being used too close together: “The start of the list had to be started within...”.

The macro searches through the text until it sees two words within a certain distance (number of words) of one another, and stops.

The usefulness of this Macro will depend on your application, and you’ll need to ‘tune’ the macro so as to (a) not stop at repetitions that are normal and acceptable (“Is that the book that John showed you”), but not to miss the useful occurrences of repetitiveness.

So the macro allows you to set:

a) the minimum length (characters) of words that it checks: e.g. minLength = 4

b) the maximum distance apart that the repeated words can be: e.g. rangeWords = 20

c) words that need to be ignored however often they are repeated:

e.g. ignoreWords = "their,these,what,that,which"

If you have, say, minLength = 4 then you don’t need to have “who,the,you” in your ignore list.

As I say, you’ll have to try it and see how best to set the conditions for your own application.

**Sub\_ FindRepeatedWords()**

## List Special Sorts

This macro creates a list of all the different special characters that occur in the document. You can choose to include in the list ordinary accented characters, such as é, á, î, ñ etc by setting listAccentedChars = True.

Warning: Try this macro first on a small file – two or three paragraphs with some known special sorts. Why? Well, if you work on a large file, it can take quite some time, and, as with any long-running macro, if you start clicking on the screen to see if it’s still working, you can cause Word to crash. I’ve added in some encouraging beeps at various stages, to reassure you that it’s working and will get there in due course. It should make a total of four beeps before finishing.

The result might look like this:

Special sorts used:

  thin space

− minus sign

±

non-breaking space

α

ε

ζ

η

You can add explanatory wording for any of the characters, by adding it into the macro, following the format used.

**Sub\_ SpecialSortsLister()**

## List All Words in a Given Font Colour

This was a requirement of someone who had special words in a text that had been highlighted using a font colour. They wanted to create a glossary of all these words, so this macro finds all the words in the text that appear in the current font colour (i.e. place the cursor in the first such word), and sorts it alphabetically.

**Sub\_ ListAllColouredWords()**

## List All Text that is Highlighted

This macro finds all the text in the current file that is highlighted, puts it into a list in a separate file and sorts it alphabetically.

**Sub\_ ListHighlightedText()**

## List All URLs in a File

This macro creates a list, in a separate Word file, of all the URLs, both the visible text and the underlying URL.

**Sub\_ ListAllLinks()**

## List All Paragraphs Starting with...

*(Video: https://youtu.be/t4ADwZ4QwTA)*

Someone asked for a macro that created a list, in a separate Word file, of paragraphs that started with a particular word. This is a more generalised version of that macro. It allows you to type in the specified word, or to select some text, and it will use that text, or if you don’t select any text, it picks up the current word at the cursor and offers you that as a possible word to use.

This macro actually copies the paragraphs, so it pulls all the formatting with it. I used the macro on this book, to give me a list of all the lines starting with “Sub”, i.e. to generate a list of all the macros, and it was a bit slow (well, there are currently 580 such paragraphs!), so I modified it so that it just picked up the text, without the formatting, which is a lot faster.

**Sub\_ ListOfParas()**

**Sub\_ ListOfTextParas()**

## List All Text in a Given Font or Highlight Colour

Place the cursor in a bit of text in the given font colour or highlight colour and run the macro. It creates a new document, copies the text and removes everything ***not*** in the required colour. It then gives you the option to sort the list alphabetically.

You can choose whether or not to remove the highlighting or font colouration using the first line of the macro:

removeColouration = True

**Sub\_ ListHighlightedOrColoured()**

## Making a Sublist of Items in a List Containing a Word or Phrase

*(Video: https://youtu.be/DnG1XCuOUlk)*

Maybe you have a list of citations:

Archi-Media founded in 1992

Engineering and Grabher 2001

Beijing the 2008

Blau 1984

Blau and Lieben 1983

Campagnac and Winch 1997

Carr et al 1999

Coxe et al 1987

Eastman et al 2008

And you want to create a sublist of all the citations from the 1990s. So, select ‘199’ and run the macro – or just run the macro and type in ‘199’ and will create a new file and find all the items in the list containing that text.

If it’s a word (or words) you’re searching for then you can set the macro to be case sensitive, or not:

caseSensitive = True

It was designed for searching through a list, but it’s just looking through the paragraphs, really, so it can be any text. For paragraphs, you might want a blank line between each paragraph found, so put:

addBlankLine = True

**Sub\_ ListOfList()**

## Making Text Boxes [Textboxes] Visible (+ Odd Fonts + TrackChanges etc.)

This macro was born out of a long hard search for a way to make rogue text boxes [textboxes] (and graphics boxes) more easily visible on screen.

I realised that if the main text could be made INvisible, the ‘added bits’ would be very clearly visible. So this macro turns all the main body text to white font so that white on white = invisible. Run it again to restore the text visibility.

If no text is selected, the macro makes the whole text black-to-white inverted. If text is selected, it only changes the selected text. So, you can whiten all the text, look through to see what’s what, then if you get to a section of interest, you can restore the text blackness on just that bit of text.

“What about text in other colours?” Well, when running FRedit on a file to make global changes prior to editing I use colour text (rather than highlighting) to show text that has been changed but where I don’t want it track-changed. So this macro ***only*** whitens ***black*** text – coloured text remains.

What’s more, track changes, since they are red, not black, remain visible.

Here’s a sample bit of text (the equations have been highlighted using *EquationsHighlightAll* ):

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Provided that the grounding arc occurs at phase A when *t*= *t*1 (*u*A = +*Um*, *Um* is the amplitude of the phase voltage of the supply), the voltages across the three-phase capacitances at the time prior to the arcing instant are the three-phase supply voltages, respectively given by the following set of equations.





Right after the arcing, charges stored in the capacitance to ground of phase A *C*1 flow into the ground through the arc and its voltage is reduced to zero; the capacitances to ground of the two non-fault phases *C*2and *C*3 are charged by line voltage *u*BA and *u*CA via supply inductance with their voltages transiting from the initial value  to the instantaneous value of *u*BA and *u*CA. The frequency of the high-frequency oscillation is determined by supply inductance and conductor-to-ground capacitance. The stable values of the voltages across the three-phase conductors are given by





~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

***Important hint***: If you do a Ctrl-A to select all text, text boxes are given a blue tint borders, which also helps visibility. Try it now, on this text here.

**Sub\_ HideShowText()**

## Finding Chronology Words in Context

*(Video: https://youtu.be/PB0hXA\_1tRo)*

This is aimed at fiction editors wanting to trace the chronology of a novel. The macro extracts, into a separate file, all the paragraphs containing appropriate chronology-type words: Monday, Wednesday, Fri, Sat, April, June, 1958, 2017, etc. These words are highlighted so that you can easily check as you scan through the list of paragraphs.

N.B. If you think of other chronological type words you would like it to detect, please let me know, as this isn’t a macro I’ll actually be using.

I wondered if things like ‘age’, ‘aged’, ‘years old’ would be useful, so I added more searches but then realised I was picking up things like ‘pages’ and ‘waged war’, so I split them into four groups, making one whole word searches:

' Case sensitive

myWords\_1 = "Monday, Tuesday, Wednesday, Thursday, Friday,"

myWords\_1 = myWords\_1 & "Saturday, Sunday,"

myWords\_2 = "January, February, April, June, July, August,"

myWords\_2 = myWords\_2 & "September, October, November, December"

' Case insensitive

myWords\_3 = "years old, tomorrow, next day, morning, evening, week, month"

' Case insensitive + whole word

myWords\_4 = "age, aged"

' Case sensitive AND whole word

myWords\_5 = "May, March, Mon, Tue, Tues, Wed, Weds, Thu, Thurs, Fri, Sat, Sun"

**Sub\_ ChronologyChecker()**

*(Video for this next section: https://youtu.be/2hrfWRyDx18)*

But if you prefer to have these chronology words highlighted within the text, you can just use FRedit, with a list something like this:

| Highlight chronology words

If you then want to unhighlight these chronology words, you can use the *UnHighlight* macro, selecting a bit of the green text, so that it knows to unhighlight the green. However, I’ve written a specific macro for this, and the highlight colour it will remove is set in the first line:

myColour = wdBrightGreen

**Sub\_ ChronoColourOff()**

## Finding Names/Words/Phrases in Context

*(Video: https://youtu.be/PB0hXA\_1tRo)*

This macro was written originally for fiction editors, to track the occurrence of specific names through a novel, but I realised that it could have a wide range of other applications, so I expanded it to be optionally case sensitive and to include phrases. You give the macro a list of names/words/phrases, and it searches for any paragraphs in the text that contain them. It creates a separate file of those paragraphs, with the searched element highlighted in your choice of colour.

If you use a search text such as ‘Brown’, it will be case sensitive and therefore not find ‘brown sugar’. However, if you use, say, ‘¬van de Waal’, it will find all the permutations and combinations of van/Van, de/De.

You can include punctuation, for example, ‘However,’ or ‘such as:’, and it can also find numbers: BS 942, 999, 2016, 22/9/48, 9/11, etc.

You can input some searches in one of four ways:

1) Put a list of names/words/phrases at the beginning of the macro

findWords = "Brown | Jones | ¬van der Waal"

but in this case all the found search words will be highlighted in the same colour, as specified by myBasicColour = wdYellow at the beginning of the macro.

2) At the end of the document you are testing, you can add, say:

Context words:

Bloggs

Brown

¬van der Waal

then when the macro runs, it will search for each of the texts in that list, colouring them accordingly.

N.B. The macro searches for ‘Context words:’, so that exact text must precede the list of searches.

(Being lazy, I’ve added a feature so that you can just put the list of words at the end of the file, put the cursor in the first one and not even bother highlighting them.)

3) You can place a list as per the example above, but put it at the end of either your FRedit list (zzFReditList) or your MultiSwitch list (zzSwitchList). N.B. These lists must be at the end of the file, with no following text.

Not having to put the list into the file under test would be very useful if you wanted to do this context search for each and every chapter file of a book. For example, you might want:

Context words:

Figure

Table

Equation

Eqn

¬chapter

¬chapters

¬section

¬sections

You can probably see my thinking here: it would be quite quick to check continuity of numbering in this way. That said, the macro FigTableBoxLister will give much more information about these elements of you file. But it might be useful for equations, and chapter and section citations.

You can, of course, do this context checking as part of your FRedit list by using:

DoMacro|WordsPhrasesInContext

However, you’ll also need to set:

returnToText = True

so that after FRedit has run this macro, the focus would return to the main text so that FRedit can continue doing the rest of its F&Rs.

4) If you want to just search for a word/phrase, then as long as there’s no ‘Context words:’ available it will prompt you to input your search at the keyboard. However, it will offer you the currently selected text or, if nothing is selected, the current word, and you can just press Enter.

Hint: If there’s a ‘Context words:’ available but you don’t want to use it, simply type a space in the middle of ‘Context’, and the macro will ignore it.

**Sub\_ WordsPhrasesInContext()**

# Pre-editing Tools

Here are a few macros, some of which you might want to use before you actually start the sentence-by-sentence reading of your text. The idea is that if you can get macros to change a lot of the obvious and repetitious things, you will be better able to concentrate on the really skilled part of the job – making sure that the text says what it is meant to say.

The most powerful single macro is *FRedit*, and there’s only a brief introduction to it in this book – it has its own documentation. Then this section also covers ways in which you can add typesetting codes to your book (<A>, <B>, <C> etc – although this can also be done with *FRedit*), ways of combining and dividing up the files that form your book, a macro to create a list of all the acronyms in a file, macros to pull all the tables and/or figures out into a separate file, and other macros to ‘do things with’ tables, frames and textboxes.

These are then followed by macros that ‘do things globally’ with footnotes and endnotes, bookmarks, comments and styles. And there’s a miscellany of other things that you might want to do before you start to read the script.

## FRedit – Scripted F&R

N.B. The text below is just a brief description of the idea of *FRedit*. **Please do not try to use *FRedit* without reading the *FRedit* instruction manual.** The manual and some sample files are available from: http://www.archivepub.co.uk/documents/*FRedit*.zip

Also, there are some (helpful?) videos:

FRedit for Beginners Part 0 (4:15) – Find FRedit and install it on your computer: https://youtu.be/B7ouU3OzWRE

FRedit won’t work! (3:20) – Help with trying to get FRedit working for the first time: https://youtu.be/nVcneZZgV2g

FRedit for Beginners Part 1 (4:17) – Here’s how to get started: https://youtu.be/X9e7770QWiY

FRedit for Beginners Part 2 (3:50) – More techniques for file clean-up: https://youtu.be/4hlnYqyfOQk

FRedit for Beginners Part 3 (4:13) – Learn more things FRedit can do for you:https://youtu.be/GO8iW0WBfp0

FRedit for Beginners Part 4 (5:08) – Adding wildcards into your FRedit armoury: https://youtu.be/Fq6p\_RdMHb4

FRedit for Beginners Part 5 (5:24) – Adding and removing formatting: https://youtu.be/MXiPpz0yVQE

FRedit for Beginners Part 6 (5:53) – A few tricks of the trade: https://youtu.be/ohPwmX1mS00

FRedit for Beginners Part 7 (16:31) – The FRedit Library: https://youtu.be/EulSXDNJw0k

Advanced FRedit use (10:49) – Some high-power uses of FRedit: https://youtu.be/wHR7Yl\_dBjI

FRedit queries 1 (7:58) – Answers to a couple of queries about FRedit: https://youtu.be/QYEIVbmIMQA

FRedit revision 1 (15:09) – Reinforcing the basic principles of FRedit: https://youtu.be/Mt4iuh6SOAM

FRedit revision 2 (5:50) – More about applying attributes, plus another gotcha: https://youtu.be/aQ\_hR\_K-INM

FRedit with notes and text boxes (4:58) – FRedit with foot/endnotes and text boxes: https://youtu.be/bGIMXppJlFM

FRedit and MultiSwitch (9:09) – Differences and similarities between the two macros: https://youtu.be/yGZHej6vaZ4

New way to run FRedit (5:41) – Run FRedit by selecting a FRedit list from a menu: https://youtu.be/1bVduGAFrhU

MiniFRedit (4:34) – Subset of FRedit’s features, using a quick-and-easy method: https://youtu.be/DfAGD9RCpNQ

FReditListChecker – Check your FRedit list for some of the ‘obvious’ errors: https://youtu.be/Z7cjf446JWM

Alternatively, try *FReditSimple*, below (also *MiniFRedit*).

When I very first started freelance editing, I noticed that each time I opened a new file, ready to edit it, I was doing a number of global find and replaces – double space to single, spaced hyphen to spaced en dash, ‘et al’ to ‘*et al*.’ and so on. I thought to myself, ‘Wouldn’t it be good if I could create a list of these changes, in a Word file and then run the F&Rs automatically? It would save me typing out the same F&Rs over and over again.’ And I realised too that this would be especially useful with a multifile job where I would want to do exactly the same set of F&Rs on each file, and wouldn’t want to forget any of them.

A friend who was then ‘into’ macros said that he could write something that would do just that, and ‘*FRedit*’ was born – ‘**F**ind and **R**eplace **edit**’.

Then, as I too have learnt how to program macros, *FRedit*’s facilities have got more and more sophisticated, and now I simply couldn’t imagine doing an editing job without it. It saves me time, and increases the consistency of my edited output.

Importantly, you can use *FRedit* with Word’s wildcard searching facility, and the fact that you don’t have to (remember or) type out those arcane strings of symbols (e.g. Find ([0-9]) ^= ([0-9]) and Replace \1^=\2) is a real boon. Where it becomes *really* powerful is that you can do a *series* of linked wildcard F&Rs. But don’t panic: you don’t have to work these things out for yourself. Other people have done the hard work – you just copy their F&Rs and paste them into your own lists of F&Rs and use them.

You may also be worried that by doing global F&Rs, you might change things that you didn’t intend to. **You are absolutely right to worry!** *FRedit* is an extremely dangerous tool if used ill-advisedly, so don’t try to do too much with it too soon. As with any dangerous tool – a sharp cooking knife or a circular saw – your skill and confidence builds the more you use it. But, in the same way that many tools have safety features that you can use (or choose not to use as your skill grows), so does *FRedit*. You can use tracked changes (but you may know that it too has its dangers!) and/or you can choose to add a coloured highlight (any colour you want) to any or all of the changes that *FRedit* makes – so at least you can see what it has changed.

The *FRedit* files, including the instruction manual, are downloadable from my website at (http://www.archivepub.co.uk/documents/*FRedit*.zip). These include full instructions as to how to install the *FRedit* macro and how to use it. There’s also a library of ideas of things that you can do with *FRedit* plus a section containing hints and tips that will help you to build up your efficiency in using *FRedit*.

The *FRedit* macro listing is ***only*** given here to make it easier if you want to update to the latest version. **Please do not try to use *FRedit* unless and until you have read the instruction manual** – you wouldn’t buy a chainsaw and start to use it without at least reading the instructions, would you?!

N.B. *FRedit* now works on either the whole text, or just the selected text

**Sub\_ FRedit()**

### Paragraph problems (PLEASE READ THIS)

*(This is an extract from the FRedit manual, but I wanted to put this warning somewhere in this book and couldn’t think where else to put it. Because it’s about F&R, this seemed as good a place as any.)*

It’s important to realise that Word is ***very*** fussy about the use of ^p and ^13 in find and replace. Here are the rules you must obey:

1) Never use ^p in a ***wildcard*** ***Find*** – only ever use ^13. (In fact, *FRedit* will see this and will warn you that what you have done is wrong.)

2) Never ***ever*** use ^13 in any ***Replace*** – only ever use ^p.

The reason for (2) is that the ‘^p’ is the thing that holds the formatting information about the paragraph. If you use ‘^13’, you’re saying, ‘create the start of a new paragraph, but give the paragraph the same style/format as the paragraph that follows.’

## FRedit Hint – Switch Track Changes On

On the final (long) chapter of my last job, I forgot to switch on track changes before running *FRedit*, and I didn't notice until I got well into the editing. That caused me a *lot* of hassle because important changes went untracked. Arrgghhh!

Not any more! *FRedit* has a facility such that I can add a line at the top of the *FRedit* list:

| Track = yes

If track changes is off, it opens a window to warn me that it’s off, and it won’t let *FRedit* go ahead until track changes is switched on.

## FRedit Hint – Checking your FRedit List

*(Video:https://youtu.be/Z7cjf446JWM )*

It’s all too easy when creating a FRedit list to introduce unintended styles and font size/name changes. They might not be obvious to the naked eye, so this macro checks the list for any funny styles and font sizes/names. They may be deliberate on your part, which is fine, but if so just click and move on.

The macro also warns you about lines that don’t have a pad character. That, too, can be deliberate, if you’re using two-line F&Rs for style changes, but again, at least you’re prompted to check.

The macro starts its checking from the current line, and stops at the first possible problem.

**Sub\_ FReditListChecker()**

## Scripted F&R – Simplified Version of FRedit

(It’s a bit like a *FRedit* trainer – a flight simulator for a trainee pilot, though you can actually use it to do some useful jobs.)

For this simplified version of *FRedit*, all you need is a list like this, of the words ***you*** want to change:

That has to be in one Word file, open on screen, then open the file on which to make these global changes; then run the macro.

***That’s it! That’s all you need to know.***

(However, if you’re feeling brave and want to try other things, keep going, and I’ll drop in some learning points [LP].)

Copy the following list and paste it into your ‘list’ file. Then find an old file of text to “play with” and then try to change it by using this list and see what happens:

that|that

which|which

the|the

[LP: Changing something with the ***same thing*** has no effect, except to add a highlight.]

Then you could try and see what this does:

¬the|the

Can you see what that did? It did something that the|the ***didn’t*** do.

[LP: The “¬” character says “change text regardless of its case; upper or lower”.]

Still feeling brave? OK, try copy-and-pasting this list:

| Double space to single space

^32^32|^32

| spaced hyphen to spaced en dash

- | ^=^32

| Double return to single return

^p^p|^p

[LP: The “^32” is exactly equivalent to “ ”, but it’s easier to see! (32 is the ASCII code for a space.)]

[LP: Any line starting with a vertical bar is ignored by the macro.]

[LP: The “^p” is one of many Word codes. For more, click Ctrl-H: Find and Replace — More >> — Special.]

Feeling positively heroic? Then try this:

| A wildcard search for number ranges: hyphen to en dash

~([0-9])-([0-9])|\1^=\2

[LP: The “~” says the rest of this line is a wildcard F&R.]

[LP: If you find wildcards a bit scary, don’t worry, just copy other people’s – see the *FRedit* library.]

One final thing to try, but this time, switch track changes ON before running the macro:

| Double space to single space

~~|^32~~

| spaced hyphen to spaced en dash

- | ^=^32

| Double return to single return

^p^p|^p

| A wildcard search for number ranges: hyphen to en dash

~~~([0-9])-([0-9])|\1^=\2~~

[LP: Any line with the single strikethrough attribute added will ***not*** be tracked, even if track changes is ON.]

**Sub\_ FReditSimple()**

## Scripted F&R – A Simple FRedit-like Tool

*(Video:https://youtu.be/DfAGD9RCpNQ )*

This is an even simpler system than *FReditSimple*, and is aimed at doing a useful job, very quickly. However, it’s ***not*** aimed at training you to use *FRedit*, because it has a different way of working.

At the end of the file you want to work on, type a hash (#) followed by two or three blank lines. Select from the hash to the end of the file and make sure it is in Normal style and pure text. (I use the macro *NormalStyleAndChar*, but I assume there’s a way of doing it off the toolbar, though I can’t see where, sorry!)

On lines following the hash, you could do things like the following. See if you can guess what each will do, when I run the macro.

#

toolbar

*FRedit*

*et al*

-ise

-isi-

-our

-ment

The first highlights every occurrence of ‘toolbar’, the next two change the words to italic, but note that the third line will ***not*** do this: “The parap*et al*igns with the fe*et al*ong the edge.” it ***only*** applies the attribute to ‘et al’ as a separate pair of words.

Then, it’s highlighting words ***ending*** in ‘ise’, and ‘isi’ anywhere in the middle of a word, and ‘our’ and ‘ment’ at the ends of words.

Other features you can use are bold and font size, and they can be combined:

**hello**

and I’ve also added strikethrough:

~~bonjour~~

so that would stop ‘bonjour’ being thrown up as a spelling error by my spelling macros.

If you want to ***remove*** some effect, add an exclamation mark:

#

!toolbar

*!FRedit*

*!et al*

These would (1) remove the highlight and (2 + 3) remove the italic.

One possible application is to check where a word/words occur within a document – and here, I’d advise working on a copy of the document!

**FRedit**

**toolbar-**

(I’ve added the hyphen, so that it also catches ‘toolbars’.) Having run MiniFRedit, you can zoom right out, so that you can see lots of pages on screen at one go, and the occurrences of FRedit and toolbar will be clear to see.)

Programmer’s extra: If you want to add other attributes, such as superscript, it’s not difficult. Add a line to read the attribute into this list:

' Check the attributes on this item

myBold = tst.Font.Bold

myItal = tst.Font.Italic

mySize = tst.Font.Size

myStrike = tst.Font.StrikeThrough

mySuper = tst.Font.Superscript

and the add an item to apply it, such as:

If myStrike Then

.Replacement.Font.StrikeThrough = True

If doUndo Then .Replacement.Font.StrikeThrough = False

End If

If mySuper Then

.Replacement.Font.Superscript= True

If doUndo Then .Replacement.Font.Superscript= False

End If

(In fact, I thought I might as well add this anyway! So it now also does sub/superscript and underline!)

**Sub\_ MiniFRedit()**

## German and French quotes

If you try to do a find and replace to change English quotes (“ & ”) with German quotes („ & “) or French quotes (« & ») then you might find the F&Rs don’t work. If so, this is probably because you’ve got the automatic curly quotes function set.