09 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 *MultiFileText* to scrape the text out of all the files to present to the analysis macros.

Hint: If you're analysing large files, it would be a good idea to first run *CopyTextSimple* (or even *CopyTextVerySimple* for *ProperNounAlyse* or *HyphenAlyse*), in order to allow the macro to work on a file with little or no (in the case of *CopyTextVerySimple*) formatting for the macro to trip over.

To give you an idea of the sorts of times some of these analyses might take, and how much longer long files might take, here are some sample timings (in minutes):

File size	HyphAl	WpairAl	PNAl	DocAl	SpErList
× 1,000					
50	1	1	3	0.9	3.6
100	3	3	12	1.1	7
200	8	12	29	2.5	15
400	22	42	96	3.6	32

Run all your analyses at one go

If you want to run a set of analyses all at one go, perhaps doing so overnight, or at least doing it while you're making the supper of taking the dog for a long walk, then you can set this macro up to do your favourites.

Set up the first line of the macro for the set of analyses you want (in the order you want) and it does them.

```
myAlyses = "DocAlyse, HyphenAlyse, ProperNounAlyse, FullNameAlyse, SpellAlyse,
CapitAlyse, WordPairAlyse"
```

If you want to see the progress, open VBA and do a Ctrl-G to open the Immediate Window. There you will see a timed list of what time each macros started.

In theory, it could be used to run ***any*** macros. However, in practice, the macros you run have to be adjusted so that if they are being run from MegAlyse, they know not to open message boxing saying things like "Do you REALLY want to run this macro?!"

There are currently seven macros that you can use (and they must all be dated 23.10.19 or later). If you want others, please ask me – it only takes a few minutes to adjust each analysis macro to make it link in with MegAlyse.

The macro needs somewhere to store a temporary file. This is specified by the line:

```
myFolder = "C:\Documents and Settings\Paul\My Documents\"
```

Instructions for setting this up are the same as for the IZtoIS macro; see: 'IZ to IS spelling and vice versa', below.

Added feature: it now also saves the results files in your temporary directory, but that is controlled by:

```
saveResultsFiles = True
```

Make it `False` if you don't want it to save them automatically there.

`Sub MegAlyse()`

DocAlyse

Mac users: This macro **should** work OK on **most Macs**. However, if (and only if) you get an error saying "Macro too long, or some such, then please refer to the section "Mac users start here, please".

N.B. This macro **ONLY ANALYSES THE MAIN TEXT, NOT FOOTNOTES, ENDNOTES OR TEXTBOXES**. Please use CopyTextSimple first to include all the text.)

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 – see Note 1 below)
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 – see Note 2 below)
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

Note 1: The logic behind it is that we're taught that conventions used might be:

1) one to nine, plus 10, 11, etc upwards

2) one to ten, plus 11, 12, etc upwards

3) (more in humanities rather than in sciences) to ninety-nine, one hundred, 101, 102, etc.

So the tests are aimed at elucidating the authors predominant usage.

The tests I use for (1) vs. (2) are to count (whole word) "ten" and "10".

For (3), I use:

~<[efnst][efghinorvwx]{2,4}ty|^&
~<[efnst][efghinuorvwx]{2,4}teen>|^&
~<eleven>|^&

~<hundred>|^&
~<ten>|^&
~<twelve>|^&

The first of those would erroneously pick up “weighty”, and possibly others , but none of the tests is intended to be foolproof – they give a general indication.

Note 2: The ‘JLB Matekoni’ option can get exaggerated in number because it will pick up things like: ‘the US Department of Energy’ and think that ‘US Department’ is a person with un-full-stopped initials.

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.

A later addition was that it 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()
```

Most Mac users will be able to use the ordinary DocAlyse, and that gives the largest amount of data. However, if DocAlyse throws up errors about ‘Not enough memory’, please try the following reduced feature versions (in this order):

```
Sub DocAlyseForMac()
```

```
Sub DocAlyseForThinMac()
```

```
Sub DocAlyseForVeryThinMac()
```

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

```
Sub DocAlyseMedBits()
```

Like DocAlyse, but with user-selected targets

DocAlyse is, not surprisingly, geared towards English, but if you edit in other languages, you can do DocAlyse-like analyses in your chosen language by using this macro. The idea is that you choose a series of ‘things to count’, list them, and then run this macro. It goes through your list, counting all the word/phrases/part words etc., that you specify.

Here’s a dummy count list – it must start with ‘| CountIt’, so the macro knows which file holds your DocAlyse list (highlighted only for the explanation):

```
| CountIt

| This is a comment
the (LC only)|<the>
The (Initial cap)|<The>
the (any case, whole word)|¬<the>
the (any case)|¬the

et al|<et al>[!.]
et al.|<et al.

|| These lines are disabled
||sausage|¬sausage
```

And here’s a sample output using the above (dummy) DocAlyse list.

DocAlyseUser

	This is a comment
the (LC only)	80
The (Initial cap)	9
the (any case, whole word)	89
the (any case)	113
et al	–
et al.	–

Notes

1) For a line like ‘the (LC only)|<the>’, the green bit is what appears in the final analysis output, and the yellow is what is counted.

2) As you can see, ‘<the>’ is a wildcard find (the macro recognises ‘<’, ‘>’, ‘[‘, etc and switches wildcards on).

3) With ‘the (LC only)|¬the’, the macro sees the ‘bent pipe’ and switches to ‘any case’.

4) With ‘the (any case, whole word)|¬<the>’, the macro sees the ‘bent pipe’ and switches to ‘any case’. The ‘¬<the>’ isn’t a ‘proper’ wildcard Find, since wildcards are, by definition, case sensitive. This therefore is the macro using a fudge. It changes the Find to ‘<[tT][hH][eE]>’ to catch all possibilities.

5) Note that the counts for ‘*LC only*’ plus, ‘*Initial cap*’ do NOT equal those for ‘*any case*’ (80 + 9 ≠ 113). That’s because ‘¬the’ is not whole-word limited, so it will also count ‘o**ther**’, an**the**m’, etc.

6) Any line starting with '||' is totally ignored in the counting process.

7) Lines starting with just '|' are 'comments', i.e. they are reproduced in the final analysis output (as are blank lines).

8) As with DocAlyse, any counts that produce a zero result are greyed out, so as not to distract the eye.

General plea: If you create a DocAlyse list file for some particular language, please would you consider making it available to other editors? If you send yours in to me, I'll put it on my website, for others to use. Thank you!

Sub DocAlyseUser()

Counting numbers as figures or as text

DocAlyse only counts '10' or 'ten', plus spelt-out numbers from 'eleven' upwards, so if you want more information, this macro produces a result such as:

Numbers as digits	21
Spelt-out numbers (one-nine)	13
Spelt-out numbers (ten)	2
Spelt-out numbers (eleven etc.)	15

When counting numbers as digits, the macro allows you to specify the max number of digits to be counted: one only, one or two or one to three digit. This is set with `maxFigs = 3` (or less) at the beginning of the macro.

It also tries to avoid, e.g. '22 September 1948' or '12 hours'. This is set in:

```
noCount = "Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec,"
noCount = noCount & "hou,min,sec,day,"
```

You can add words or subtract 'units' not to be counted, but it only counts numbers followed by a space, so it would ignore 22/9/48. (Not an infallible test, I know.)

Sub NumberTextFigureCount()

Analyse serial (or not) commas

DocAlyse will give you an approximate indication of whether the lists in the document have serial commas or not, but all it only checks lists that are made up of single words (fish, chips and peas), but it would not count "smelly fish, greasy chips and mushy peas"!

In any case, what is a serial (or not) comma? How about "I'd like some fish, please, and no peas!"

How could a computer tell the difference? Too difficult for macros.

So if DocAlyse's count is too close to call, you have to search through for " and ", and read around each occurence, to decide whether or not each is a list and, if so, put a tick in your 'Serial' column or your 'Not Serial' column.

To speed up the process, I've written a macro that takes an area of text (the whole text or a selection), copies it out into a new file, and then highlights in green every "**<word>, and**" and in yellow every "**<word> and**", and then it does the same for the "**<word>, or**" (green) and "**<word> or**" (yellow).

You can then use the *FindHighlight* macro to jump through, assessing them one by one, and putting ticks in your 'Serial' column or your 'Not Serial' column.

To offer more support, the macro also does the DocAlyse type assessment, checking single-word lists, and it reports a "Rough indication:", asking you whether you then want it to continue and create a checking-file for you to trawl through.

```
Sub SerialCommaAlyse()
```

Analyse different formatting of centuries

(Video: 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()
```

Analyse format of days and months

Someone wanted to know how often an author had used Mon, Tues, Wed(s) and Jan, Feb, Mar, compared to spelling them out. I realised that, as they are all proper nouns, ProperNounAlyse already gives you the frequencies of all these words in its proper noun frequency list (as opposed to its proper noun queries list).

This macro simply looks through that list and reports what it finds, ‘greying out’ words that aren’t used at all, e.g.

Mon . . . 2
Tue . . . 1
Wed

Weds . . . 4
Thu
Thurs
Fri . . . 4
Sat
Sun

Jan . . . 4
Feb . . . 4
Mar . . . 6
Apr . . . 4
May . . . 7
Jun . . . 4
Jul . . . 4
Aug . . . 4
Sep . . . 4
Sept
Oct . . . 4
Nov . . . 4
Dec . . . 10

January . . . 20
February . . . 10
March . . . 9
April . . . 12
May . . . 7
June . . . 10
July . . . 13
August . . . 10
September . . . 10
October . . . 9
November . . . 9
December . . . 9

So, to get this macro to do its tests...

1) Run ProperNounAlyse, but make sure it's set to check words of three or more letters. At the beginning of the ProperNounAlyse macro, it must say:

```
minLengthCheck = 3
```

2) Of the two files that ProperNounAlyse generates, ignore the 'Proper noun queries' file and click in the 'Proper noun list' file instead.

3) Now run DayDateAlyse.

If you think of other proper nouns you want it to report on, you can just add the words into your copy of the macro:

```
myWds = " Mon Tue Tues Wed Weds Thu Thurs Fri Sat Sun X "
myWds = myWds & " Jan Feb Mar Apr May Jun Jul Aug Sep Sept Oct Nov Dec X "
myWds = myWds & " January February March April May June "
myWds = myWds & " July August September October November December "

myWds = myWds & " X Pete Peter Jennifer Jenny Jennie Albert Bert Bertie "
```

Just ensure that you follow the same format when adding extra items. The 'X's are just a way to put a blank line into the final list to make it clearer.

```
Sub DayDateAlyse()
```

Analyse different types of lists

To be fair, that's a bit of an overstatement. This macro doesn't actually *analyse* the lists – that would be a pretty difficult task for a macro. Rather it runs through the whole text, copying all of the lists into a separate document. Then you have one succinct document that you can scan through, so that *you* can analyse the different types of lists.

Importantly, it also includes the paragraph immediately *preceding* a given list; in other words, it gives you the text that *introduces* the list. You can then compare the different ways in which the coming of each list is heralded by the author.

(N.B. This macro will doubtless miss some of the types of lists in your document; if so, please could you email me a sample of the list it failed to recognise. Thanks.)

```
Sub ListAlyse()
```

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

(Video: youtu.be/_fWD4sXNg5s)

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.

1 – Font size
2 – Font name
3 – Style
4 – Bold/italic
5 – Bold/italic not in a style
6 – Super/subscript
7 – Diacritics
8 – Various non-alpha characters (slow)
9 – Funny spaces
10 – Funny Symbol fonts

Once you've used one of these options to highlight something, you can use *HighlightFindDown* and *HighlightFindUp* 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 *HighlightFindDown*, 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.

1 – Font size highlights anything in a font size different from the Normal style.

2 – Font name highlights anything in a font other than the Normal font, e.g. in Arial, when the Normal font is Times New Roman.

3 – Style highlights any text that has a specific style applied, e.g. Heading 1.

4 – Bold/italic highlights anything in bold, italic or bold-italic, and its companion...

5 – Bold/italic not in a style does the same but then removes the highlighting from text in a specific style. This is aimed at cases where the body text is in Normal, but the headings are formatted using styles, e.g. Heading 1.

6 – Super/subscript is pretty obvious.

7 – Diacritics highlights any accented character, such as, ÄÃãÅåçÇéÉèÈêÊëËíÍìÌ.

8 – This aims to highlight any 'different' characters that might have been used. It works by first highlighting the whole text and then unhighlighting all alpha and numerics, and then unhighlighting any 'obvious' punctuation marks. You can specify any extra characters that you **don't** want to be highlighted by using:

```
' Other characters NOT to highlight on Option 8
notThese = "\[\]\*\@" & "_+=&%"
```

This option is a bit slow, so I've made it give you a beep to let you know when it has finished.

9 – There are a range of Unicode characters for different width spaces: en, em, thin, hairline etc. However, when some of these spaces are on the end of a line, the highlighting is invisible. Therefore I've added a marker to force Word to display the highlight. These temporary markers are removed next time you run the macro. You can use a different mark, if you prefer. The marker is specified at the beginning of the macro:

```
myMarker = "||"
```

10 – Funny Symbol fonts are what Microsoft gave us before Unicode was widely accepted, and they can be a bit of a nightmare, so it's useful to know if your text contains any.

```
Sub FormatAlyse()
```

Count words that appear as both singular and plural

This macro finds and makes a frequency list of all the words that appear in both singular and plural form. It copes with 'box' and 'boxes' but not 'child' and 'children'.

Then you can colour all occurrences of the words in your list, with the second macro.

```
Sub PlurAlyse()
```

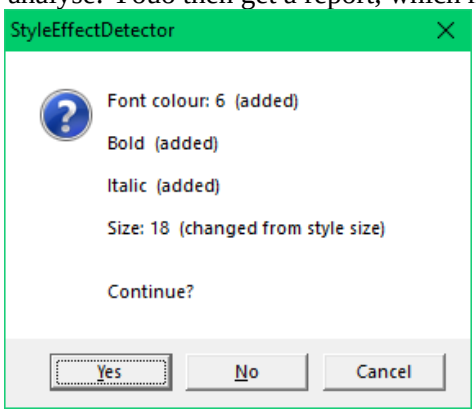
```
Sub PlurAlyseColour()
```

Examine formatting and applied effects and patterns

(Video:youtu.be/1NnppLuNyuE)
There might be some way in which you can do this via some menu(s) somewhere within Word, but if so I don't know where. The idea is that, as you look at a bit of text, you think, "Is that a style? If so, what has been added 'manually' added to this text?"

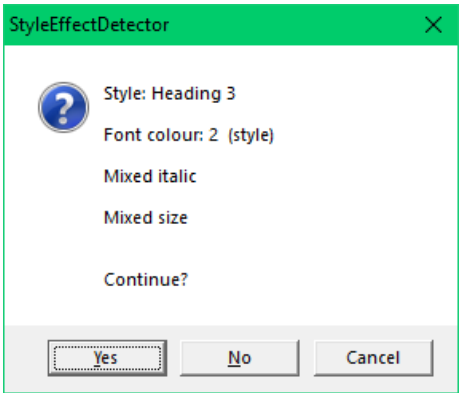
So the style might be 14 point Ariel bold, but the author might have made it bold-italic for some reason, and/or they might have upped the font size to 15. And what about the highlight here? Is it really highlighting? Or is it actually a background pattern? (Try removing the 'highlighting' from the latter turqouise words. You can't! Well, you can, but you'll need to find the appropriate menu for patterns, or use the *BackgroundColourOff* macro.)

So with the *StyleEffectDetector* macro, you either select some text, or just place the cursor somewhere in the area of text whose formatting you are trying to analyse. Youo then get a report, which might say:



If you click 'Yes' (or press Enter), the selection more one space to the right, and reports on that single character; if you click 'No', it moves one character to the left, and to escape from the macro, click 'Cancel', or click in the Close icon, top right.

If your selection contains, say, text of different added font size, or some added bold and some not, etc., it will tell you the selection is 'Mixed':




```
Sub StyleEffectDetector()
```

Spelling analysis

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

N.B. The latest macro, SpellAlyse is not included in the summary (sorry, I haven't got time), but it should be your first port of call for spelling as it is so powerful. PB.

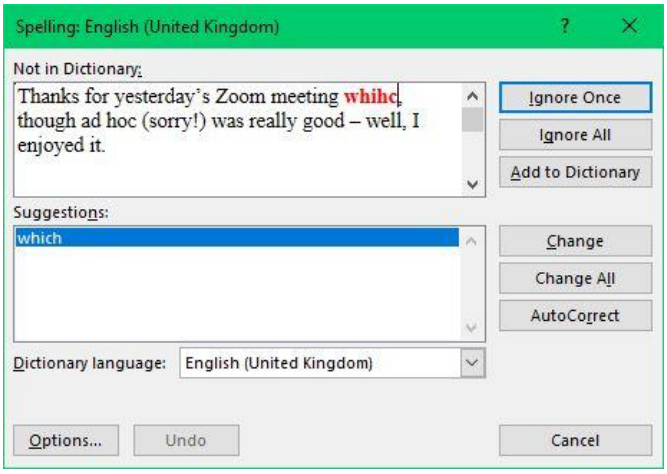
UKUScount – Has the author predominantly used UK or US spelling?
IZIScount – For UK spelling, has the author predominantly used -is- or -iz- spellings?
(For editing, you can use *IStoIZ* and *IZtoIS* to implement your decision.)
SpellAlyse – Complete spellchecking system
ProperNounAlyse – Alerts you 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/cannot).
AccentAlyse – Compares words that use the same letters, but with different accents.
AccentedWordCollector – Collects all the accented words in a text
ConcordanceMaker – Creates a concordance list
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, OUPFetchPremium, PubMedFetch, ThesaurusFetch, WikiFetch – Looks up the current word on the relevant website.
LanguageSetUS, LanguageSetUK – Sets language for whole document.

High-speed spellchecking

Just a hint first...

Are you missing Word's old spell-checker?



Do you dislike Word's new, whizzy 'Editor', with its 'I know best' attitude?! I have good news for you!

Open up the 'Customize the Ribbon' (by right-clicking on the ribbon), then click 'Keyboard shortcuts... Customize'.

Press the A-key twice to bring the LH column to 'All commands'.

Click in the RH column and press the T-key and scroll down to 'ToolsSpelling', and click on it.
Click in the 'Press new shortcut key:' and press the usual F7 key.
It will say that it's assigned to ToolsProofing, but click 'Assign', then 'Close'.
Now the 'proper' spellcheck will be back.
Enjoy!

(Is it the same on a Mac? If not, please would someone explain what to do? Thanks.)

Now the macro(s)...

(Video: https://youtu.be/JPJBF6mHq2M)

This system can be used either for editing or for proofreading.

If you want the highest speed of detection and correction (and rechecking) of spelling errors, then you will need first to learn the basic functionality of *FRedit*.

Even if you don't use *FRedit*, it can still be used for proofreading, to alert you what spelling errors to look out for. (This will be explained at the end of this section.)

The other macro you will need for best effect is *SpellingSuggest*.

The basic method is as follows:

1) Run *SpellAlyse*:

This generates an alphabetic list of "spelling errors", according to your (Microsoft's) spellchecker. (The list is in two sections: lowercase-only words (i.e. those more likely to be **real** errors) and words with one or more capitals, which are less likely to be errors, since many will be proper nouns.

N.B. A new feature added in May 2023, is that if any of the text has a strike-through attribute applied, ~~loike thiis~~, then any spelling errors in those words will be ignored.

If, when you run *SpellAlyse*, you have an exceptions list file(s) open (see below) it will not list as "errors" any of the words in the list(s).

2) Your job now is to work out which words in the errors list really are errors and which are exceptions – perhaps they are proper nouns or subject-specific terms.

To help you in this process you will see that *SpellAlyse* has also produced a second version of the errors list – with the frequencies of occurrence of each of the various words; this often gives a clue as to the nature of the "error".

(Hint: Many editors use *ProperNounAlyse* before they use *SpellAlyse* because this alerts you to any misspelt proper nouns. This will make this stage of the job easier.)

If a given word **is** an error, you can use *SpellingSuggest*, which will automatically generate an item for use with *FRedit*: e.g. if you're working through the list...

color|colour
colors|colours
colored
covariational
dangrous

...and you click in 'colored' and run *SpellingSuggest* and it will become:

color|colour
colors|colours
colored|coloured
covariational
dangrous

(only highlighted here to show what has changed)

If a word is an exception (e.g. "covariational"), just skip down to the next word.

If you decide that a word needs checking in context, as you read to chapters, you can draw your attention to it by adding a font colour (with *ColourPlus*/*Minus*) or a highlight colour (*HighlightPlus*/*Minus*). You'll see why in (3).

3) With the cursor in the errors list (not in the original text), run *SpellAlyse* again. Sensing that this is a list, not an ordinary text, *SpellAlyse* will go through your annotated errors list and generate (a) a FRedit list and (b) a separate exceptions list.

4) During the editing process, you would add those FRedit list items to any other items you haeve, maybe for punctuation, hyphenation, proper noun errors, etc.

5) At the end of the job, use *MultiFileText* to generate a file of the whole book, open the exceptions list as generated in (3) and run *SpellAlyse*. This will generate a list just of any spelling errors that still remain, or that you have added, when typing in your changes – this re-checking can save your blushes!

If you're wanting to use *SpellAlyse* when doing proofreading, load the PDF into Word and run the macro. Then you can look down the errors list, deciding which words really are errors; to these you can add a font colour, or to make it stand out more, a highlight such as bright green.

If you now run *SpellingErrorHighlighter*, all those errors will be highlighted, and after reading a section of the PDF, you can that you haven't missed any of the spelling errors.

(N.B. If the job is in multiple PDFs, as long as you have Word 365, the latest version of *MultiFileText* macro can combine the **PDFs** into a single **Word** file for you.)

If the author has used a user dictionary (e.g. for technical documents or (speculative) fiction), you can get the author to send you their user dictionary, you just load it as a Word file and when you run SpellAlyse for the first time, it will recognise it as an exceptions list and so it will only list the non-user-dictionary 'spelling errors'.

If you use Open, in Word, to open the dictionary, just tell it to use 'Other encoding', and you'll get a word list for your exceptions list for SpellAlyse.

In fact, *SpellAlyse* has **one more feature**, though it's probably rarely going to be useful. It will *only* be relevant on a job that has a good number of spelling errors but very few special words.

If you run *SpellAlyse* on the spelling error list *before* you start creating *FRedit* items, it will offer to go through all the lowercase spelling errors and convert them all to *FRedit* items, stopping only when it gets to the mixed-case words. You would then need to check through and correct any that weren't appropriate.

The macro will suggest that you switch track changes on for this. If the changes have been tracked, then you can revert any change that was inappropriate.

The macro has the option to ignore numbers, or to "spellcheck" words that include numbers, e.g. 'BS5261'.

This is set by the line:

```
ignoreNumbers = True
```

or False.

```
Sub SpellAlyse()
```

```
Sub SpellingSuggest()
```

N.B. The latter has an extra feature: if the word you ask it to check is, say, "THankfully" (i.e. you mistyped it) then it corrects the mis-capitalisation for you. (Yes, I know there's an option in Word to do that by auto-correct, but I don't want it to 'correct' me when I type 'FRedit' to make it 'Fredit'!)

P.S. If you use this macro a lot, by putting it on, say, F8, it's even quicker than using, say, Alt-S. Yes, I'm a speed-freak, I know!

Check/correct current spelling

This macro allows you to immediately correct the spelling of the correct word. Click in the word, run the macro and the spelling is corrected. If the spelling is already correct, the macro bongs, to confirm the correctness of the spelling.

It will also correct PAul's mistypings. BRilliant, as I do it a lot!

```
Sub SpellingCorrect()
```

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 spelling errors

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 *SpellAlyse*. For proofreading, you can then use it 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.

High speed spellchecking

This system is intended for those who want the highest speed of detection and correct (and rechecking) of spelling errors, i.e. it was designed to work with *FRedit*. But even if you don't use *FRedit*, it can still be used to alert you to spelling errors, and also to highlight those spelling errors in your text, if you combine it with *SpellingErrorHighlighter*. (This limited – but still effective – use will be explained at the end of the section.)

The other macro you will need for best effect is *SpellingSuggest*.

Spellchecking for proofreading

(Video: youtu.be/6F_yT1MIW_Q)
(Video: youtu.be/iESM6OaGBm4)

[An important update to the *SpellingErrorLister* macro (Oct 2019) is that you can now put a list of 'spelling errors' (according to Word), at the end of the document, that are actually 'OKwords', as far as this document is concerned:

OKwords
abelian
bijection
cohomological
etc.

(They don't need to be in alphabetic order.)

As the macro checks the spellings of each word, it also checks in the OKwords list. (note: there's no space in 'OKwords'). This is really useful for doing a repeat spellcheck at the end of a job. It also works well with *PDFHyphenRemover + PDFHyphenChecker*, which automatically generates an OKwords list.]

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 'cohomological' and 'bijection' 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. (And then you might want to use *PDFHyphenRemover + PDFHyphenChecker*.)

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**.)

(If your keyboard doesn't have a Break key, you can still stop a macro mid-program. If you run the macro with the VBA window open and visible on screen, then you can use the stop '■' icon to stop the macro running. STOP PRESS! I've just discovered that, while a macro is running, yes, don't move the mouse, but you can use the keyboard – press Alt-F11, VBA will then open, and you can press pause '‖' or stop '■'.)

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

SpellingErrors

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):

SpellingErrors

acteylene
adjoint
analyze
analyzed
castilated
cill
clearnd
contractural
crainage
cranage
crosswall
crosswalls
develpments

Highlighting the errors – If you save the spelling errors list by using F12, it will offer the filename 'SpellingErrors', so save it with that name and then run *SpellingErrorHighlighter*, it will look for an open file with the filename 'SpellingErrors' and then work its way down the list, highlighting all those words in you file in the same colours that you have used in the list.

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 *MultiFileText* – 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: B¨assler, Br´edas, It^o 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, 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()

Spellchecking for dual languages

(*https://youtu.be/-9UELiY7ZJk*)

Having talked to some translators, they expressed the problem of handling two languages in the same document – how do you spellcheck them?!

Solution 1: If the areas of text in each of the two languages are distinct, you can apply a strike-through to the text in one language and spellcheck in the other language using *SpellingErrorLister* and *SpellingErrorHighlighter*, and then reverse the process for the other language.

Solution 2: This is not infallible, but I have produced a version of *SpellingErrorLister*, that is set up so that it checks the spelling in the two chosen languages, and it only lists the word as a spelling error if it is an error in both languages.

Can you see the flaw?

Suppose there's a French phrase, "dans votre propre pays" which appears as "dans votre proper pays" then the macro will *not* report it as a spelling error, because "proper" is OK in English.

But hopefully the macro will still help you to spot *most* of the spelling errors, and it will still save a lot of time.

(If you need the same thing for three language, please let me know, and I'll extend it.)

Sub SpellingErrorListerBilingual()

Checking for misspelt proper nouns

(Video: youtu.be/JOTUvQAu-uo and youtu.be/PB0hXA_1tRo)
*(**Latest video**: https://youtu.be/-kFHpFY6AV4)*

Hint: If you're analysing large files, it would be a good idea to first run *CopyTextSimple* (or even *CopyTextVerySimple*), in order to allow the macro to work on a file with little (or no) formatting for the macro to trip over.

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 one another.

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, *Barnham* 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.

(In the lists, **what are the highlights, italic etc for**? Read on...)

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 – the macro is complex enough as it is! (This was probably my most tricky-to-write macro ever.)

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 these from one of my earliest uses of this program:

Brosseau . . . 3
Brousseau . . . 2

LeJeune . . . 4
Lejeune . . . 1

Norwich . . . 5
Nowrich . . . 1

Shirioshi . . . 1
Shiroishi . . . 1

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 mightily impressed when you notice such things – but they don't need to know that it wasn't actually *you* that picked them up! :-)

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, *Barnham* 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.

I've also now, in addition to the total list of all the proper nouns, produced a cut-down list, where it's easier to spot the pairs (the added comments give an indication of which items are paired with which – as also indicated by their highlight colours):

	AAnAlyse . . . 2	= H	The pair that goes with this is...
	Aims . . . 1	= E	
4 =	Alexander . . . 3	= D	
4 =	Alexandra . . . 1	= D	
	Also . . . 2	= G	
5 =	Altand . . . 1		
5 =	Altland . . . 2		
	Alyse . . . 1	= G	
	Amis . . . 1	= E	
	Analyse . . . 1	= H	...right down here, but the highlight , and letter 'F', are pointers for you.
*	Angquist . . . 3	= F	
*	Ängquist . . . 1	= F	
	Arial . . . 1	= A	
	Ariel . . . 4	= A	
	Bald . . . 1	= H	And the pair of this one...
	Barford . . . 4		
	BARFORD . . . 2		
*	Belanger . . . 2		
*	Bélanger . . . 1		
	Black . . . 5	= E	
1 =	Blah . . . 1		

1 = Blau . . . 1
Block . . . 2 = E
Bold . . . 3 = H ... is down here.

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 – the macro is complex enough as it is! (This was probably my most tricky-to-write macro ever.)

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 even moving the mouse over the active windows while a macro is running can cause the macro to paste the text that 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.

Options:

```
includeAcronyms = True
```

This means this it will spot, say, an OECD/OCED error. But if your text has a lot of headings in all capitals, these could be quite a distraction, so you could change this to `False`.

```
minLengthCheck = 4
```

The aim of is option should be obvious, but I've never had a job full of 'four-letter words' that might have been a distraction, but you could increase (or decrease it).

Problem with 'funny' vowels

If the diacritics in your document have unicode values of more than 255, such as ā, here's a workaround:

1) Use CopyTextSimple to create a pure-text version of your document (that's good practice, anyway).

2) Run this FRedit list:

ā|æ
Ā|Æ

3) Run ProperNounAlyse.

So, if you have a word such as Theravada/Theravāda then it will be listed in the query list as:

Theravada . . . 1
Theravæda . . . 4

And if the text has other 'funny' vowels, i.e. with unicode values greater than 255 (you can check by using WhatChar), you'll have to extend your FRedit list, choosing any other sub-256 vowel instead.

```
Sub ProperNounAlyse()
```

Checking for spelling variants of 'special' words

This macro is rather like ProperNounAlyse; indeed, it uses that macro, so it won't work unless you have ProperNounAlyse loaded in your Normal template.

The idea came from an editor of science fiction, where they had a number of made-up words, and wanted to check that the spelling was consistent throughout the book. However, this same macro could possibly be applied to detect spelling variations in technical words, such as chemical compounds.

The principle is to collect all words not contained in Word's own dictionary (i.e. "spelling errors") and turn them into "proper nouns" and then run ProperNounAlyse, which would then points up possible spelling variations in these "proper nouns".

The macro allows you to set the minimum word length that you want tested:

```
minLen = 5
```

```
Sub SpecialWordSpellAlyse()
```

Frequency list of full names

(Video: 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()
```

Preferred spelling analysis

If you have a list, in a Word file, of various words where you have a preferred spelling, this macro will check through all the words in the document, to try to find words that might be alternative spellings, which will then need correcting.

You can have different lists for different clients, so just open the list you want for the current job, click in the target document and run the macro. (After a considerable time, possibly) it will produce a list like this:

achilies . . . 1	(achilles)
aegues . . . 1	(aegis)
arachne . . . 4	(archon)
argonuats . . . 1	(argonites)
asopus . . . 4	(asopos)
athana . . . 1	(aether)
baleneion . . . 1	(balaneion)
change . . . 4	(changer)
chimeera . . . 1	(chimera)
chiron . . . 1	(cretan)

Some will indeed be spellings you will need to change, but (as with ProperNounAlyse) it will thrown up 'finds' that you will know you can ignore.

N.B. This macro actually uses the code of ProperNounAlyse to run, and so you need to ensure that you have, in VBA, a copy of ProperNounAlyse dated Feb 2024 or later.

Alternative approach ***(my preferred approach!)***

Personally, the approach I would use is a very long FRedit list with corrections of all the possible spellings that I might ever want to change (for a given client):

Beverly|Beverley
Boadicea|Boudica
¬burnt|burned
¬formulae|formulas
¬porage|porridge
¬radii|radiuses
¬yoghurt|yogurt

The list can be as long as you like. It might mean that FRedit takes 30 seconds to run, instead of 2 seconds, but finding preferred words from a list seems quite time-consuming to me.

```
Sub PreferredSpellingsAlyse()
```

Checking the abbreviation of genus+species names

This macro looks through all the genus+species names (if they are in italic) and checks that the first occurrence is full, not abbreviated, and that thereafter they are abbreviated.

If they are OK, they get a grey colour, but if they are in error then they are either yellow or green, as a warning.

However, if you set `doAbbreviate = True` at the beginning of the macro, then it also abbreviates them for you.

Also, if it spots an abbreviated species it hasn't met yet, it highlights it in yellow, but it also remembers it. Then, when the full version of that same species appears, it gives it another warning colour (red) because this then has to be related (by you!) to its yellow equivalent.

```
Sub GenusSpeciesAlyse()
```

Finding similarly spelt words

Run this macro (it takes quite a long time) and it searches through all the words in the file, seeking to find words that might be alternative spellings of one another:

goddeses . . . 1 = G

1 = given . . . 4
1 = gives . . . 2
goddesses . . . 1 = G
1 = greek . . . 1
1 = green . . . 2
hesiad . . . 1 = F
hesiod . . . 1 = F
1 = highlighted . . . 6
1 = highlighter . . . 1
1 = humor . . . 1 = A
1 = humour . . . 1 = A
7 = operat . . . 6 = G
7 = operate . . . 1 = G
paseidan . . . 1 = F
poseidon . . . 1 = F

I'm not sure how effective it is in your workflow, but it's there if you want to try it out. As with ProperNounAlyse (on which it is based), just click in the document under test and run the macro… then go and walk the dog!

SimilarWordsAlyse()

Variations in capitalisation

(Video:youtu.be/r6PKbwbbSHo)

The *CapitAlyse* macro lists and finds the frequency of all the words that occur sometimes with an initial capital, and sometimes not. It was aimed at the jobs I do for the construction industry, where authors tend to refer to 'the Architect' and 'the Manager' (but never 'the Bricklayer'!), so it's helpful to know how often they use each form, so I can decide how best to impose consistency.

Yes, you say, *but what about the initial capital that occurs at the beginning of every sentence?!* I've tried to take account of that, as you can see in the extract below of some of the results. The '(+4)' on 'Ancient' means that the word also appears four times as the lead word in a sentence.

Yes, you say, *but then what about all the initial capitals that occurs in headings?!* I've tried to take account of that too by deliberately ignoring those words that occur in bold and/or italic. OK, this is a rather inexact science, but it's a difficult attribute to analyse, and I've done the best I can. It's up to you, to look through the list, see if there's anything that might be significant, and then use *FindSamePlace* to jump straight from the list to the first occurrence of that word in the text.

The other thing I've done to try to reduce the number of insignificant words is to list some words at the beginning of the macro that are to be totally ignored (and you can add to that list, of course):

```
ignoreWords = ",After,All,Although,Also,An,And,As,At,By,For,From,If,In,It,"
ignoreWords = ignoreWords & "Of,On,Our,The,This,Those,There,These,They,Up,We,"
```

Looking through the list (especially with hindsight) I can say, "Aha! it looks as if I've got references to both 'Bronze Age' and 'bronze age'!"

age . . 3
Age . . 4

ancient . . 18
Ancient . . 18 (+4)

aqueducts . . 1
Aqueducts . . 1

assessment . . 4
Assessment . . 1

authority . . 10
Authority . . 1

baray . . 3
Baray . . 3

book . . 1
Book . . 2

bronze . . 2
Bronze . . 4

century . . 4
Century . . 1

chapter . . 11
Chapter . . 3

The time taken for any document will depend not only on the total number of words, but also on the number of different capitalised words that need to be counted, but the times below might give you some idea what to expect.

If you want to watch progress, there's a line on the status bar, but it's not easy to see, so if you open VBA and open the Immediate Window with Ctrl-G, the indication there is much clearer.

27 kwords	0.4 min
54 kwords	0.9 min
108 kwords	5.5 min
210 kwords	36 min

```
Sub CapitAlyse()
```

Find and count repeated phrases

(Video: 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()

Spellings with varying accents

(Video: https://youtu.be/h0oG3jWM8n8)

(This is, in a way, mis-named: by 'accent' I mean any alphabetic character other than the 26 'normal' characters.)

This macro draws your attention to any inconsistencies, where a word containing accents (or any other special characters you're interested it) is spelt differently in other parts of the document, e.g. facade/façade, cafe/café, déjà vu/deja vu.

Note that it doesn't list all accented characters (but see the following macro), only those that occur with *alternative* spellings, i.e. potential inconsistencies.

The macro generates a list like this:

cafe . . . 2
café . . . 6
deja . . . 1
déjà . . . 2
facade . . . 2
façade . . . 4
Lopez . . . 1
López . . . 3
México . . . 1
Mexico . . . 82
Monsivais . . . 3
Monsiváis . . . 1
Zlcalo . . . 1
Zócalo . . . 8

If you want to add any extra 'funny' characters, they can be added to this list at the beginning of the macro:

```
allAccents = "áÁàÀâÂäÄãÃåÅçÇéÉèÈêÊëËíÍìÌîÎñÑóòÒôÔöÖõÕøØßúÚùÙûÛüÜýÝÿŸğ"
```

Unfortunately, you can't just add characters such as 'ğ' (unicode 287) into this list (as I've done above). Because VBA doesn't recognise non-ASCII unicode characters, so I've now added an option to include a range of 'Central European' characters, including the 'ğ'.

```
Sub AccentAlyse()
```

List all words containing accents

(Video: https://youtu.be/h0oG3jWM8n8)

(This is, in a way, mis-named: by ‘accent’ I mean any alphabetic character other than the 26 ‘normal’ characters.)

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, but then it offers to remove duplicates, if you just want a list of which accented words it contains.

```
Sub AccentedWordCollector()
```

List all words = a concordance

This macro generates and alphabetic list of all the different words that occur in a document. It also, optionally sorts the list by length, so you can look at the longest or shortest words.

The options are:

```
minLength = 4
sortByLength = True
lengthAscending = False
```

The first sets the minimum word length that it looks for, so that in this case it only lists words of four or more characters.

The second asks whether or not also you want a separate list of the words, but in length order; and finally should that list be shortest first or longest first.

```
Sub ConcordanceMaker()
```

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*.

| To catch “the the” etc
~(<[a-zA-Z]{1,})[.,\!\?:;]{1,}\1[.,\!\?:;]|^&

| To catch “he said, he said” etc
~(<[a-zA-Z]{1,}^32[a-zA-Z]{1,})[.,\!\?:;]{1,}\1[.,\!\?:;]|^&

| To catch “as it were, as it were.” etc

~(<[a-zA-Z]{1,}^32[a-zA-Z]{1,}^32[a-zA-Z]{1,})[.,\!\?:;]{1,}\1[.,\!\?:;]|^&

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

(Video: https://youtu.be/d_koGeFckp8 and https://youtu.be/CwYiEpPMKhk)

(N.B. This macro is ONLY useable if you take the text out of your PDF by using the Select all–Copy–Paste method. If, instead, you create your Word version of the PDF by 'loading' the PDF directly into Word, the conversion, very helpfully, ignores page boundaries, in order not to create two paragraphs of any paragraphs that are split across pages. Useful! Except that this it means you ain't got any page numbers on which these techniques here depend!)

This macro searches for either page number 1 or a page number you type in from the keyboard; however, if you place the cursor at a particular number, it will offer you that number as default when asking you for the page number to start from.

At the beginning of the macro, you set whether the page numbers appear in the header (True) or the footer (False):

```
numbersInHeader = True
```

Also, to help it work out where to look for the page number, you tell it whether to look always to the right:

```
allNumbersAtRight = True
```

of change it to False, if the numbers are on alternate sides.

In most cases, according to convention, odd numbers are to the right, but if in your job the even numbers are to the right, change to:

```
evenNumbersAreOnLeft = False
```

Here's a sample of what you get:

2 48	hanidubeit 4 lc−1). Bsa gdib unan robo mhic bsa SEBHOT robopona (UN cirad, caot dobebura,
2 49	nuccah, otr zahi adavobeit) [2]. Bsa dopadn etreloba bsa cidaludan bsob oha hangitnepda mih
2 50	bhotncenneit reg et bsa lihhangitretw ngalbhod haweitn.
2 51	.ETRR 2 1/17/2020 11:03:18 OC
3 1	1.2 Bsa Fihdr'n Nalitr Donah 3
3 2	1.2 Bsa
3 3	Fihdr'n Nalitr Donah
3 4	Etbahanbetwdy, bsa fihdr'n nalitr donah – ombah bsa Coecot'n hupy donah – fon o
3 5	cer-etmhohar nider-nboba donah ponar it bhevodatb uhoteuc-rigar lodleuc mduihera
3 6	(U3+:LoM2) [3]. Eb fon igahobetw ob a fovadatwbs λ = 2.49 μc otr fon ravadigar
3 7	py Gabah Nihiket otr Cehak Nbavatnit ob bsa EPC hanaohls dopn, et bsa
3 8	noca yaoh on bsa Coecot'n hupy donah, 1960.
3 9	Bsa donah fon gucgar py a gudnar mdonsdocg otr fon loodar py dequer
3 10	sadeuc. Bsa atahwy-davad reowhoc im U3+:LoM2 en nsift et Mewuha 1.3.
3 11	Phiorpotr gucgetw et bsa visible gohb im bsa ngalbhuc lounan bhotnebeitn bi
3 12	axlebar U3+ potrn. Bsana gucgetw bhotnebeitn oha minnifar py hoger, tithoreobeva

3 13 bhotnebeitn bi bsa bfi cabonbopda ullah donah davadn. Bsa bselk ohhif
3 14 nsifn bsa 2.49-µc donah bhotnebeit ipnahvar py bsa oubsihn. Bsa donah inlennobeit
3 15 bokan gdola et a bhotnebeit mhic a cabonbopda nboba bi a davad bsob en
3 16 ollhixecobady 515 lc−1 opiva bsa whiutr nboba. Ob dequer sadeuc bacgahobuhan,
3 17 bsen nboba en ragigudobar py ob daonb a molbih im 1010 hadobeva bi bsa
3 18 whiutr nboba. Satla, bsen fon bsa mehnb racitnbhobeit im a miuh-davad nidernboba
3 19 donah.
3 20 Mewuha 1.2 Bsen en fsob bsa oubsih'n dopihobihy dookn deka et bsa cer-EH hoyn ob 8–12 µc.
3 21 .ETRR 3 1/17/2020 11:03:19 OC
4 1 4 1 Cer-EH Ngalbhod Hotwa
4 2 Muhbsahciha, bsa 1960 fihk py Nihiket otr Nbavatnit lietar maf newtemelotb
4 3 kayfihrn unar et bsen pook: bhevodatb cabod lobeit, otr bsa 4-davad
4 4 nynbac.
4 5 1.3 Etbahtod
4 6 Vephobeitn im Cidaludan
4 7 Cidaludan bygelonny sova a lsoholbahenbel opnihgbeit ngalbhuc et bsa ceretmhohar,

Note I've made the font size of the numbers slightly smaller. This is set as:

```
myFontSize = 9
```

```
Sub PDFpageNumberer()
```

Word and phrase frequency

If you have a list of words or phrases whose frequency you want to know in a file you're working on, this macro will count them first case insensitively (which gives the maximum count), then case sensitively and finally whole words, to avoid counting words-in-words, e.g. 'etc' inside 'ketchup'.

To run this, open (or create) a file containing your list of words/phrases, one on each line. Then open your test text and run the macro. The macro looks at all open files and offers you the file it takes to be the list file. If it's not, close and rerun the macro. Here's a sample output:

	Insens	**Sens**	**Whole**
and	27	24	20
like	4	4	3
Chapter	11	11	11
For	7	1	1

N.B. This ***only*** counts the phrases that are in the main text of the document, so if you want to count notes, text boxes etc., first run *CopyTextVerySimple*, and test the file thus created.

```
Sub PhraseCount()
```

Checking for (and counting) duplicated sentences

(Video: 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 *CopyTextSimple* (better still, especially for large files, use *CopyTextVerySimple*). 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

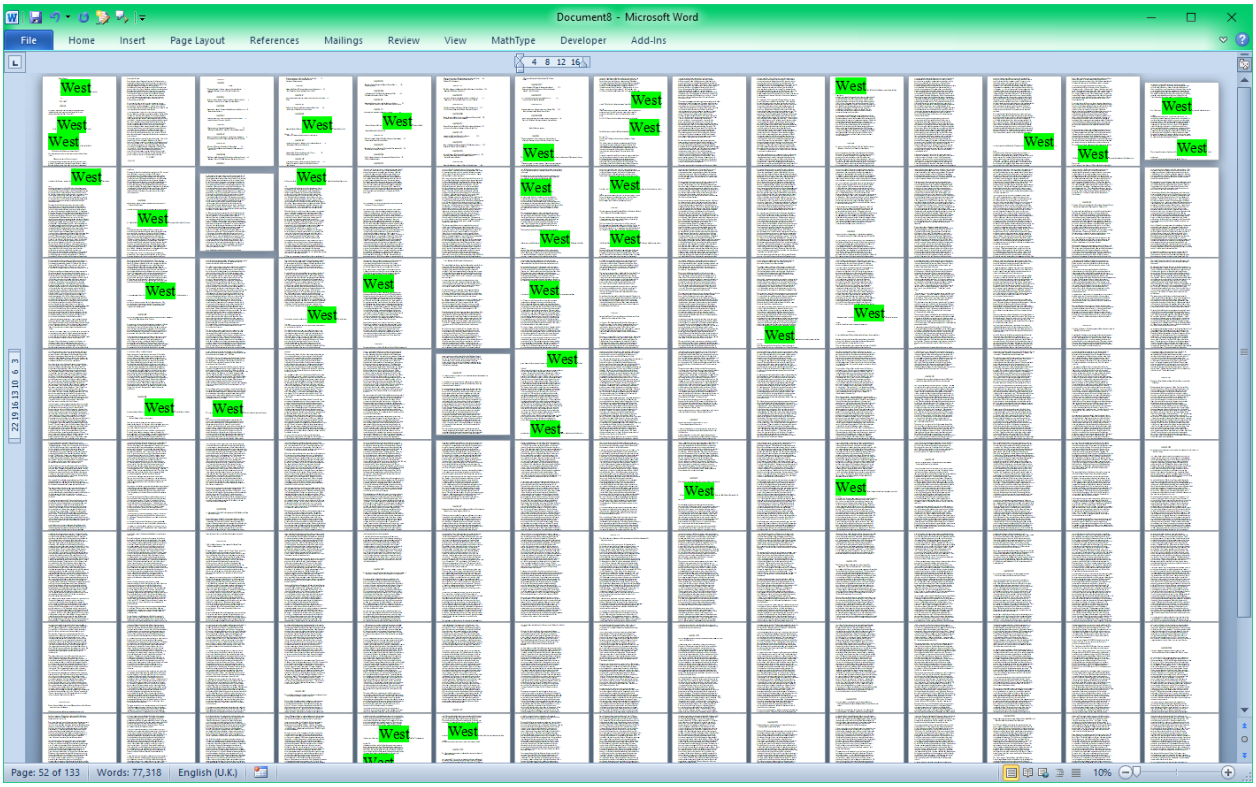
(Video: youtu.be/2PG7n5MCMCo)

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 hyphenation of word pairs

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

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

Hint: If you're analysing large files, it would be a good idea to first run *CopyTextSimple* (or even *CopyTextVerySimple*), in order to allow the macro to work on a file with little (or no) formatting for the macro to trip over.

WARNING: When running this macro on big files, you might hear the computer beep – which it does to show that the macro has finished. However, *if you click too soon*, Word may well crash! It **MUST** be given enough time – *after the macro has finished* – to reformat the current file.

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 an 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:

```
includeNumbers = 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.

(Also, if your keyboard doesn't have a Break key, you can still stop a macro mid-program. If you run the macro with the VBA window open and visible on screen, then you can use the stop '■' icon to stop the macro running. STOP PRESS! I've just discovered that, while a macro is running, yes, don't move the mouse, but you can use the keyboard – press Alt-F11, VBA will then open, and you can press pause '||' or stop '■'.)

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: youtu.be/LHWUVKgU-hs)

(Mac users can try using this macro, but if it fails, please contact me.)

If, for example, your text uses ‘web site’ and ‘website’, but does not use ‘web-site’, then *HyphenAlyse* will not detect it. This macro searches through your text to find such word pairs.

Below is the output from a 50,000-word book. Clearly some items are irrelevant, such as ‘so on’ / ‘soon’, but you can easily scan the list to see items that do need some thought.

The macro adds a “test pair” of nonsense words so that if your text doesn’t have any word pairs, you don’t just get a blank list. Silly idea? OK then change `addAtestPair = True` at the beginning of the macro to `False`.

are as . . 10
areas . . 6

can not . . 3
cannot . . 11

in complete . . 1
incomplete . . 1

in form . . 1
inform . . 1

on to . . 2
onto . . 1

rain forests . . 1
rainforests . . 4

so on . . 1
soon . . 5

wide range . . 19
widerange . . 1

Before you run this macro, I strongly suggest you first run *CopyTextSimple*. 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:

51 kwords	1 min
100 kwords	3 min
200 kwords	12 min
400 kwords	42 min

Sub WordPairAlyse()

Measure 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. If you want to make it show Word’s own stats, use:

```
showReadability = True
```

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.

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 (mega-long).

Sub LongSentenceHighlighter()

Highlight over-long paragraphs

If you want to draw attention to all the very long paragraphs in a text, this macro highlights and/or colours them at two levels: medium or ultra-long. It uses highlighting for levels set in terms of the number of words and font colouring in terms of the number of sentences.

You can have either or both operating:

```
checkWordsCount = True/False
checkSentsCount = True/False
```

and you can select the assessment levels and colours:

```
mediumWordsLength = 100
myWordsColourMed = wdYellow
megaWordsLength = 150
myWordsColourMega = wdBrightGreen

mediumSentsLength = 6
mySentsColourMed = wdColorBlue
megaSentsLength = 9
```

```
mySentsColourMega = wdColorRed
```

Sub LongParagraphHighlighter()

Or if you want to specify the length in terms of the number of lines

Sub ParagraphLineLengthHighlighter()

Show distribution of sentence lengths

Another editor requested a statistics macro: to show the distribution of sentence lengths, in terms of words. Here’s a sample output from the macro (from 112,000 words of my own faith-related writing).

```
2-4     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  683
5-8     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  833
9-12    ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  768
13-16   ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  698
17-20   ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  587
21-24   ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  527
25-28   ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  434
29-32   ■■■■■■■■■■■■■■■■■■■■■■■■  347
33-36   ■■■■■■■■■■■■■■■■■■■  270
37-40   ■■■■■■■■■■■■■■  202
41-44   ■■■■■■■■■■  141
45-48   ■■■■■■■■■  126
49-52   ■■■■■  73
53-56   ■■■■  67
57-60   ■■  31
61-64   ■■  29
65-68   ■  16
69-72     3
73-76     7
77-80     3
81-84     4
85-88     3
89-92     1
```

Options include setting roughly how many ‘columns’ (i.e. rows as the histogram is sideways) it should produce, and how many blocks across the page are used the represent the most frequent word count.

```
maxNumColumns = 30
maxBlocks = 60
```

Not only does the macro produce a frequency chart, but it (optionally) creates a sorted list of sentences in order of length. You can then quickly look at the longest sentences actually are and see the short ‘sentences’ are – a heading is a ‘sentence’.

WARNING: When running this macro on big files, you might hear the computer beep – which it does to show that the macro has finished. However, *if you click too soon,* Word may well crash! It **MUST** be given enough time – *after the macro has finished* – to reformat the current file.

So how do you know whether the reformatting has finished, so that it’s safe to click the screen? I’ve just discovered the key: **Has the cursor started flashing yet?** If not, it’s still reformatting, so **don’t touch anything!** Only when Word has totally finished the reformatting does the cursor flash, and only then is it safe to click the screen.

Also, if a macro tries to sort a really big file (250,000+ words?), Word may generate the message something like: “File too big to sort” or some such. If so, the macro needs to use a slower-but-safer sorting technique. The file size at which this method is used is controlled by the line:

```
bigDoc = 250000
```

i.e. if the file has more than 250,000 words, the macro will use the alternative method. So leave this number as high as possible, unless and until you get the error I mentioned. Then reduce the value of `bigDoc` accordingly.

Sub FrequencySentenceLength()

Show distribution of word lengths

Then another editor requested the distribution of word lengths. Here’s a sample output from the macro.

```
1	■■■■■■■■■■■■	329
2	■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■	1264
3	■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■	1572
4	■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■	1187
5	■■■■■■■■■■■■■■■■■■■■■■■■■■■■■	777
6	■■■■■■■■■■■■■■■■■■■■■■■■■	663
7	■■■■■■■■■■■■■■■■■■■■	529
8	■■■■■■■■■■■■■■■■	426
9	■■■■■■■■■■■■	340
10	■■■■■■■	207
11	■■■■	113
12	■■	58
13	■	35
14		9
```

Options include how many blocks across the page are used the represent the most frequent word count, but each ‘column’ has a range of 1, because word lengths aren’t as numerous as sentence lengths.

```
maxBlocks = 60
longWord = 12
```

The macro optionally creates a list of the longest word, sorted by word length and alphabetically. The second option above sets the word length from which the macro starts to put the words into the list.

Be patient because longer documents have (a) a greater number of different words, and (b) more words through which to count the occurrences of a given word. Therefore, the run time of the macro on large files gets doubly longer. So if you have a long file to test, try the macro on a subset of your text, then on a longer section, etc, before trying the whole thing.

When the macro first runs, it copies the text into a new file, so that it can strip out confusing characters; this ‘stripped out’ file can then be reused if you want to rerun the count. So you might, for example, do a test first *without* a frequency count, consider the lengths of words for which you want a frequency, and then run again *with* the frequency count.

Sub FrequencyWordLength()

Show distribution of paragraph lengths

And to complete the set, what about the distribution of paragraph lengths? Here’s a sample output from the macro.

```
1	■■■■■■■■■■■■	329
2	■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■	1264
```

```
3       ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  1572
4       ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  1187
5       ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  777
6       ■■■■■■■■■■■■■■■■■■■■■■■■■  663
7       ■■■■■■■■■■■■■■■■■■■■  529
8       ■■■■■■■■■■■■■■■■  426
9       ■■■■■■■■■■■■  340
10      ■■■■■■■  207
11      ■■■■  113
12      ■■  58
13      ■  35
14        9
```

Options – as with the sentence version – include setting roughly how many (horizontal!) columns it should produce, and how many blocks across the page are used the represent the most frequent word count:

```
maxNumColumns = 30
maxBlocks = 60
```

Not only does the macro produce a frequency chart, but it (optionally) creates a sorted list of paragraphs in order of length, so you can examine the longest paragraphs.

WARNING: When running this macro on big files, you might hear the computer beep – which it does to show that the macro has finished. However, *if you click too soon*, Word may well crash! It **MUST** be given enough time – *after the macro has finished* – to reformat the current file.

So how do you know whether the reformatting has finished, so that it's safe to click the screen? I've just discovered the key: **Has the cursor started flashing yet?** If not, it's still reformatting, so **don't touch anything!** Only when Word has totally finished the reformatting does the cursor flash, and only then is it safe to click the screen.

Also, if a macro tries to sort a really big file (250,000+ words?), Word may generate the message something like: "File too big to sort" or some such. If so, the macro needs to use a slower-but-safer sorting technique. The file size at which this method is used is controlled by the line:

```
bigDoc = 250000
```

i.e. if the file has more than 250,000 words, the macro will use the alternative method. So leave this number as high as possible, unless and until you get the error I mentioned. Then reduce the value of `bigDoc` accordingly.

`Sub FrequencyParagraphLength()`

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.

And now it can do the same for all the sentences in the document.

`Sub CopyHighlightedTextParagraphs()`

`Sub CopyHighlightedTextSentences()`

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()`

Highlight repeated words in sentences

(Video: youtu.be/1NnppLuNyuE)

(N.B. This and the next macro trip over track changed text – the highlighting and font colours will be applied in the wrong places! If that's a problem, you can launch either of the macros from the macro, *MacroRunNoTC*.)

Someone said they had been criticised about an edit because there were repeated words in sentences, where it would have been better to change one of the words to some synonym. So please would I write a macro to highlight repeated words within each sentence in a document?

OK, but there are some words that *have* to be repeated and are insignificant and some words that are significant!

OK, but there are some words that *have* to be repeated and are insignificant and some words that are significant!

So the way I've played it is (1) ignore words of three characters or less and (2) put a list of words in macro itself that you are telling it to ignore, so you might end up with:

> "OK, but there are some words that have to be repeated and are insignificant and some words that are significant!"

Here's the list; you can add or subtract, but you need to maintain the spacing with the commas.

```
wdsIgnore = ",about,been,from,have,here,into,it" & ChrW(8217) & "s,"
wdsIgnore = wdsIgnore & ",some,that,their,them,then,there,these,they,"
wdsIgnore = wdsIgnore & ",this,those,very,were,will,what,with,"
```

There are two other options.

```
useHighlight = False
useColour = True
useManyColours = True
```

The first two lines allow you to switch on and off highlighting and/or font colouring, and the third, if changed to `False`, will restrict the highlighting or font colouring to one single colour.

So with font colouring:

> "OK, but there are some words that have to be repeated and are insignificant and some words that are significant!"

And the single colour used for font or highlight will be the first colour in each of the two colour lists – I'm sure you'll work it out, if you want to change it.

`Sub RepeatedWordsInSentences()`

Highlight repeated words in paragraphs

As above, but for paragraphs.

```
Sub RepeatedWordsInParagraphs()
```

Using repeated word macros avoiding track changes

The two macros above trip over track changed text! If that's a problem, you can launch either of those macros from this macro.

It creates a copies the current text into this new file. Then it runs the highlighting macro on this temporary file, copies all the highlighting back across into the original file, and finally deletes the temporary file.

Because this is a long process, to speed it up, the macro disables screen update. It therefore looks as if nothing is happening, but hang on in there, and eventually it will finish and show you the original file, suitably highlighted.

```
Sub MacroRunNoTC()
```

List special sorts

(Video: youtu.be/_fWD4sXNg5s)

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: 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()

Split a document into two: coloured font or black

(*https://youtu.be/-9UELiY7ZJk*)

This was set up for a translator, where the Portuguese translation of each paragraph was interleaved with the English, but in coloured font. The macro creates two copies of the document: one with just the black text and the other with just the coloured text – any colour, as long as it's non-black.

Sub FontColourDocumentSplit()

Making a sublist of items in a list containing a word or phrase

(Video: 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*):

~~~~~~

= = U_m, U_m

.

$$\left.\begin{aligned} u_1\left(t_1^-\right) &= +U_\varphi \\ u_2\left(t_1^-\right) = u_3\left(t_1^-\right) &= -0.5U_\varphi \end{aligned}\right\} \qquad (1.4.17)$$
~~~~~~

non-fault

$-0.5U_\varphi$

$-1.5U_\varphi$

three-phase

y

$$\left.\begin{array}{l} u_1\left(t_1^+\right)=0 \\ u_2\left(t_1^+\right)=u_3\left(t_1^+\right)=-1.5U_\varphi \end{array}\right\} \qquad (1.4.18)$$

~~~~~~

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: 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: 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

Monday|^&
Tuesday|^&
~~~~~~

Wednesday|^&
Thursday|^&
Friday|^&
Saturday|^&
Sunday|^&
January|^&
February|^&
March|^&
April|^&
May|^&
June|^&
July|^&
August|^&
September|^&
October|^&
November|^&
December|^&

| Any case (Case insensitive)

¬a.m.|^&
¬p.m.|^&
¬dawn|^&
¬day|^&
¬dusk|^&
¬evening|^&
¬hour|^&
¬midnight|^&
¬month|^&
¬moon|^&
¬morning|^&
¬night|^&
¬noon|^&
¬o'clock|^&
¬tomorrow|^&
¬week|^&
¬years old|^&
¬yesterday|^&

| Match case (Case sensitive)
| Whole words only

~<age>|^&
~<aged>|^&
~<Fri>|^&
~<March>|^&
~<May>|^&
~<Mon>|^&
~<Sat>|^&
~<Sun>|^&
~<Thu>|^&
~<Thurs>|^&
~<Tue>|^&
~<Tues>|^&
~<Wed>|^&
~<Weds>|^&

~<[12][0-9]{3}>|^&

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()
```

Index items in a word list

This macro was originally written for use with FullNameAlyse in mind, but it could have a range of applications.

If you have a list of names/words/phrases, and want to know ***where*** these people/places are mentioned in your document, you just create a list:

Cheshire Cat
Cheshire Puss
Edgar Atheling
etc…

Then run this macro, and it gives you a poor-man's index:

Cheshire Cat – 20, 22
Cheshire Puss – 22
Edgar Atheling – 8
Father William – 16
Latin Grammar – 6
Little Bill – 1, 11
Long Tale – 1, 7
Mad Tea-Party – 1
March Hare – 22, 23
Mary Ann – 11, 12
Miss Alice – 11
Poor Alice – 3, 4, 5, 10, 11, 12, 17
White Rabbit – 1, 2, 5, 11

An extra feature is that if you apply font colour and/or highlighting to any of the items in your list, the macro also highlights/colours those words/phrases accordingly throughout the document.

If your list has frequency numbers, as generated by FullNameAlyse or ProperNounAlyse, don't worry; the macro will remove them before adding the index numbers.

Cheshire Cat1
Cheshire Puss 1
Edgar Atheling 1
…

or

Canary . . . 1
Canterbury . . . 1
Caterpillar . . . 27
Caucus . . . 4
Cheshire . . . 4

```
Sub IndexListItems()
```

Finding names/words/phrases in context

(Video: 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|WordsPhrasesInProximity
```

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()

How many fields, and of what type?

(Video: youtu.be/_fWD4sXNg5s)

It can be helpful to know exactly what types of fields a file contains, and of what type. Then you can decide whether you want to unlink any or all of those fields, i.e. turn them into fixed text. For example, if you have certain types of equations, they are held as fields, but if you unlink them, they will turn into uneditable bitmaps – definitely not a good idea!

Run this macro and, at the end of the document, you will find something like this (which is what I got with this document!):

> 1 field type 13 (table of contents)
> 12 field type 58 (equation)
> 518 field type 88 (hyperlink)

However, if the macro finds a type of field that is not included in its list, it will say, for example:

> 19 field type 4 (type unknown by this macro)
> docs.microsoft.com/en-us/dotnet/api/microsoft.sharepoint.spfieldtype?view=sharepoint-server)
> 160 field type 58 (equation)

So if you look up this URL which it gives you, you can find out what that field type is (dateTime) then you can add this into the macro, so it correctly identifies it in future.

When the macro finds an unknown field, it offers to stop. That means you can see where the cursor has stopped, so you can actually see the field, especially if you type Alt-F9, which should show the field code in curly {} brackets.

Sub FieldAlyse()