A BAE Book Example

BAE Faculty

2018-12-18

Contents

Chapter 1

Introduction

This is a *sample* book written in **Markdown**. This is a great package written by Yihui Xie. I am using his template, and added some additional info I thought was useful along the way! This is really a very succint tutorial, so take it as an introduction to your discovery of Reproducible research!!

The **bookdown** package can be installed from CRAN or Github:

```
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")
```

Remember each Rmd file contains one and only one chapter, and a chapter is defined by the first-level heading `#`.

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.name/tinytex/.

Chapter 2

Introduction: Text features

The markdown syntax is very simple and this is one of the reasons why it is so attractive. You can find all the syntax with this R Markdown reference document, but I thought I would go through them as I have learned several tricks I could not easily find in the document.

2.1 Paragraph Breaks and Forced Line Breaks

To insert a break between paragraphs, include a single completely blank line.

To force a line break, put *two* blank spaces at the end of a line.

```
To insert a break between paragraphs, include a single completely blank line.

To force a line break, put _two_ blank spaces at the end of a line.
```

2.2 Headers

The character `#` at the beginning of a line means that the rest of the line is interpreted as a section header. The number of `#`s at the beginning of the line indicates the level of the section (1,2,3, etc.). For instance, `Components of R Markdown` above is preceded by a single `#`, because it is of level 1 but `Headers` at the start of this paragraph is preceded by `###` because it is of level 3. Up to six levels are understood by Markdown. Do not interrupt these headers by line-breaks. Make sure that in your .Rmd file, you leave a blank line before a header, otherwise, `pandoc` will not render it as a header.

2.3 Italics and bold

```
*italics* and _italics_
```

renders *italics* and *italics*

```
**bold** and __bold__
```

renders **bold** and **bold**

2.4 Supbscripts and superscripts

To write sub- and superscripts, like in NO$_3$$^-$ or PO$_4$$^{3-}$ write as

```
NO~3~^-^ or PO~4~^3-^
```

PO$_4$$^{3-}$ does not look as neat as $PO_4^{3-}$

```
PO~4~^3-^ does not look as neat as $PO_4^{3-}$
```

but looks more seamless in a normal text because the former does not appear as an equation while the other does.

2.5 Lists

```
* unordered list
* item 2
    + sub-item 1                #4 <spaces> before +
    + sub-item 2                #4 <spaces> before +
```

- unordered list
- item 2
  - sub-item 1
  - sub-item 2

```
1. ordered list
2. item 2
    + sub-item 1                #4 <spaces> before +
    + sub-item 2                #4 <spaces> before +
```

1. ordered list
2. item 2
   - sub-item 1
   - sub-item 2

In the ordered list, there is a subtlety unrevealed in the Rmardown documentation, which is that the numbering *always* increases, and that *only* the number value of the first item matters. So, one cannot have a list in a decreasing order (which is too bad because when one makes a list of his/her publications, it is nice to have a decreasing order…), and the only number that matters is the first one. So this code

```
5. ordered list
7. item 2
2. item 2
                                # blank line for list to take into effect
    b. sub-item 1               #4 <spaces> before b
    a. sub-item 2
```

renders this:

5. ordered list

6. item 2

7. item 2

    b. sub-item 1

    c. sub-item 2

2.6 Quotations

> R Markdown is, in particular, both "free as in beer" (you will never pay a dollar for software to use it) and "free as in speech" (the specification is completely open to all to inspect).

Is a quotation from Dr Shalizi which code starts with a `>`

```
> R Markdown is, in particular, both "free as in beer" etc.
```

2.7 Computer type

This is to differentiate regular text from `code text` so that both can be easily differentiated: `R` vs R.

```
This is to differentiate regular text from `code text` so that both can be easily differentiated: `R` v
```

An entire paragraph of code which is rendered as a "code box" in html (but not in pdf) starts with three "back-ticks", and end the same

2.8 Symbols and Special characters

The principal keys, like the alphabet, are understood univocally across platforms such as Windows, Mac OS, or Linux. However, there are special characters such °, ^{2} or µ that have different embedded codes across the different platforms. For example, if you and your co-worker work on the same document and one works using Windows and the other uses Mac, the actual symbol in the code text may not show the same one from one to the other plateform.

For example, if I write in an R markdown document "10 m^2" and I have added the ^{2} symbol by typing on a PC "Alt+0178", as this corresponds to the ascii code for ^{2} in Windows, the same document open on a Mac will render "10 m?", because it cannot interpret the embedded code for ^{2}…

Several consequences:

1. **NEVER** use special characters in variable names in an `R` code,
2. in an R markdown document, use the HTML Number or HTML name in the text

HTML Number and Names can be found on this ascii page or on this page for greek letter codes and some other symbols

So to type T°C, m^2, µm, or p you type like this

```
So to type T&deg;C, m&sup2;, &micro;m, or p&epsilon; you type like this
```

Do not forget to add the ";" !!!

Chapter 3

Editing equations in R Markdown

We do not write nearly as much math in our field than our colleagues in statistics or math. But we do enough to have a bit of a tutorial. I have borrowed this entire section from the excellent tutorial from Dr Shalizi again. R Markdown gives you the syntax to render complex mathematical formulas and derivations, and have them displayed very nicely. Like code, the math can either be inline or set off (displays).

3.1 Inline equations

Inline math is marked off witha pair of dollar signs (`$`), as $\pi r^2$ or $e^{i\pi}$.

```
Inline math is marked off witha pair of dollar
signs (`$`), as $\pi r^2$ or $e^{i\pi}$.
```

3.2 Set off equations

In the R markdown tutorial I have other instructions about equations that are really related to R markdown. I now prefer the way it is done in `bookdown` much better, particularly for the referencing.

This simple LaTeX syntax

```
\begin{equation}
H_2O + H_2O \rightleftharpoons OH^- + H_3O^+
\label{eq:autoprotolysis}
\end{equation}
```

yields

$$H_2O + H_2O \rightleftharpoons OH^- + H_3O^+ \tag{3.1}$$

Notice that the number (3.1) is automatically added as you put in the text `\@ref(eq:autoprotolysis)`.

Of course, it is possible to write a little more complicated equations. The limit is LaTeX. I have not explored all what there is but here is an example that was useful for me:

```
\begin{equation}
H_nA \rightleftharpoons H_{n-1}A^- + H^+ \hspace{0.6cm} K_{a1} = \frac{[H_{n-1}A^-][H^+]}{[H_nA]} \hspac
H_{n-1}A^- \rightleftharpoons H_{n-2}A^{2-} + H^+ \hspace{0.6cm} K_{a2} = \frac{[H_{n-2}A^{2-}][H^+]}{[
\\
\vdots \hspace{6cm} \vdots \hspace{6cm} \vdots\\
\\
HA^{(n-1)-} \rightleftharpoons A^{n-} + H^+ \hspace{0.6cm} K_{an} = \frac{[A^{n-}][H^+]}{[HA^{(n-1)-}]}

\label{eq:HnApoly}
\end{equation}
```

yields

$$H_nA \rightleftharpoons H_{n-1}A^- + H^+ \qquad K_{a1} = \frac{[H_{n-1}A^-][H^+]}{[H_nA]} \qquad [H_{n-1}A^-] = [H_nA]\frac{K_{a1}}{[H^+]} H_{n-1}A^- \rightleftharpoons H_{n-2}A^{2-} + H^+ \qquad K_{a2} = \frac{[H_{n-2}A}{[H_{n-}} \tag{3.2}$$

Again, you see that the numbering of the equation is automatic!!

Chapter 4

Code chunks

So this is one of the great beauties of the `R markdown` platform: You insert text, equations, and code chunks! Code chunks are used to run `R` code, `python` code, and others such as `C` and `fortran` codes! In `bookdown`, one also uses them to insert pictures, videos, and tables.

4.1 Inserting pictures

This code:

yields:

So the first thing in the code is its name. It then becomes possible to reference the figure using `Figure \@ref(fig:diagenesis-diffusion-directions)` to say: look at the cool stuff in Figure 4.1. Then, there are other settings for the code, which are detailed below. The figure caption is announced with the `fig.cap=`.

4.2 Inserting several pictures

It is also possible to insert several pictures lined up using this code:

which yields:

4.3 Inserting videos

Similarly, it is possible to insert videos from the web using this code:

which yields:

ATP synthase in action. Obtained with permission from HarvardX

4.4 Inserting R code

And obviously, a nice thing about `R markdown` is to be able to insert `R` code chunks like the one below to make some pretty complicated figures (the one here is not that much, and is not particularly well written…)

which yields:

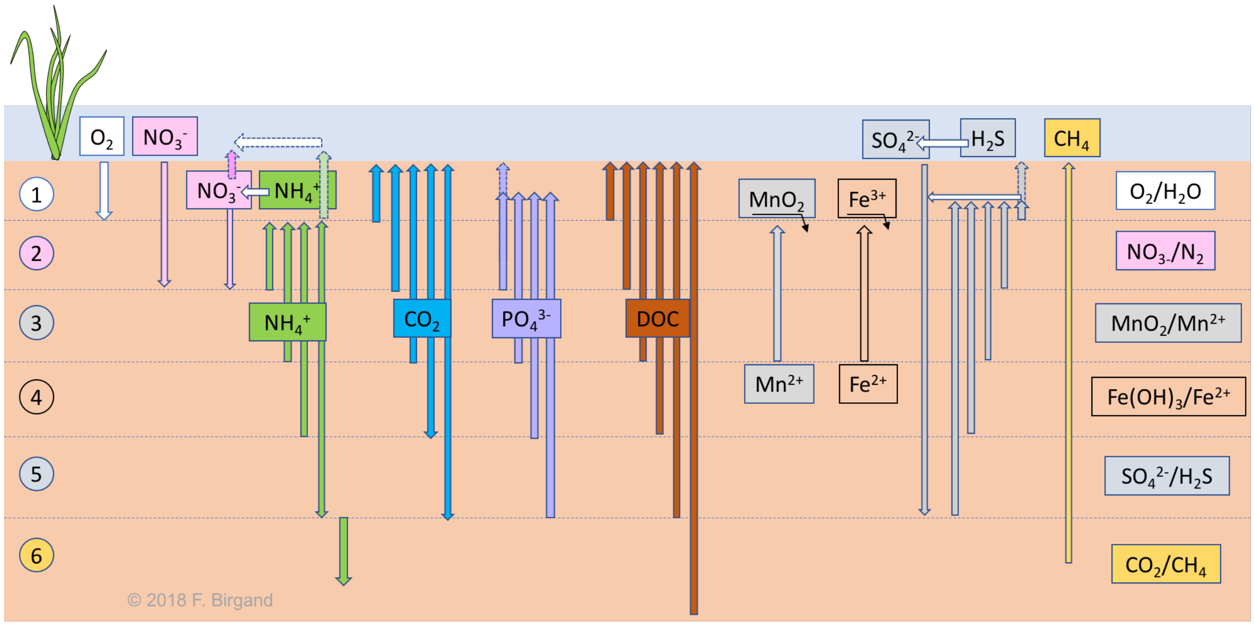


Figure 4.1: Diffusion fluxes of electron acceptors and all other soil diagenesis processes of a theoretical layered wetland soil

Figure 4.2: Small and large structures can be built from the addition of bricks, one at a time

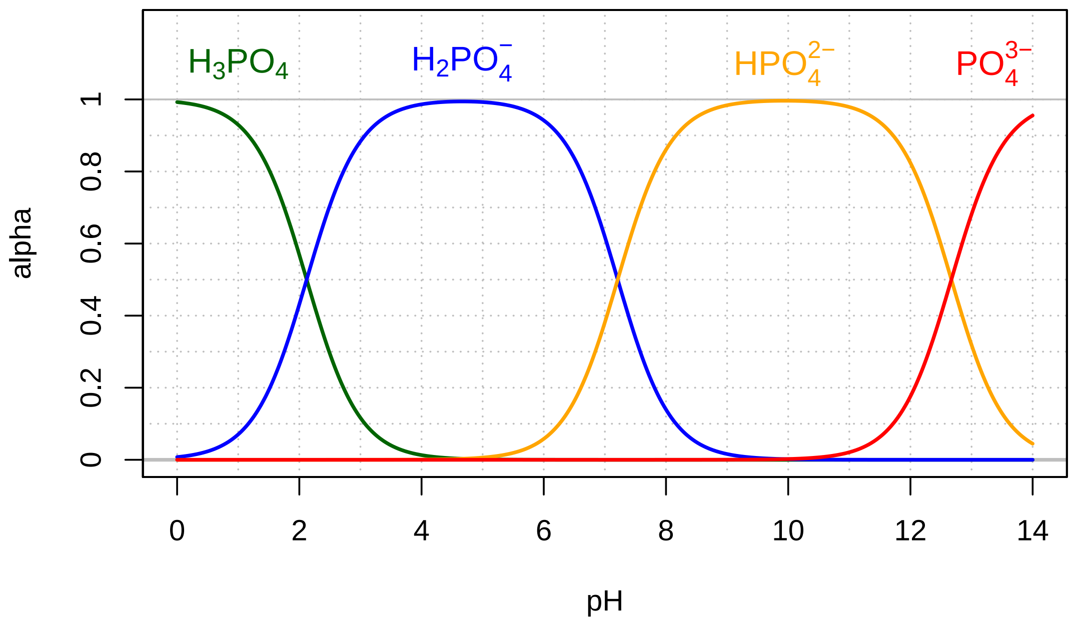


Figure 4.3: Molar fraction for the conjugate acid forms of the triprotic phosphoric acid in dilute solutions at 25°C

4.5 Inserting tables

To me, this is where Rmarkdown is the weakest for now. Tables are not that easy to handle... Here is a Chunk code that works with `pander`. I had to use this because otherwise, it would not render the equations well, but it does not have a caption, which is what I was trying to have anyway.

Equilibrium reactions	Log K
$H^+ + OH^- \quad H_2O$	-14.00
$H^+ + e^- \quad H_{2(g)}$	0.00
$H^+ + e^- + \frac{1}{4}O_{2(g)} \quad \frac{1}{2}H_2O$	20.78

There is another package which I like in many ways, but it is still imperfect: `KableExtra`. In the code below, it is possible to insert picture with text.

The beauty is that I was able to get the caption here and this is very useful. I suppose that for most applications, `KableExtra` is still the best thing outthere. Again, to reference the table, use `\@ref(tab:ElecAllocTab)` to say that table 4.5 is very messy!!

Examples of electron allocations on the C, N, S, and P atoms generating different inorganic and organic molecules relevant to environmental and ecological engineering

Nb of e^- stored on the atoms

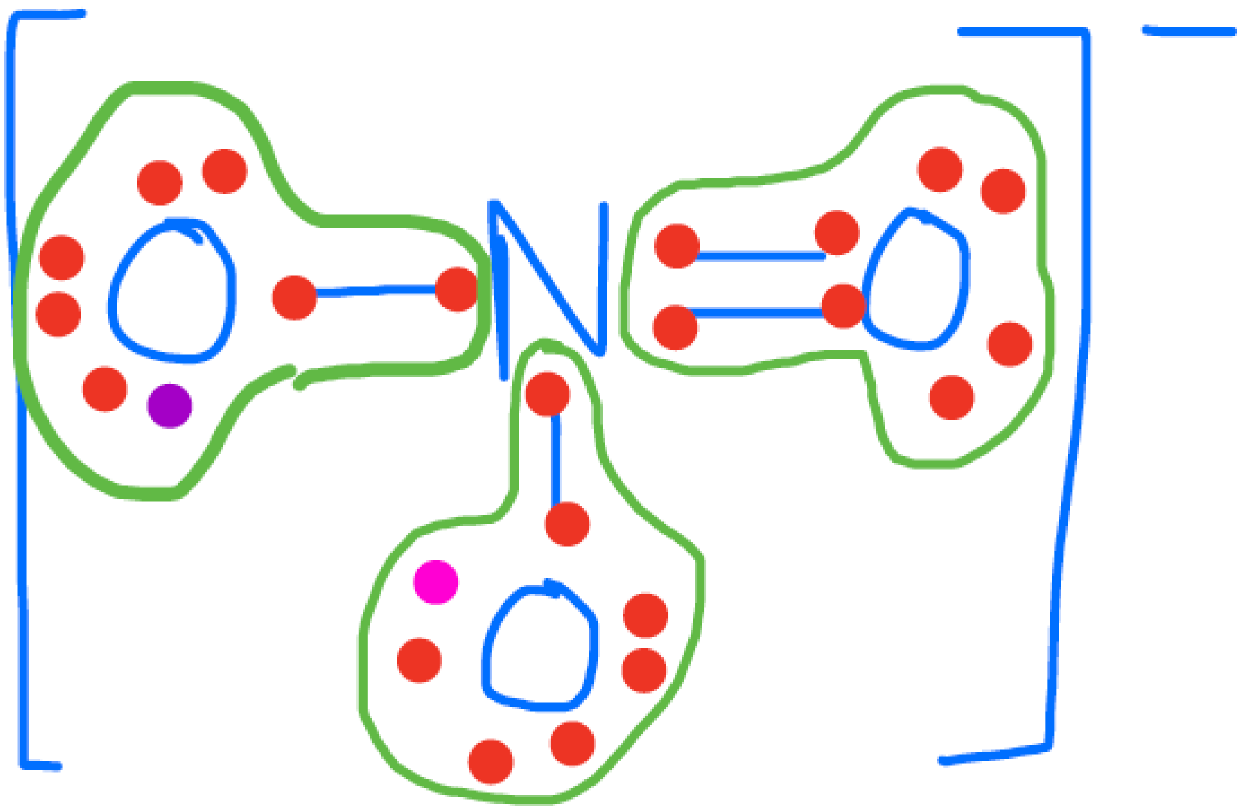
C

N

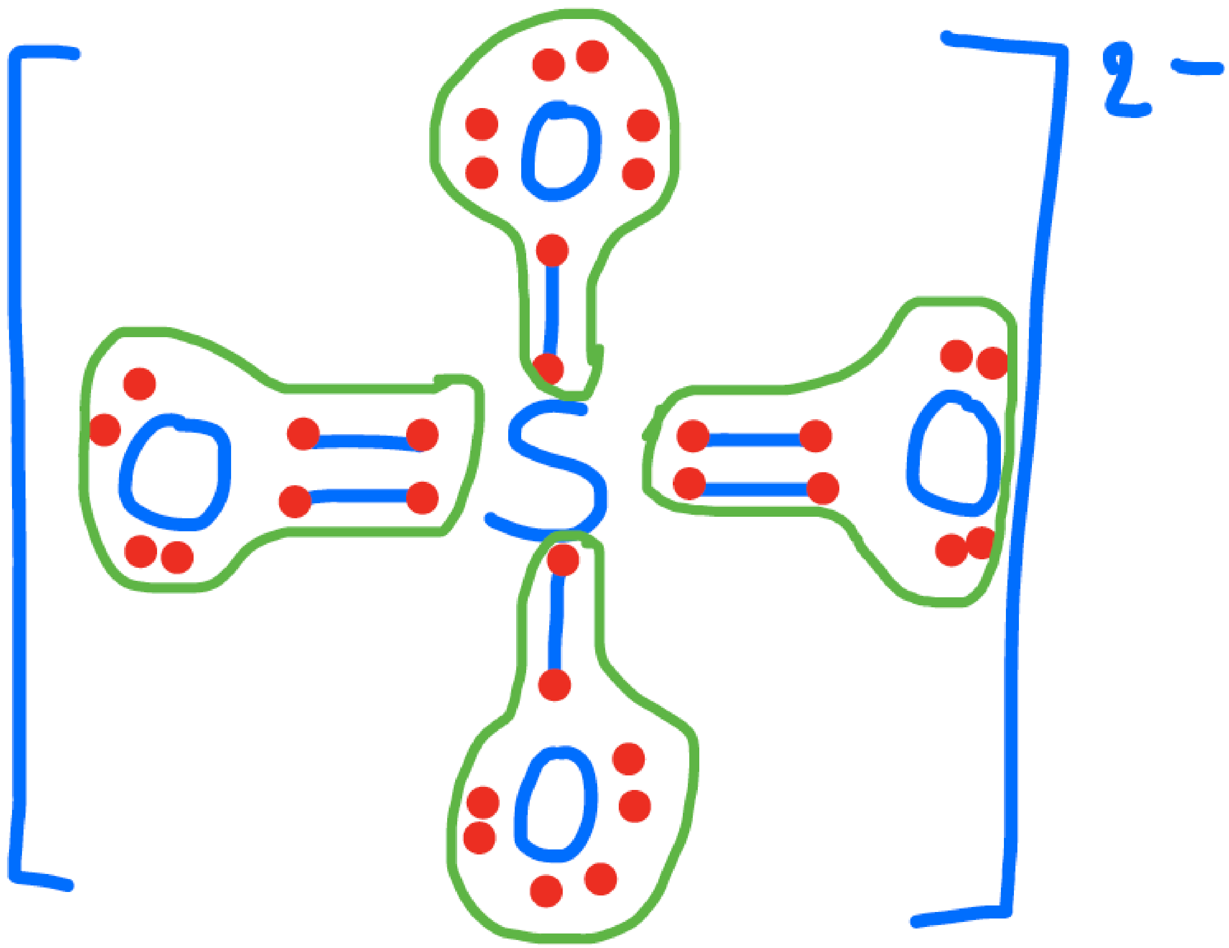
S

P

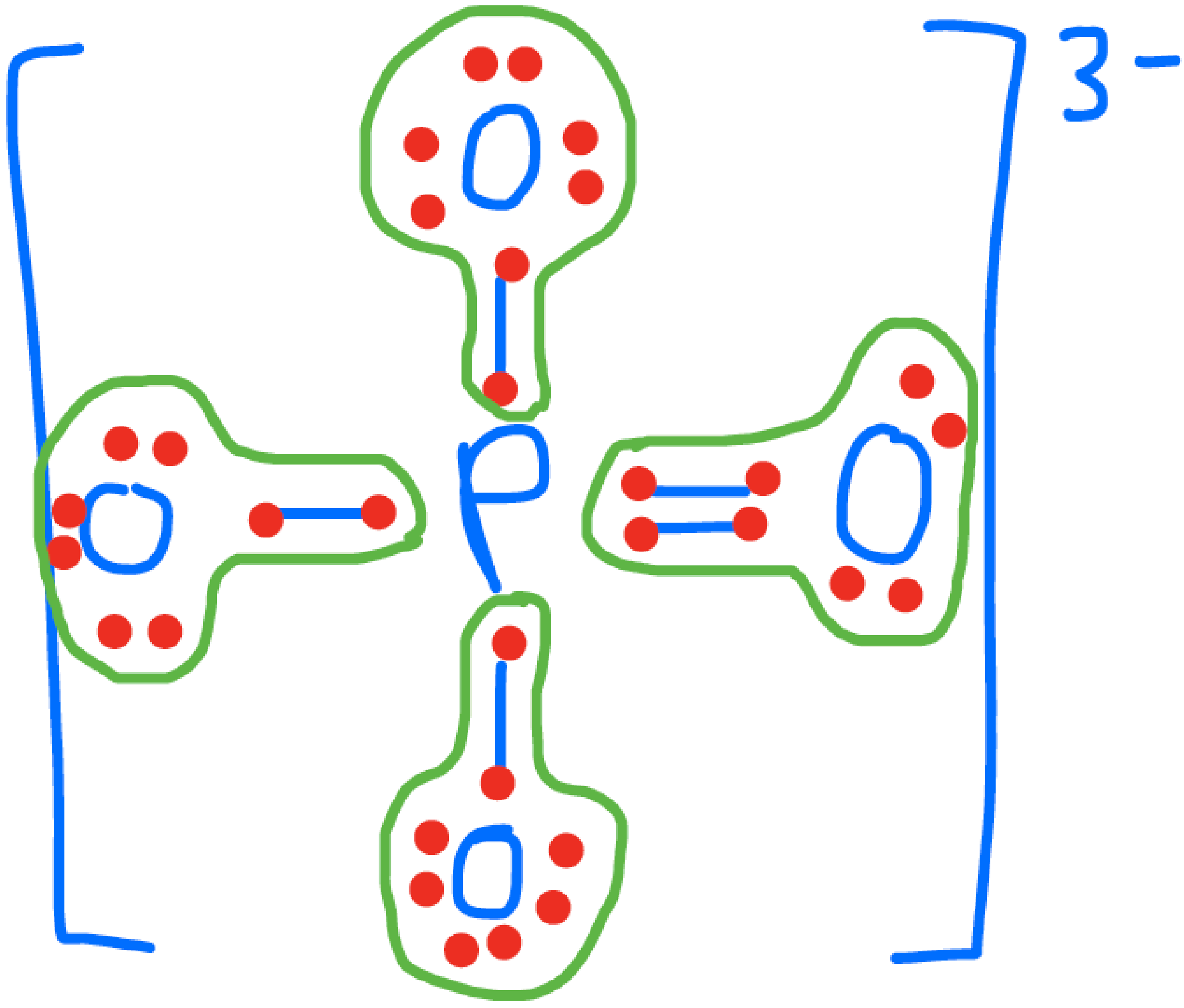
0



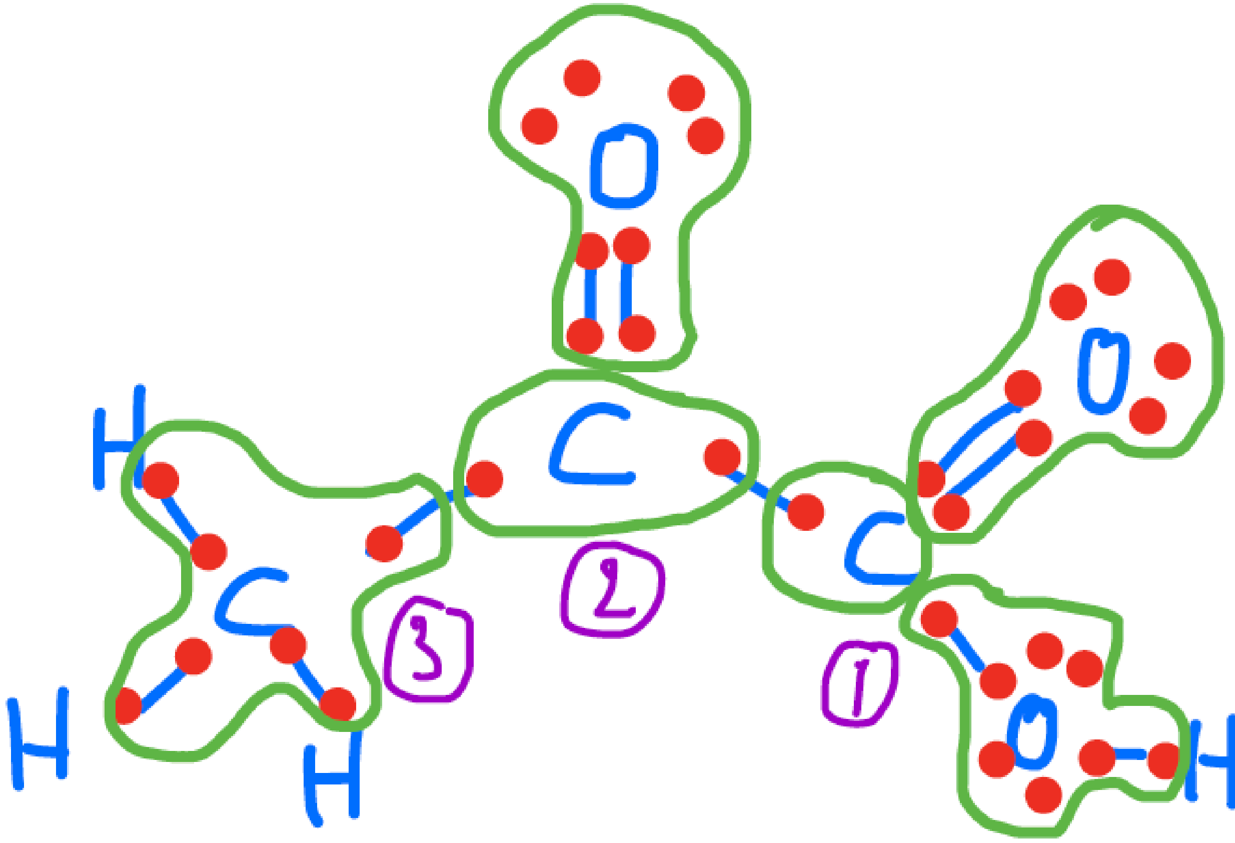
nitr***ate***



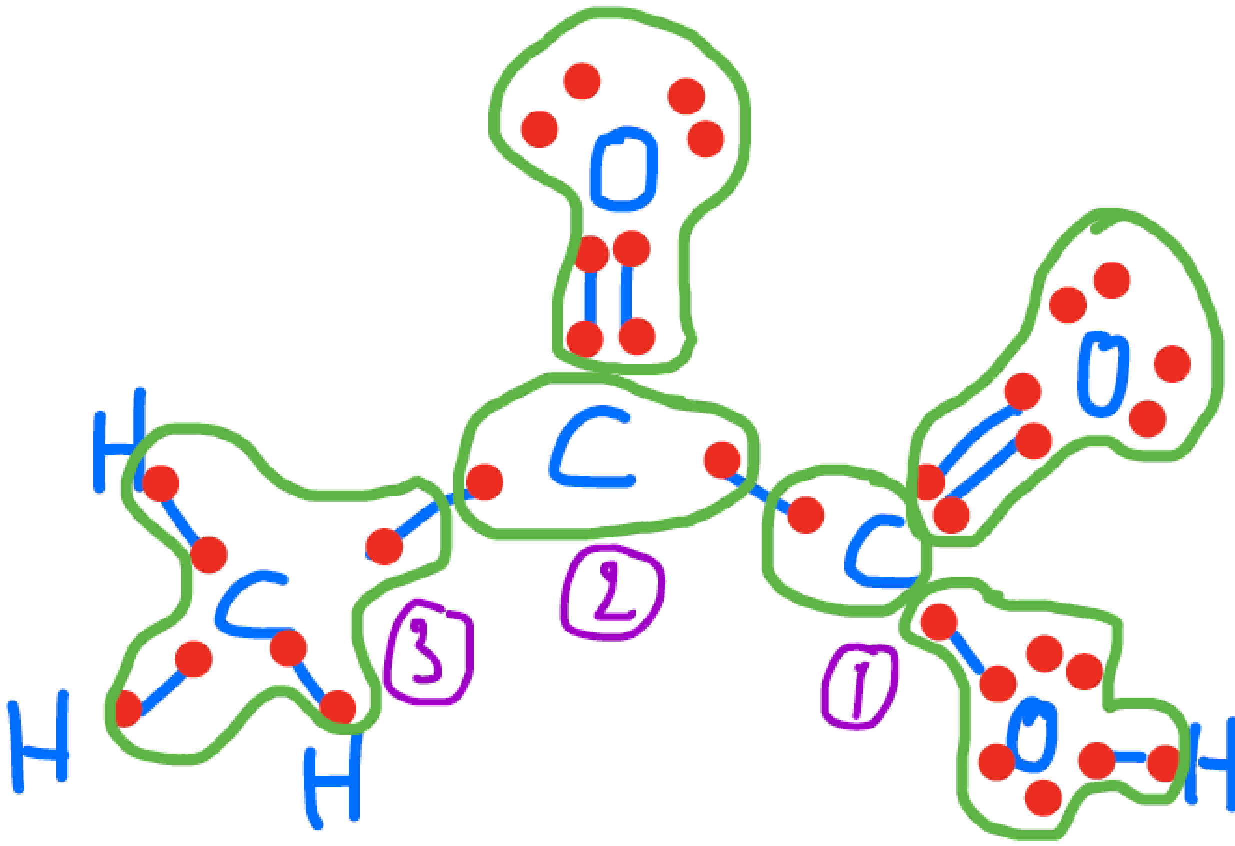
sulf***ate***



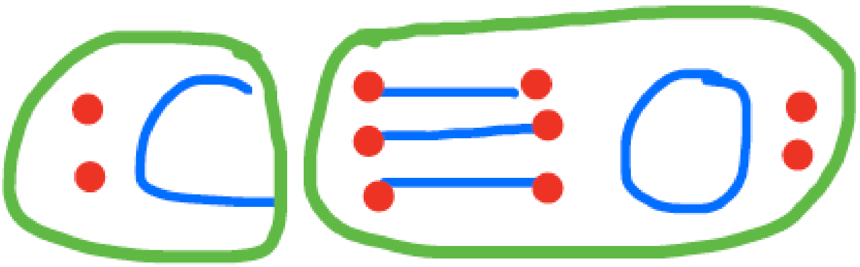
phosph***ate***



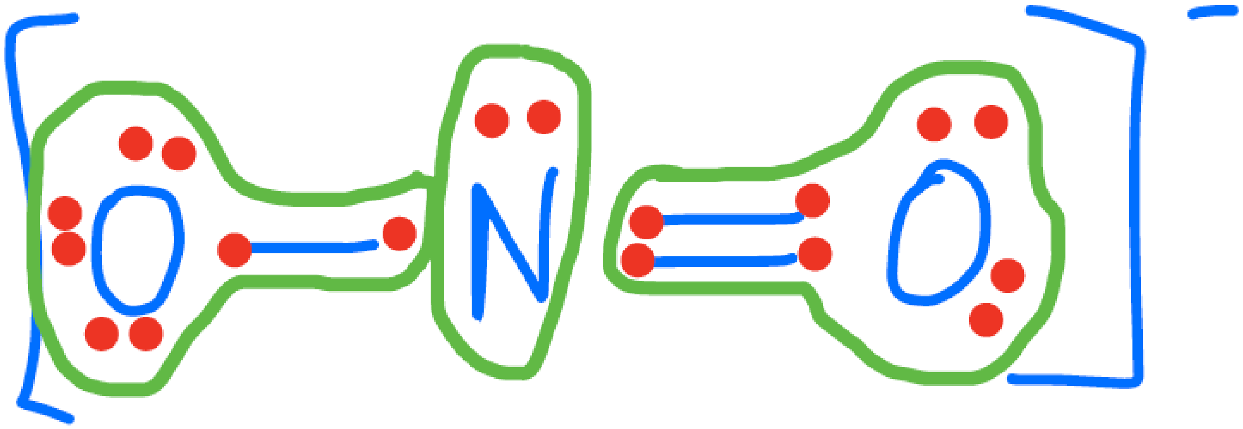
C#1 pyruvic acid



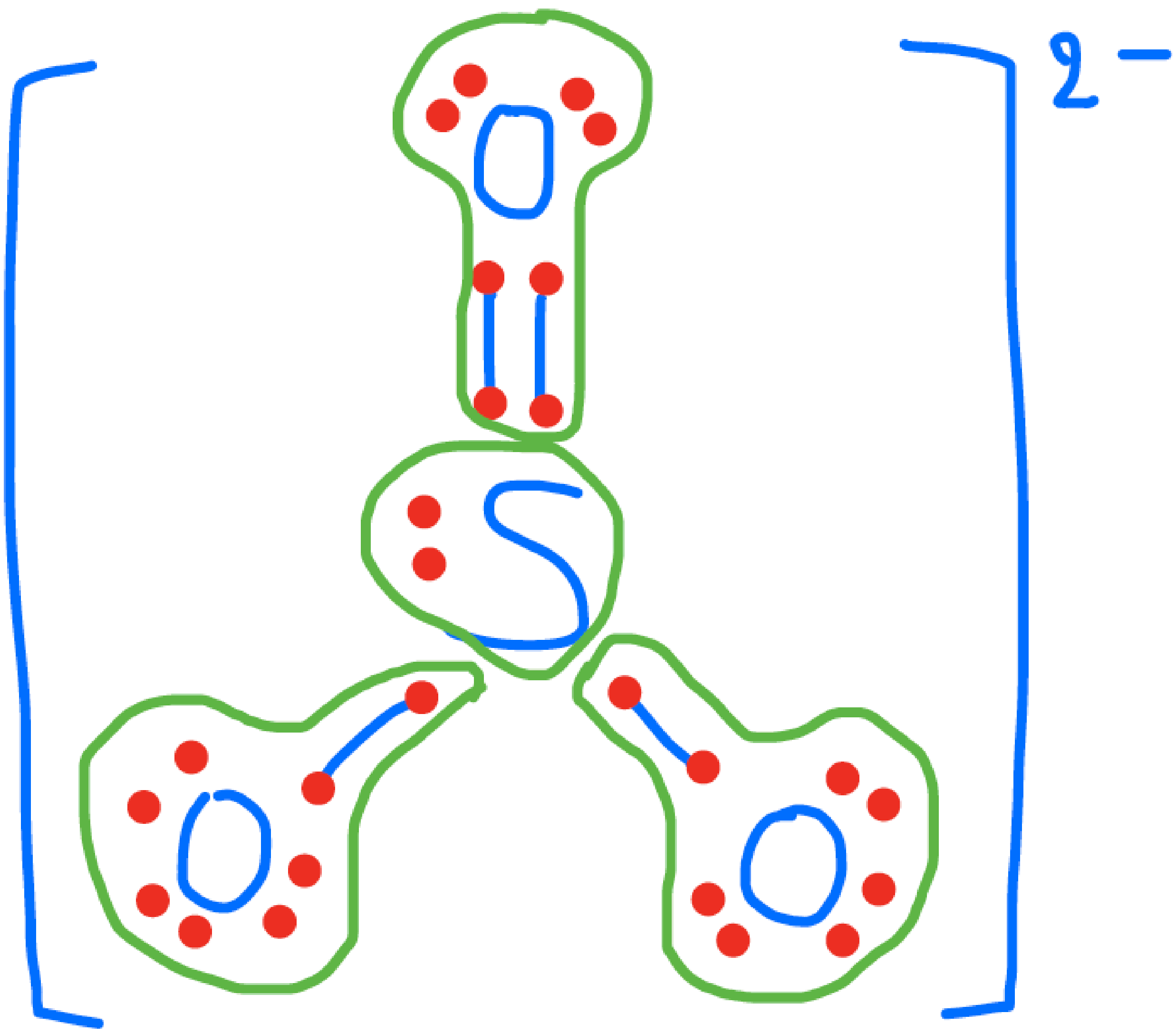
C#2 pyruvic acid



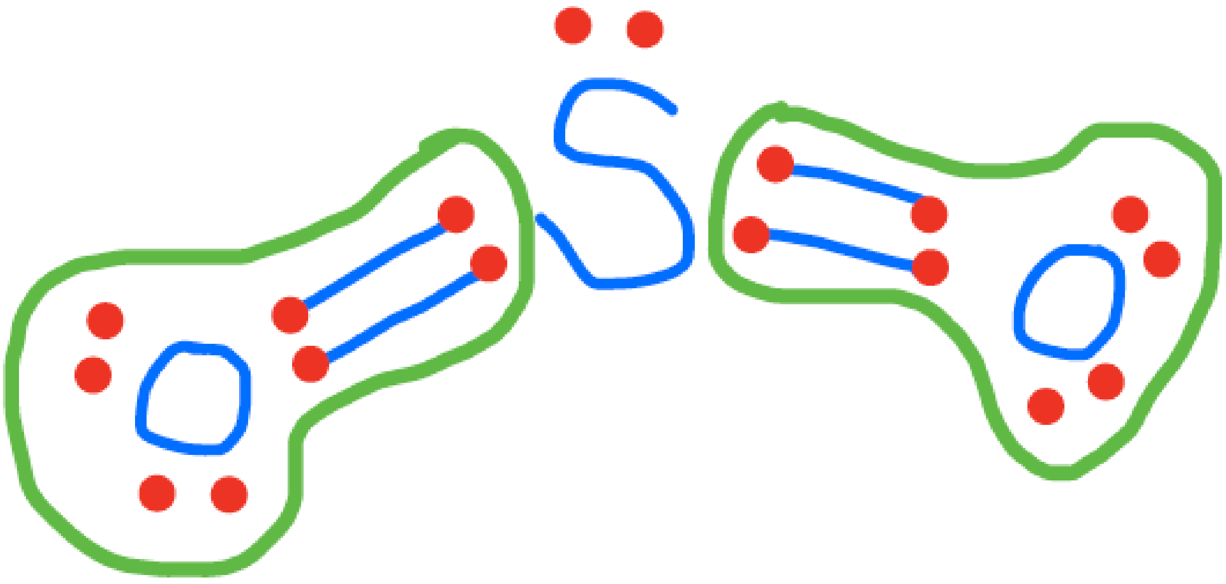
carbon monoxide



nitr***ite***

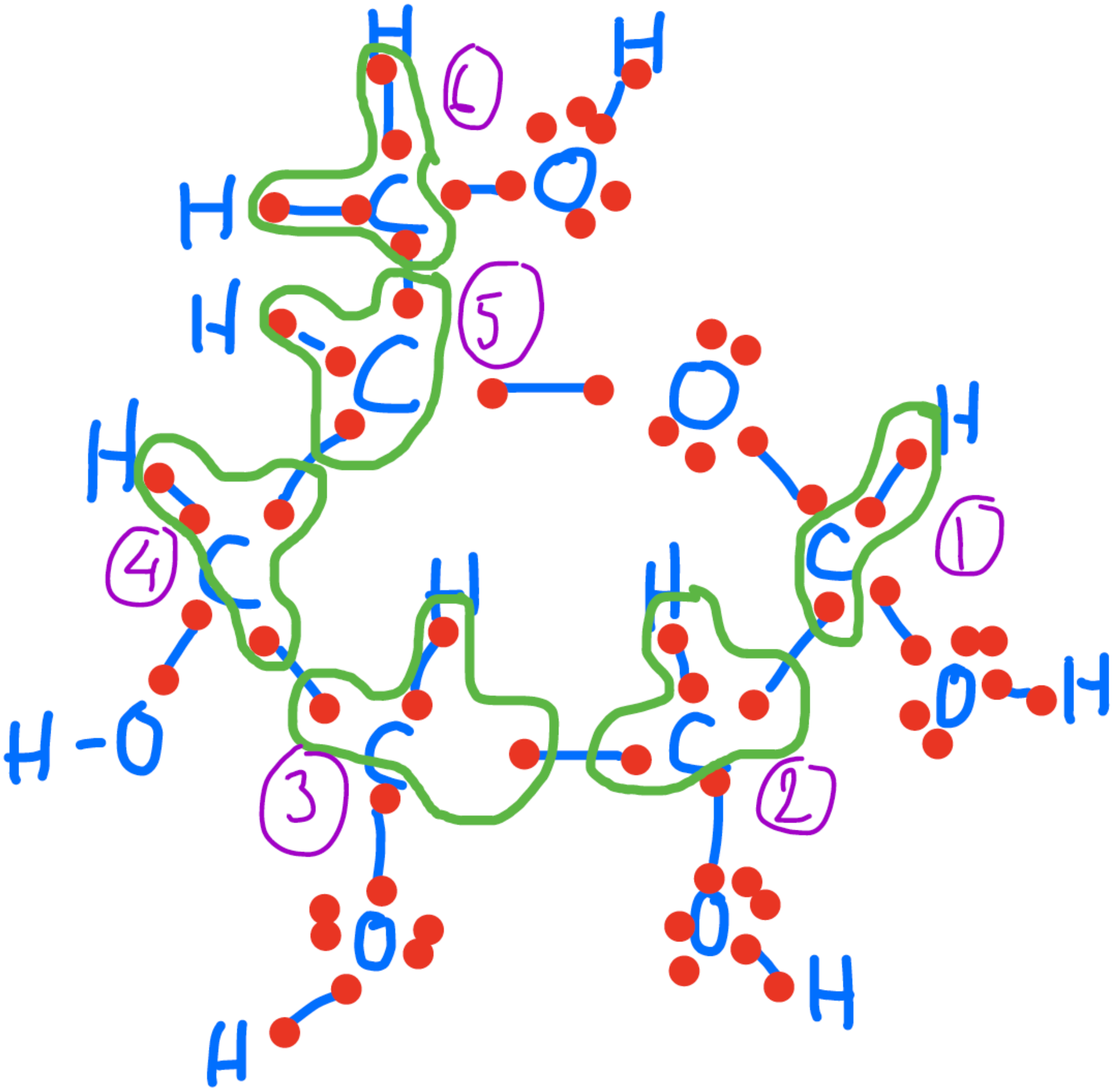


sulf***ite***

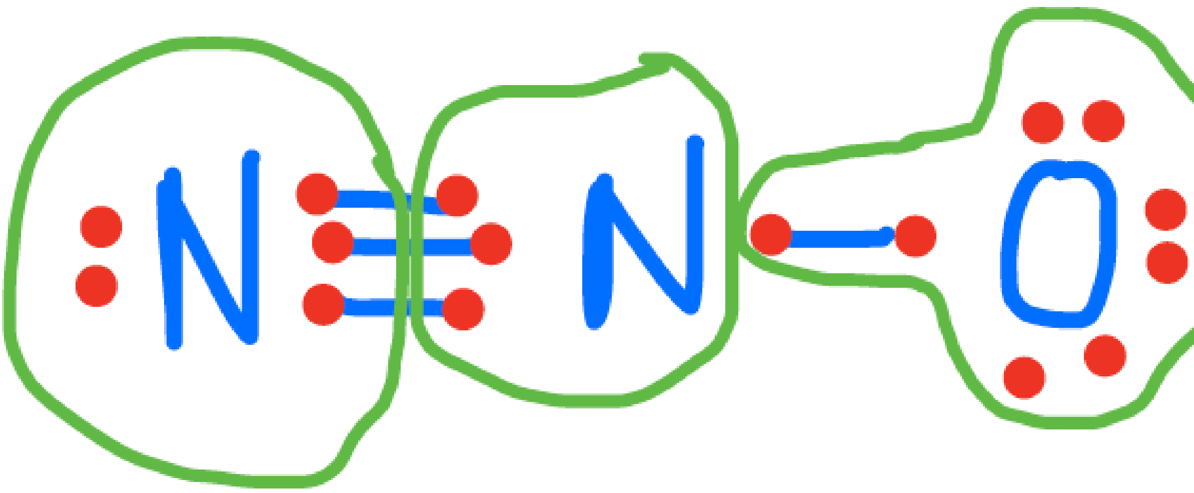


sulfur dioxide

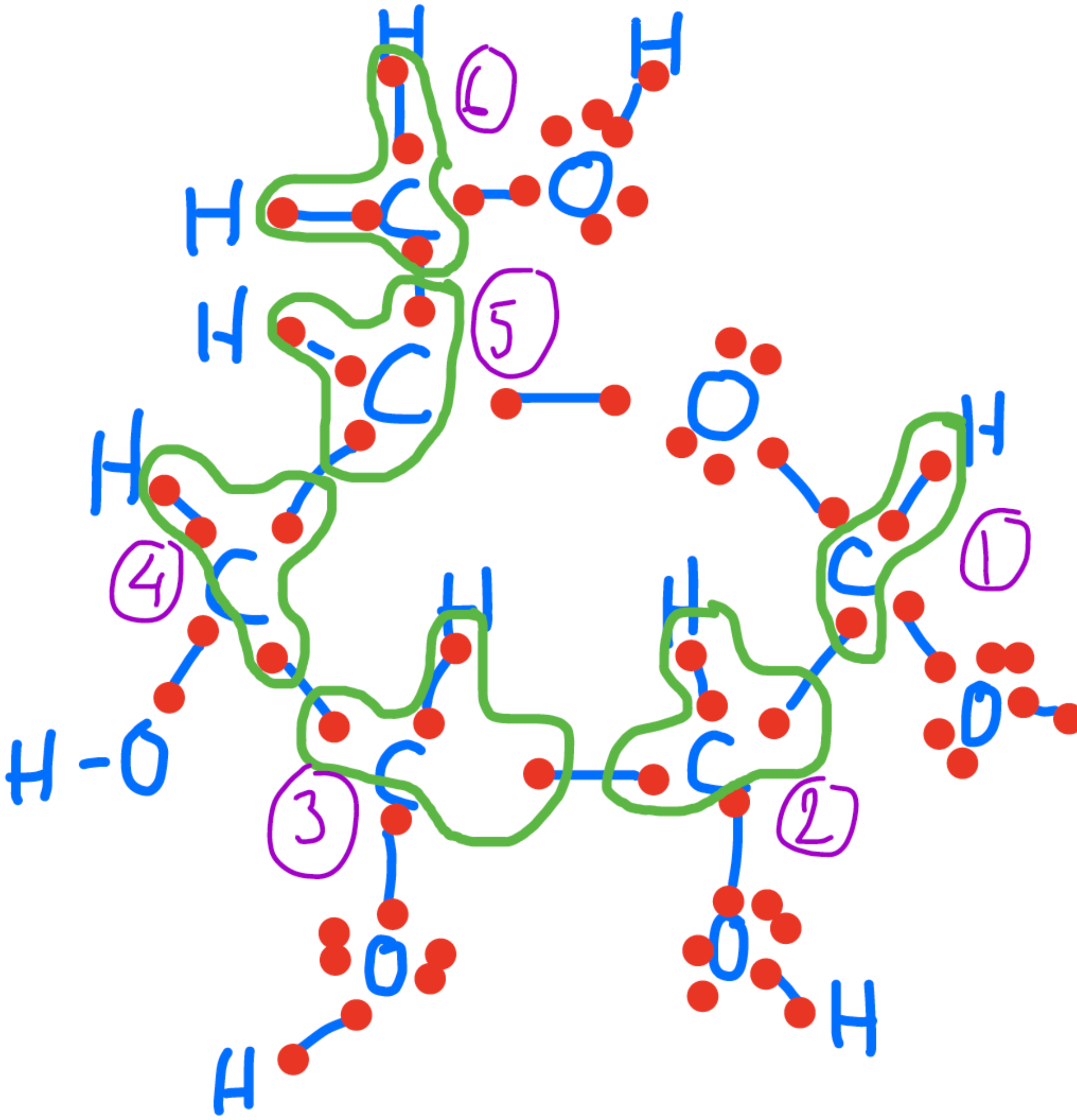
3



C#1 of glucose

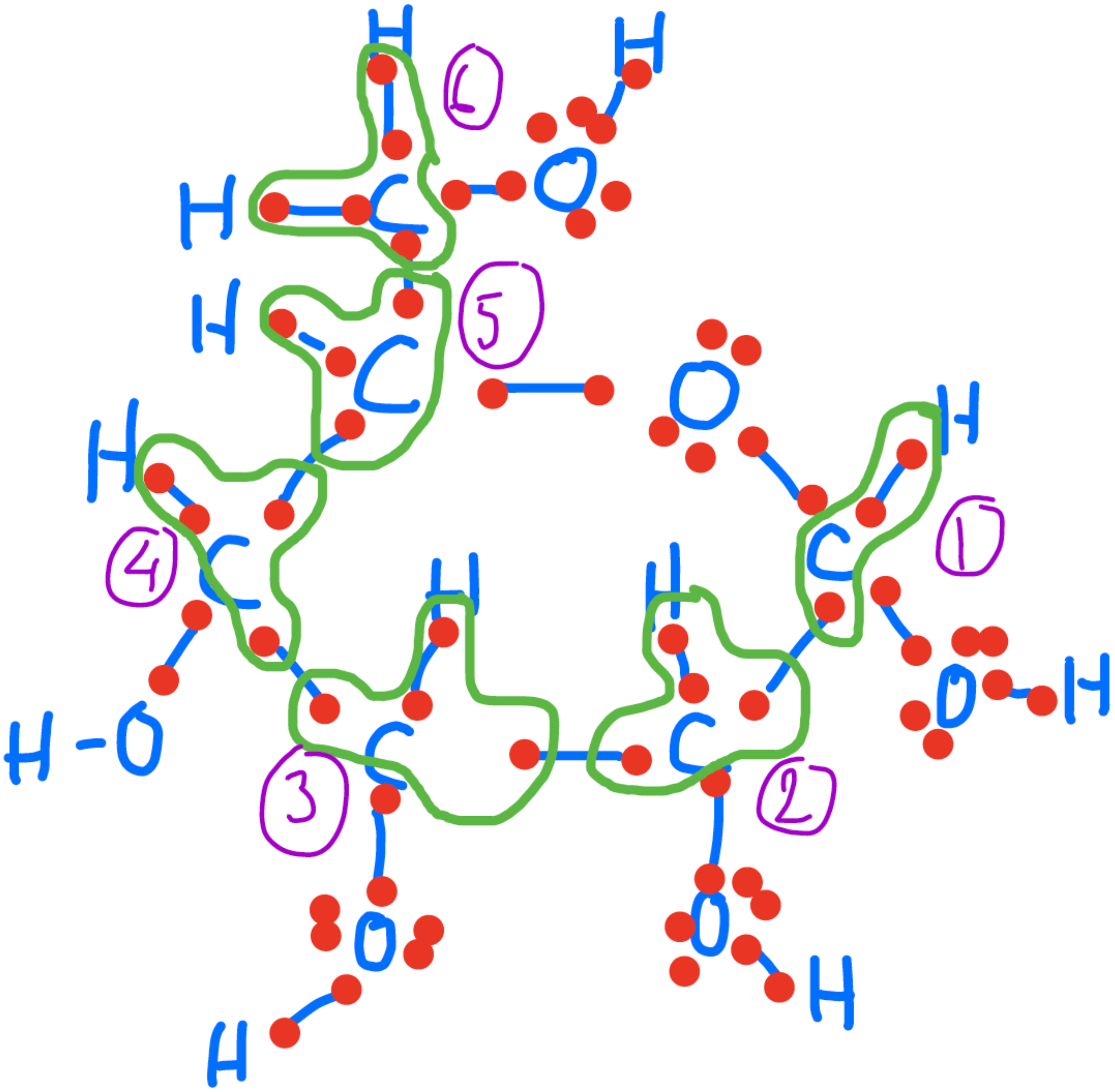


N#2 of nitrous oxide

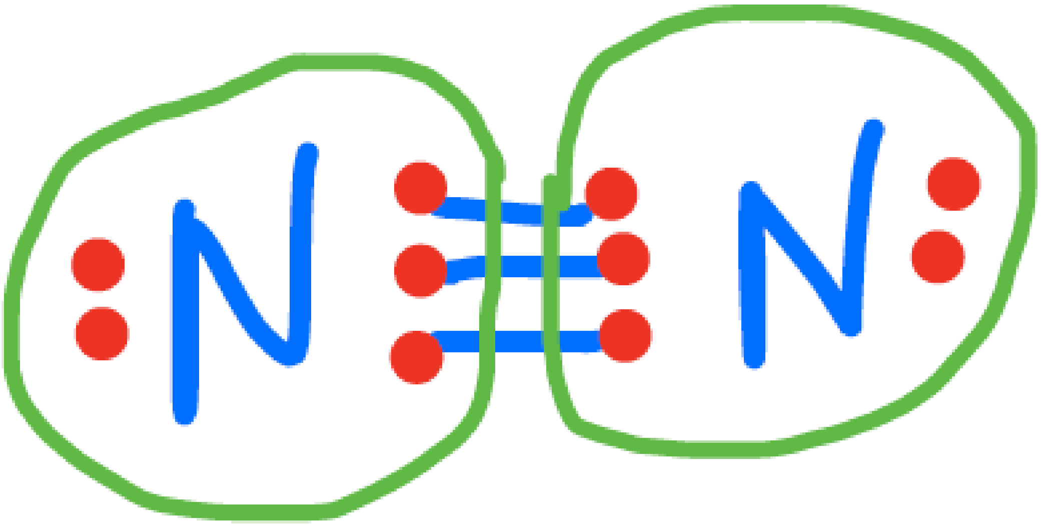


C#2 to C#5 of glucose

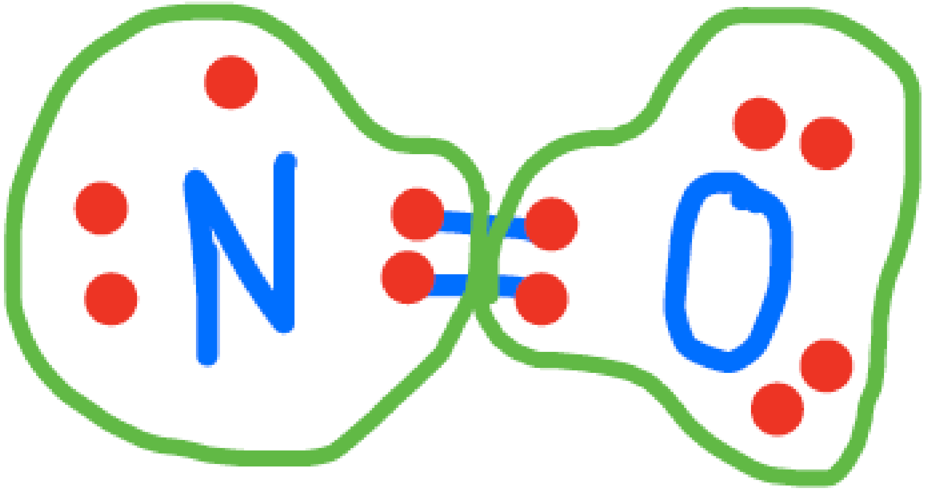
5



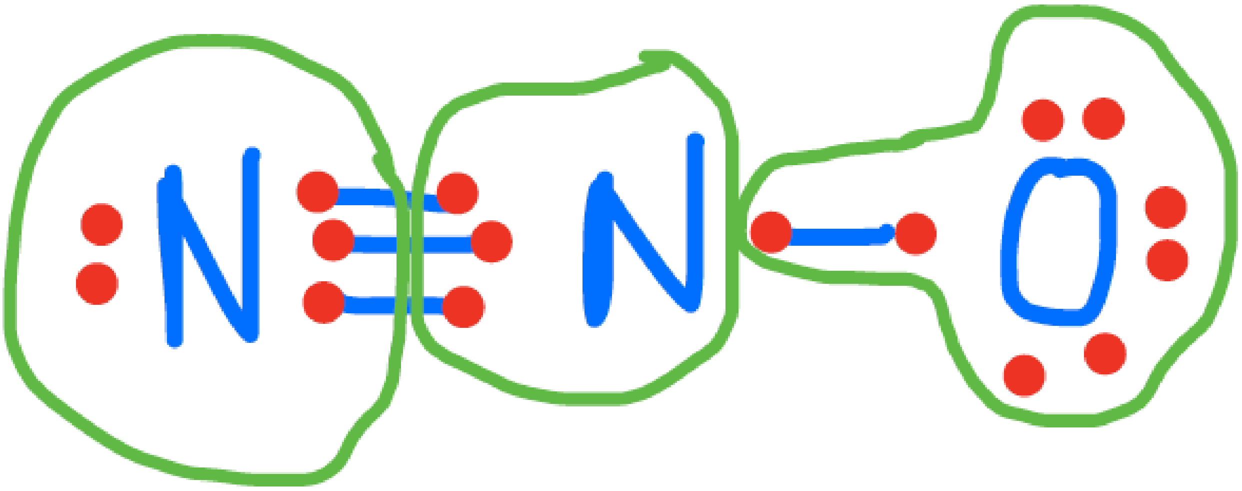
C#6 of glucose



dinitrogen ni-

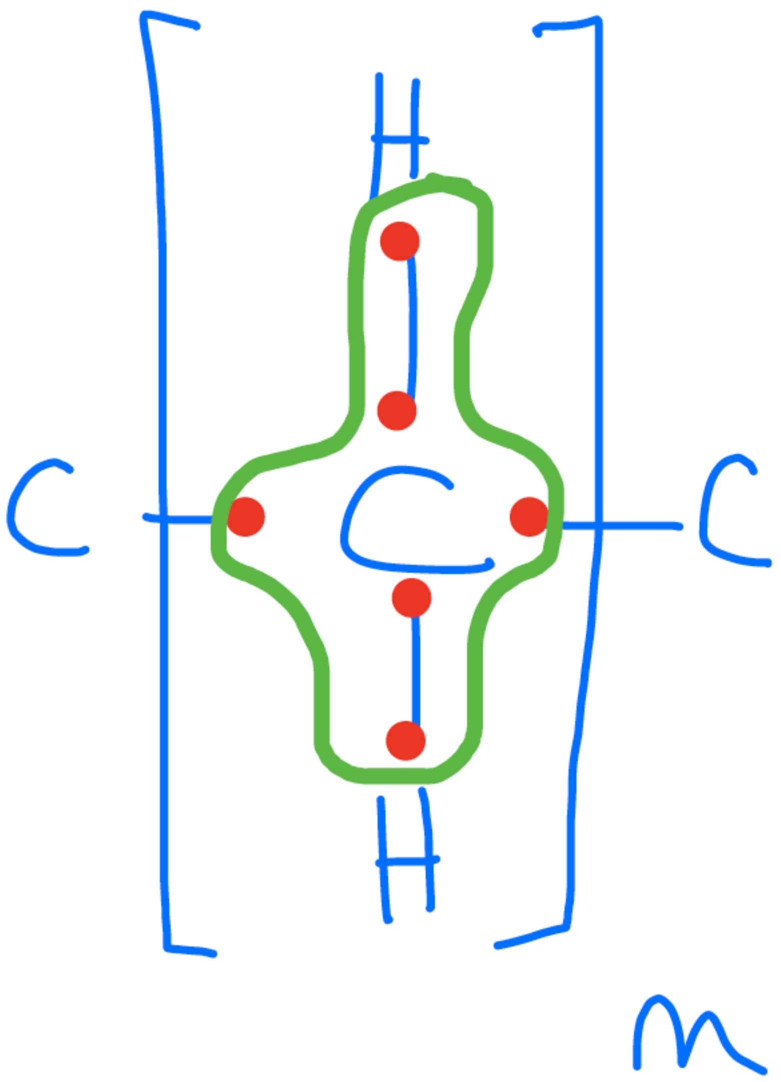


trogen monoxide N#1 of

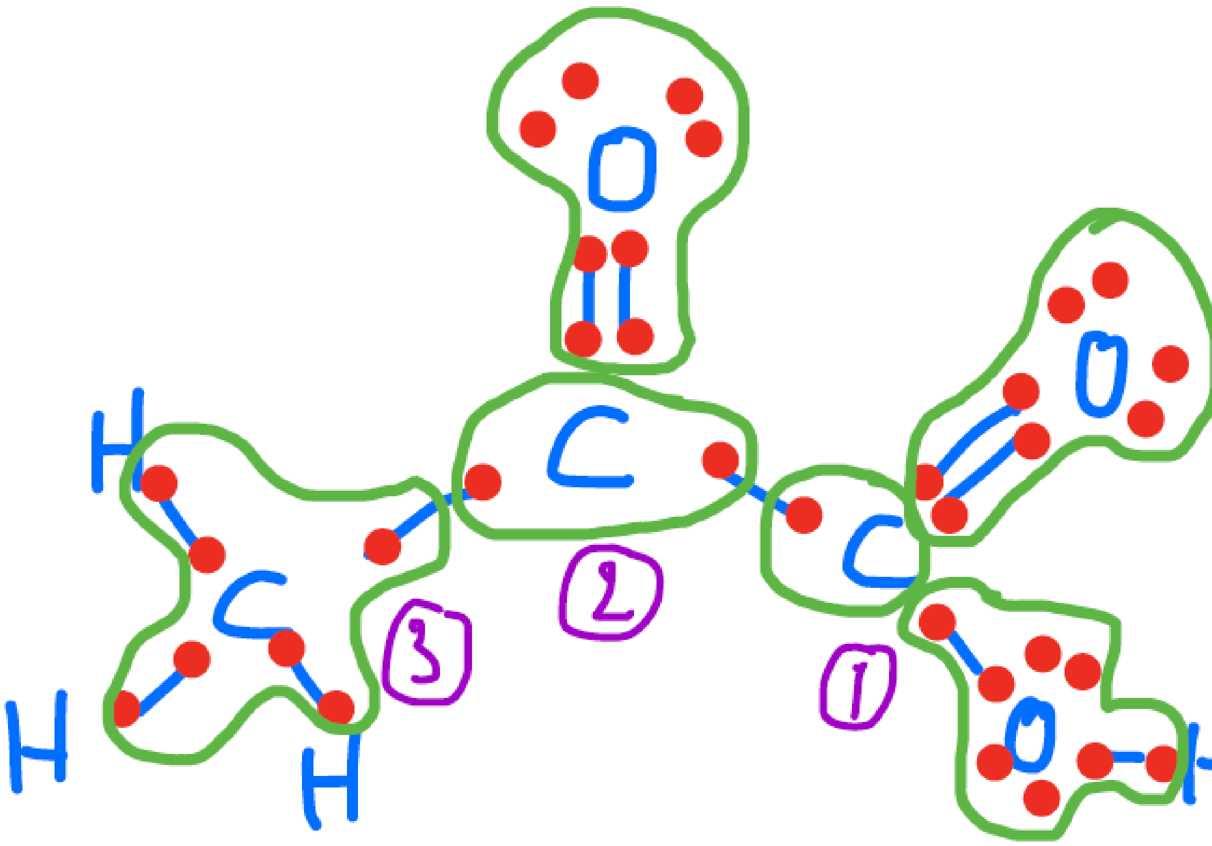


nitrous oxide

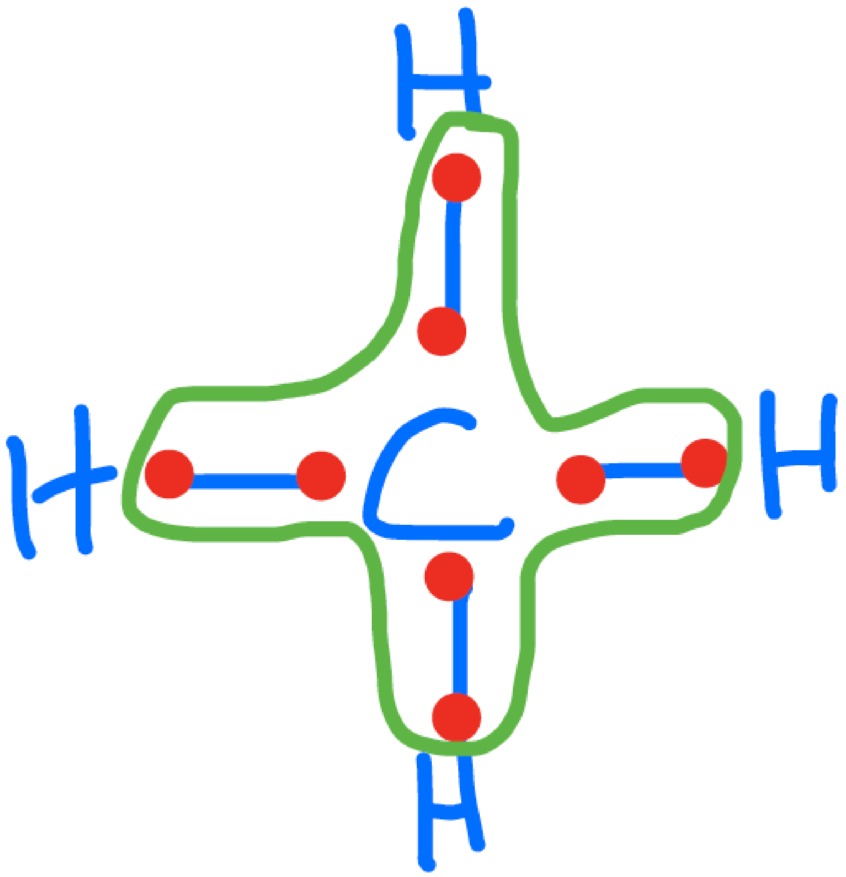
6



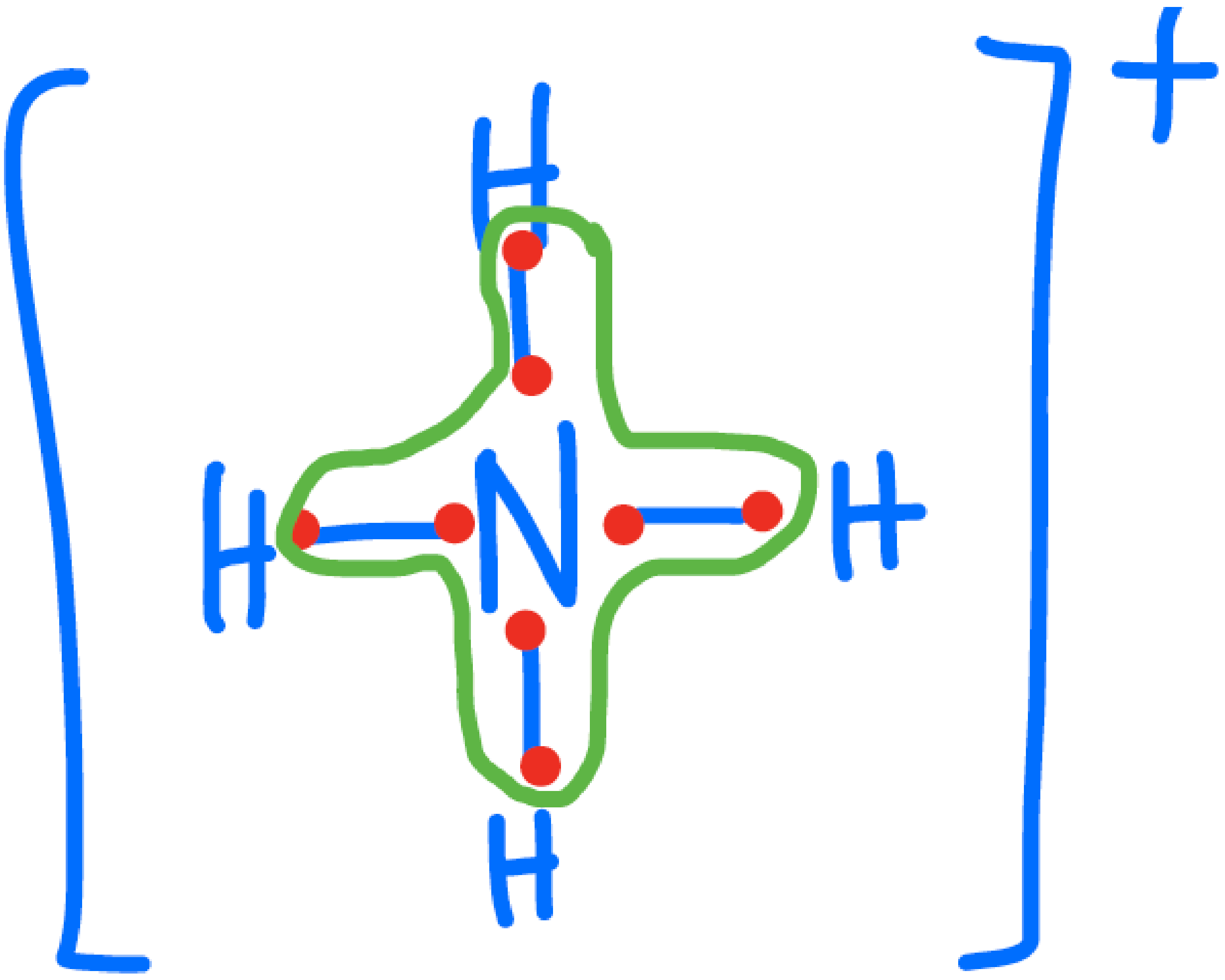
C of fatty acid



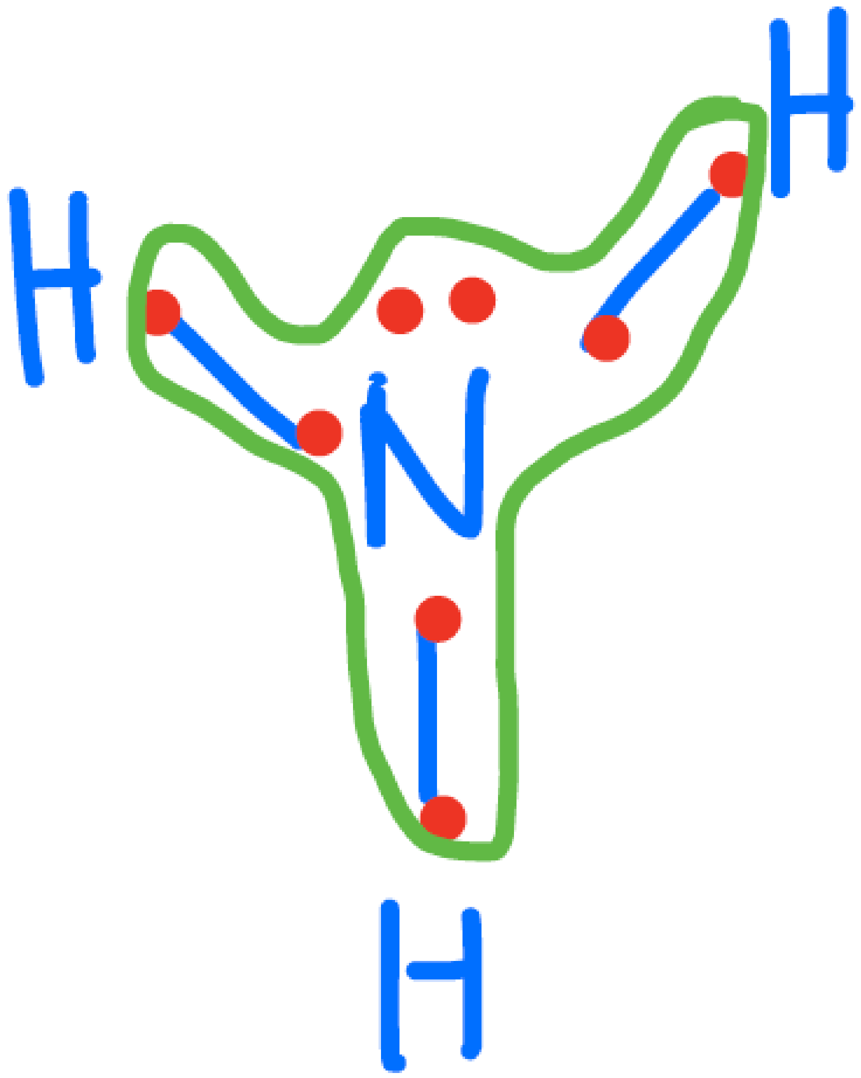
pyruvic acid (C#3)



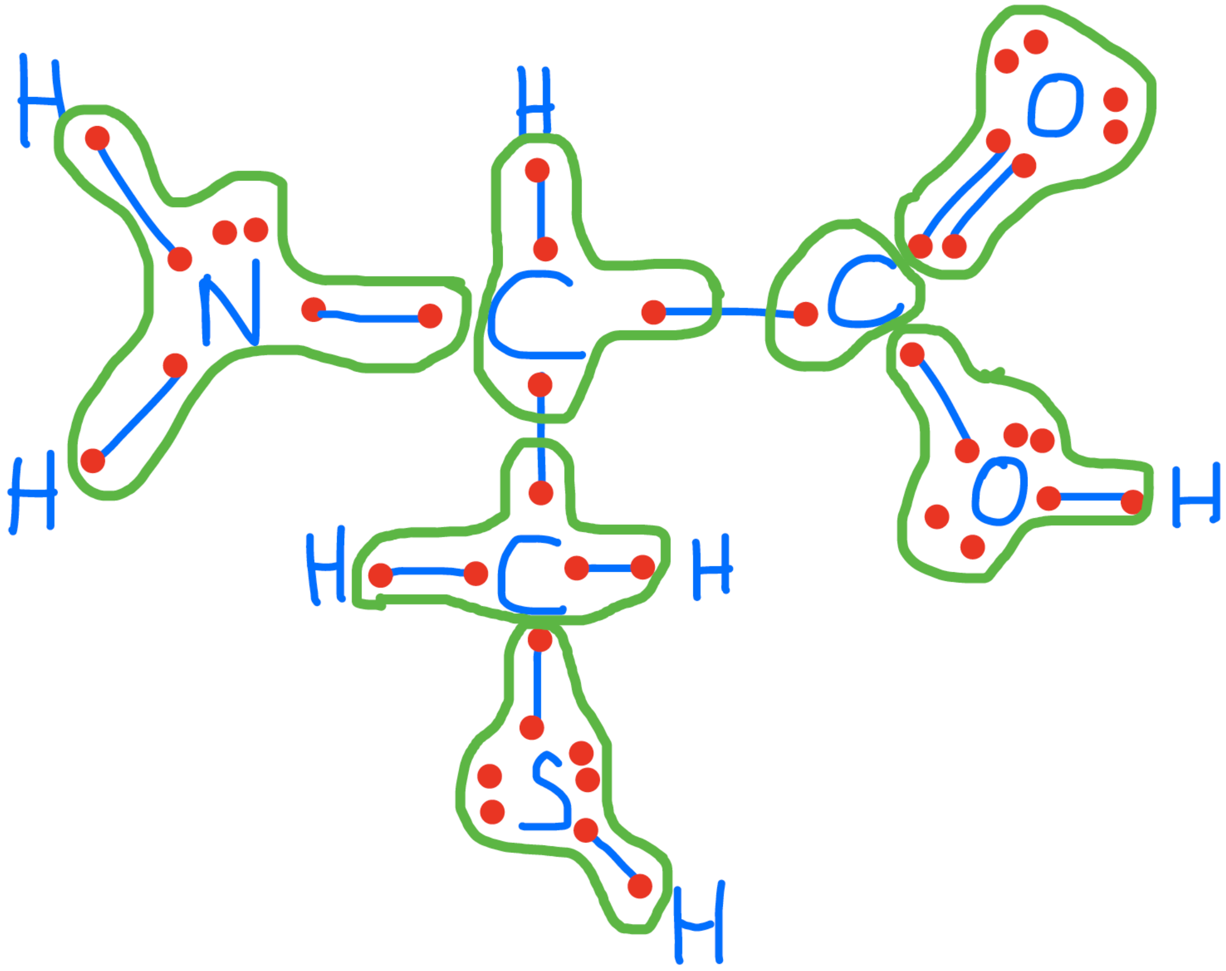
methane



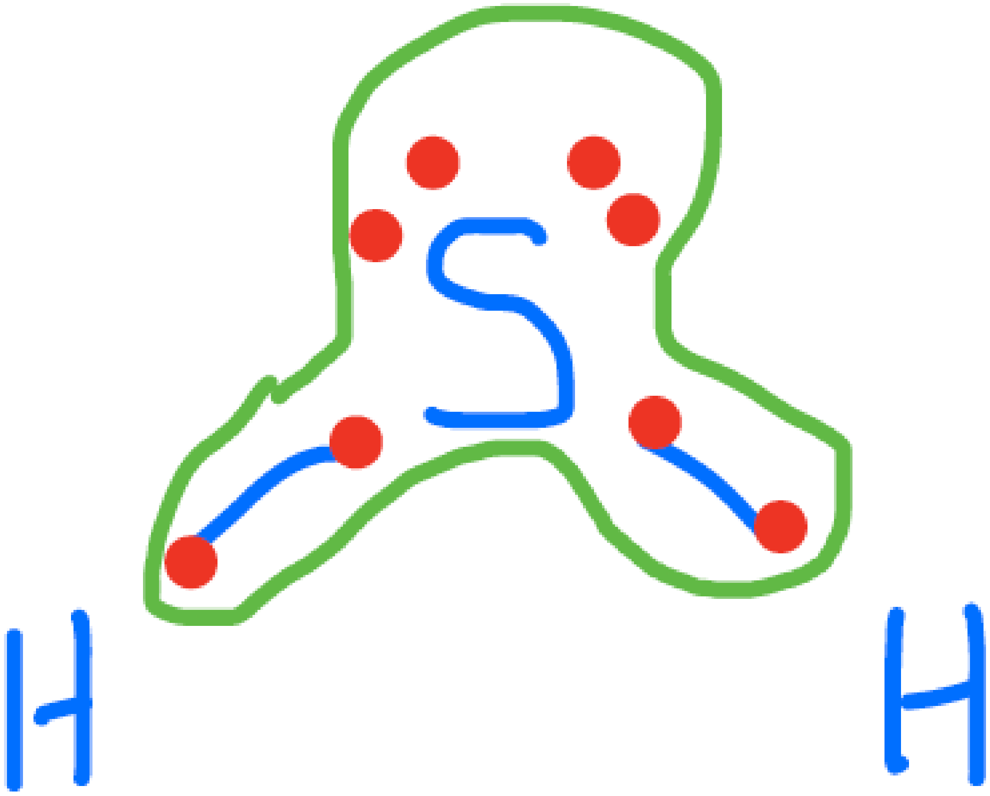
ammonium



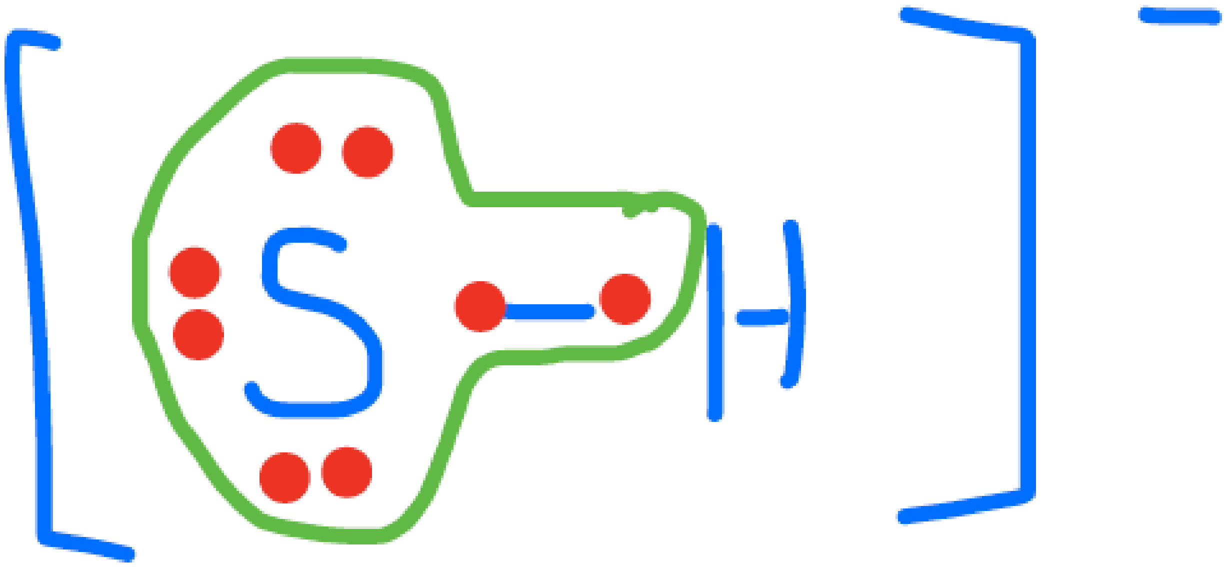
ammonia amine groups in



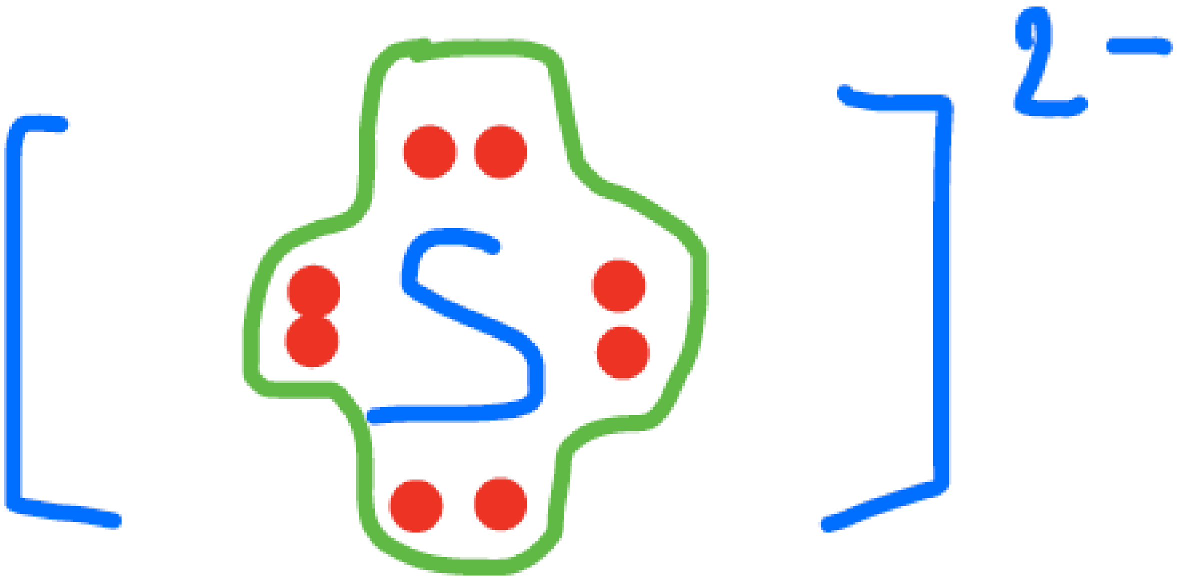
amino-acids



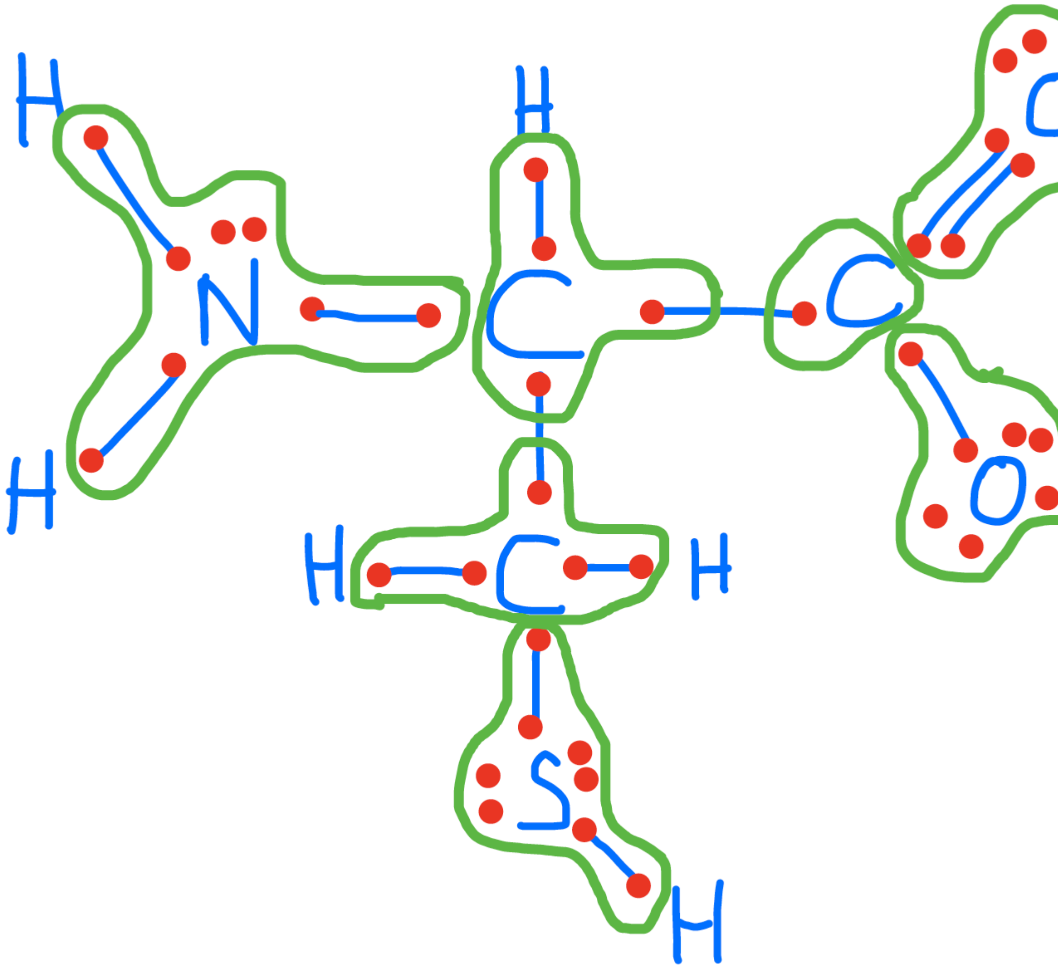
dihydrogen sulfide



hydrogen sulfide



sulfide



thiol groups in organic molecules

4.6 Other important things about code chunks

Code chunks (but not inline code) can take a lot of **options** which modify how they are run, and how they appear in the document. These options go after the initial `r` and before the closing `}` that announces the start of a code chunk. One of the most common options turns off printing out the code, but leaves the results alone: `` ```{r, echo=FALSE} `` Another runs the code, but includes neither the text of the code nor its output. `` ```{r, include=FALSE} `` This might seem pointless, but it can be useful for code chunks which do set-up like loading data files, or initial model estimates, etc.

Another option prints the code in the document, but does not run it: `` ```{r, eval=FALSE} `` This is useful if you want to talk about the (nicely formatted) code. Another option on the results of the code is that it generate all results "as-is", which is very nice when your code generates mark-up text to be rendered by `Pandoc`.

`` ```{r, results="asis"} `` By default, the results of a chunk with have `##` as a prefix. You can remove this by putting

`` ```{r, comment=FALSE} ``

Sometimes, running of the code will generate warnings and messages. These can be turned off in the output by using

`` ```{r, warning=FALSE, message = FALSE} ``

4.7 Naming Chunks

You can give chunks names immediately after their opening, like `` ```{r, clevername} ``. This name is then used for the images (or other files) that are generated when the document is rendered.

4.8 Adjusting figure sizes and alignments

These details are discussed in the accompanying written article on instantaneous vs. interval-average flow data.

4.8.0.1 "Caching" Code Chunks (Re-Running Only When Changed)

By default, R Markdown will re-run all of your code every time you render your document. If some of your code is slow, this can add up to a lot of time. You can, however, ask R Markdown to keep track of whether a chunk of code has changed, and only re-run it if it has. This is called **caching** the chunk.

```r
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

One issue is that a chunk of code which hasn't changed itself might call on results of earlier, modified chunks, and then we *would* want to re-run the downstream chunks. There are options for manually telling R Markdown "this chunk depends on this earlier chunk", but it's generally easier to let it take care of that, by setting the `autodep=TRUE` option.

1. If you load a package with the `library()` command, R Markdown isn't smart enough to check whether the package has changed (or indeed been installed, if you were missing it). So that won't trigger an automatic re-running of a cached code chunk.
2. To manually force re-running all code chunks, the easiest thing to do is to delete the directory R Markdown will create (named something like *filename*`_cache`) which it uses to store the state of all code chunks.

4.8.0.2 Setting Defaults for All Chunks

You can tell R to set some defaults to apply to all chunks where you don't specifically over-ride them. Here are the ones I generally use:

This sets some additional options beyond the ones I've discussed, like not re-running a chunk if only the comments have changed (`cache.comments = FALSE`), and leaving out messages and warnings. (I'd only recommend suppressing warnings once you're sure your code is in good shape.) I would typically give this set-up chunk itself the option `include=FALSE`.

You can over-ride these defaults by setting options for individual chunks.

4.8.0.3 More Chunk options

See [http://yihui.name/knitr/options/] for a complete listing of possible chunk options.

Chapter 5

Referencing and Cited References

5.1 Autoreferencing

The beauty of the system is that the referencing is all automatised.

- So to reference an equation use `\@ref(eq:HnApoly)` to obtain equation (3.2)
- So to reference a figure use`\@ref(fig:molarfracPO4)` to obtain figure 4.3
- So to reference an table use`\@ref(tab:ElecAllocTab)` to obtain table 4.5

5.2 Cited references

I have created and maintained for over 15 years a database written in Microsoft Access of around 2000 articles, most of them photocopied and typed in the database by hand during and after my Ph.D. (yes, I should have a medal!…) but I have to admit that this system is past its time… I have now switched to Paperpile. There are many other referencing software out there now (a list and comparison can be found here). The beauty of them is that they will all export in recognized formats and this is what we care about here. So no more copy/paste of references from some pdf. We are going into *reproducible research*. This means that all the articles you want to work with will be entered in the reference database of your choice. Make sure you start one ASAP!

In the YAML header, you can now see `bibliography: [book.bib, packages.bib]`. This means that there are files named `book.bib` and `packages.bib` that are located in the same directory as the all the .Rmd files, which when we 'knit', Pandoc will look for. You can access these `.bib` file in the GitHub directory. There is actually a possibility to add within the `.Rmd` document itself, the text needed for Pandoc to read the references, but unless you only have very few references, this can become very messy quickly. See the RStudio Bibliographies documentation for more details.

In short, in this file, there is a list of articles which information is coded according to the format chosen. By the way, a lot of the details for the bibliography are available on the RStudio Bibliographies documentation. R Markdown will recognize up to 10 different formats. The one Paperpile exports is `.bibtex`, so this is the one I have usually add in the YAML header. The filename extension will determine the *rendering* process, so make sure you have the right extension as well.

So, in the `.bib` file I have, the first article appears as

```
@ARTICLE{Kuhne2009-tn,
  title   = "Improving the Traditional Information Management in Natural
  Sciences",
  author  = "K{\"u}hne, Martin and Liehr, Andreas W",
  journal = "Data Sci. J.",
  volume  =  8,
  pages   = "18--26",
  year    =  2009,
  issn    = "1683-1470, 0308-9541",
  doi     = "10.2481/dsj.8.18"
}
```

here is another reference (Maxwell et al., 2018) or this website (Wikipedia contributors, 2018)

The first item after "ARTICLE{" is the unique identifier for the article. The identifier of this article is `Kuhne2009-tn`. When one wants to cite this article, one will say something like "reproducible research has been suggested to become the norm (Kühne and Liehr, 2009)". And you would code like

```
"reproducible research has been suggested to become the norm [@Kuhne2009-tn]"
```

If you want to say that "Kühne et al. (2009) have shown that etc.", you would add a "dash" just before the "@" and code it as such:

```
Kühne et al. [-@Kuhne2009-tn] have shown that etc."
```

If you want to cite several references, you would add a semicolon between the two such as in:

```
"reproducible research has been suggested to become the norm [@Kuhne2009-tn; @Buckheit1995-ls]"
```

Notice that among all the fields from the example above, there are `doi` and `issn`. `doi` stands for "Digital Object Identifier". `issn` stands for International Standard Serial Number (ISSN), which is an eight-digit serial number used to uniquely identify a serial publication. The `doi` value in this example is `10.2481/dsj.8.18`, which is unique in the world. These `doi` are applied to articles but not only. They are also applied to data. Eventually, all data that will be used in an article will have to have its own `doi` and all the codes that are used to analyze the data will refer to this unique `doi`. This is not quite implemented yet (in 2017), but will likely be by 2022. `doi` or `url` (not added here; stands for Uniform Resource Locator. Quite a mouthful, really, for what it is: a web address) are not necessarily exported from your reference software by default. Make sure you add this possibility, as `doi` is almost routinely added in the reference list in most journals.

The reference list of the citations will appear right after you place

```
# References
```

at the bottom of your Rmd docucment, if it a single Rmd document, or as a separate chapter for a book. When rendering your document, the list will appear automatically afterwards, and if you have in text notes, these will appear underneath.

5.2.1 Link to citations

In a single Rmd document (not a book) One very nice feature is to create hyperlinks from the in-text citations to the citations in the reference section. For this, just add `link-citations: true` in the YAML header.

5.2.2 CSL and styling of citations

CSL stands for Citation Style Language. The CSL line command is an option for the citation styles to be used in your document. You can comment it out by adding a "#" in front of it and the default .CSL file will apply without you noticing it. Each journal has its own way of handling how an article/reference should be cited in the text, and in the reference section, and there are hundreds of different styles out there... You can read lots of details on this CSL primer about how all this works.

While I was writing this tutorial, I did not specify at first the citation style. And I kept getting, using the same example as before, "(Kühne and Liehr 2009)", although I wanted "(Kühne and Liehr, 2009)", i.e., with a comma between the authors and the date, because this is the way I always did it and I think it is a lot better this way. Then I started thinking about the journals for which the inline citations are numbers, sometimes in brackets, sometimes without brackets… what a nightmare…! Actually it is extremely simple: all this is done automatically when you specify the CSL corresponding to the journal style following which you wish to write.

For this, you can pick at https://github.com/citation-style-language/styles the *.CSL file you are looking for (actually there are too many of them and they are not all displayed). The fastest way is to Google " CSL", and you will land on the CSL file you are looking for. I recommend that you click on the `raw` icon on GitHub, and copy all the file in a text editor. Warning! You need to make sure that your text editor does not add any weird formatting or add an extension at the end of the file. I had that problem and I could not see the extension on my computer, although I had the option to display so. Save the file in the same directory as your .Rmd.

So, to go back with my struggling to style the inline citation and the "missing comma", it turns out that the default CSL file uses a "Chicago author-date format" (I am not sure what this means exactly), which in text styling is "(author date)", without the comma…! If you use "journal-of-hydrology.CSL", you will see that all of a sudden after you `knit`, the commas just appeared…! Eureka! If you use "nature.CSL" (because we all want to be ready when we publish in Nature!), you will see that there are no more in-line citations, just numbers in superscript, hyphenated when needed, and the references are all in order, with the journal names in italics, the journal volume in bold, the year in parenthesis at the end, etc.! Is not that wonderful? Without your doing anything, other than adding the citation properly in the text following the guidelines above.

You can, if you want and have a lot of time to waste, take existing CSL files and modify them to have your own custom citation style. Make sure you take an "independent" CSL file and modify it. Most of them are dependent upon a "source" or independent one, and the code in the CSL file is just saying how the dependent file should slightly alter the independent one. Good instructions on how to do this can be found on the excellent CSL primer.

Chapter 6

Final words

Now, it is your time to play!!

Bibliography

Kühne, M. and Liehr, A. W. (2009). Improving the traditional information management in natural sciences. *Data Sci. J.*, 8:18–26.

Maxwell, B. M., Birgand, F., Schipper, L. A., Christianson, L. E., Tian, S., Helmers, M. J., Williams, D. J., Chescheir, G. M., and Youssef, M. A. (2018). Drying–Rewetting cycles affect nitrate removal rates in woodchip bioreactors. *J. Environ. Qual.*

Wikipedia contributors (2018). Denitrification. https://en.wikipedia.org/w/index.php?title=Denitrification&oldid=872283218. Accessed: 2018-12-17.