

Running Perl scripts

Ver. 1.4

Thomas Hamann

Copyright: Document copyright © Thomas Hamann/Naturalis Biodiversity Center 2012-2016. This document is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) license.

This project was subsidized in part by the EU project “pro-iBiosphere” (Grant agreement 312848).

Introduction

This document shortly explains:

- How to install Perl
- How to run Perl from the Windows Command Prompt
- How to use the FlorML Perl scripts

It is assumed that you are using the Windows operating system.

Table of Contents

Running Perl scripts.....	1
Introduction	1
Installing and setting up Perl	3
The Perl operating environment: The Command Prompt.....	4
Starting the Command Prompt.....	4
Using the Command Prompt: Basics.....	5
Changing drives and folders.....	7
Running FlorML Perl scripts	9
Basics.....	9
Important notes on file names	10
Keeping track of the semi-automated mark-up process	11
Changing the file extension of files that are processed.....	12
Checking the output and/or making manual changes.....	12
Further actions.....	14
Appendix I: FlorML Script running orders.....	15
Running order for Flora Malesiana	15

Bug fixing scripts	16
Running order for Flore du Gabon.....	17
Running order for Flora of the Guianas	18

Installing and setting up Perl

Before you can run your Perl scripts, you should have the Perl programming language interpreter installed. To do this, you may need help from IT support, because Perl will have to be installed on your own computer and you need to have additional permissions to be able to use it.

Perl can currently be had in two different flavours: Perl 5 and Perl 6. You will be using **Perl 5**. Some operating systems, especially Unix-based ones, already include Perl, but it is preferable to install the latest version available for your computer system. At least use version **5.14** or higher, because that version has good Unicode support (support of foreign symbols and letters). You will need this to properly work with the special symbols that are present in many taxonomic works. Perl and extensive documentation can be found at <http://www.perl.org>. If you are on Windows, install the latest version of ActivePerl Community Edition (<http://www.activestate.com/activeperl/downloads>). As of March 2016, the current version of Perl is 5.22.1.

After installation, ensure that you have enough additional Windows permissions to be able to run Perl scripts by testing whether you can run a script (see next sections for instructions). It might be preferable (to be able) to run the scripts locally instead of over the network due to latency (i.e. slowdown) and reliability issues. Furthermore, Perl can be extended with additional modules, the installation of which also require more permissions. It is perhaps best to ask your IT support to be made a local administrator to parts of your computer.

Perl runs from the Command Prompt, the use of which is explained in the next section.

It is useful to ensure that Windows shows file extensions in Windows Explorer and other programs. See the "Image processing and metadata addition for online e-Flora use" manual (**image processing.doc**) under "Windows set-up" step 1 for instructions.

Why Perl? It has been suggested that other programming languages, such as Python, can also be used to perform these tasks. Although this is indeed true up to a degree, Perl has the advantage of having been created specifically to handle text. Any text. Plain text, HTML, XML, foreign writings, etc. Several features that are practical when handling text are very simple to implement in Perl. For example, Unicode support is a matter of adding *only two* distinct lines at the top of a script. Furthermore, Perl makes it possible to do the same things in various ways, therefore suiting various types of users. Finally, it has really good documentation and a community that is friendly to newbies and oldbies alike.

The Perl operating environment: The Command Prompt

Starting the Command Prompt

To use Perl, you need to use the Windows Command Prompt, which is a text-only non-graphical user environment. If you ever used MS-DOS, you will know what to expect. The following explains where to find the Command Prompt.

- 1) First, click on the Start menu, and then click on "All Programs" (Figure 1).

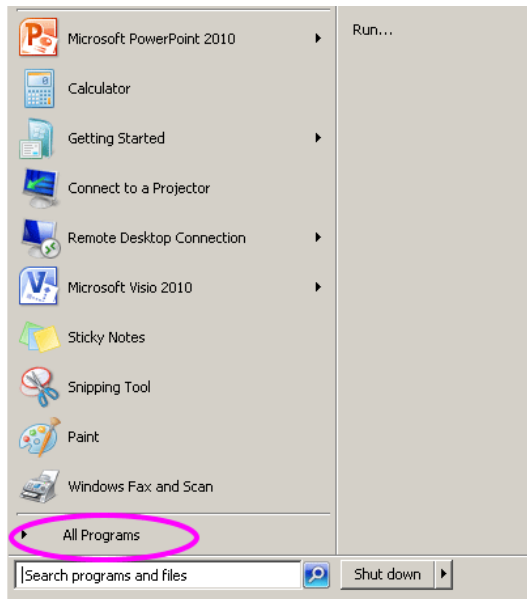


Figure 1: "All Programs" in the Start menu.

- 2) Now click on "Accessories" (Figure 2).
- 3) Click on "Command Prompt" (Figure 3) to open it.

Tip: To make it easier to access the Command Prompt, it can be pinned to the Start menu or the Task bar. This is achieved by performing steps 1 and 2 above, then *right-clicking* on "Command Prompt" and selecting the appropriate option in the menu that shows up (Figure 4). The next time you need the Command Prompt you can then access it easily.

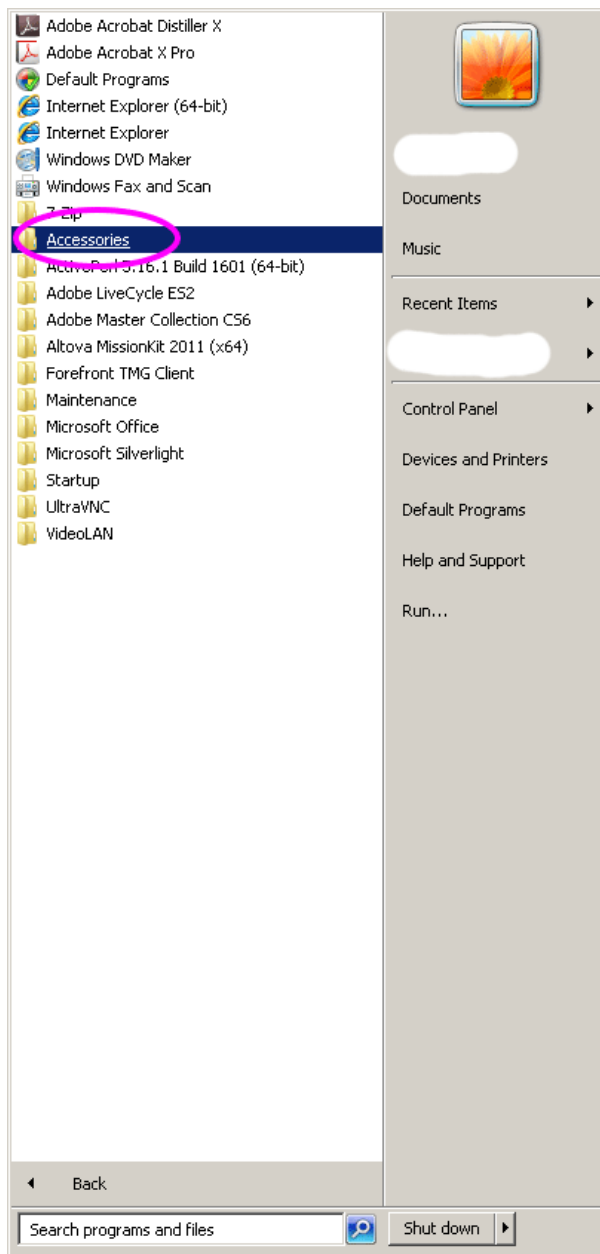


Figure 2: "Accessories" in the Start menu.

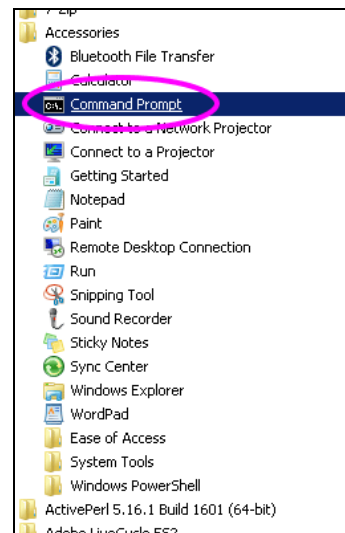


Figure 3: "Command Prompt" in Start menu.

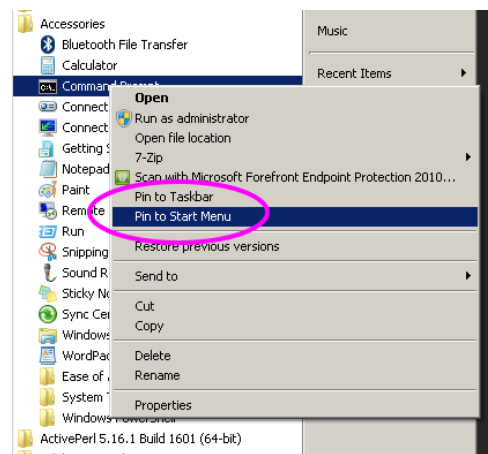


Figure 4: Options to pin the Command Prompt to either the Start menu or the Task bar.

Using the Command Prompt: Basics

The Command Prompt window is shown in Figure 5.

Conventions:

- When text is shown in `this font`, it is a command that you can type into the Command Prompt window.
- Drive and folder paths are always shown in *italics*, e.g. *H:*

By typing `perl -v` and hitting the "Enter" key you can check which version of Perl you have installed. You should see something similar to Figure 6.

This is also a quick check to see whether Perl was installed properly. If instead of version information you get an error message, look up on the internet how to set your Perl PATH properly (or get technical help from your IT support to do so).

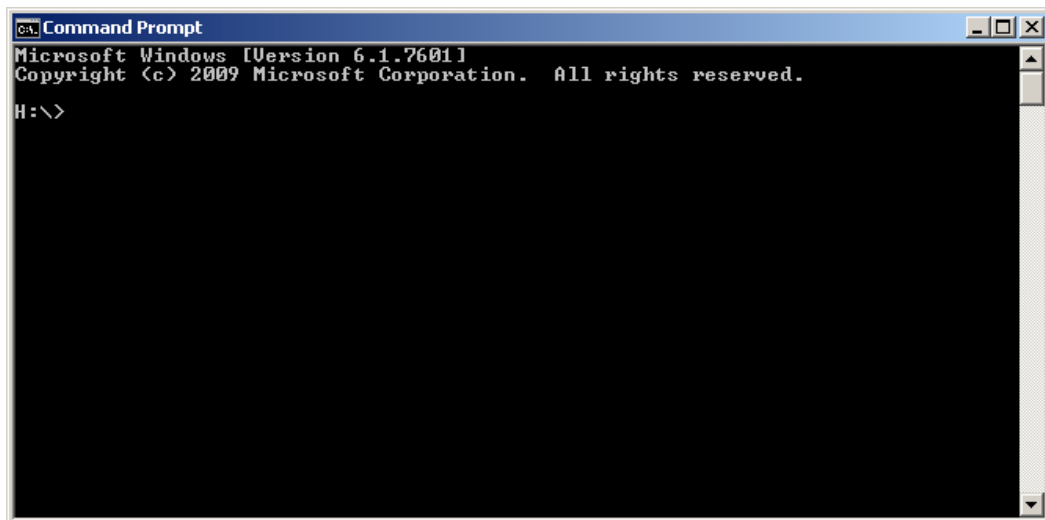


Figure 5: The Command Prompt window.

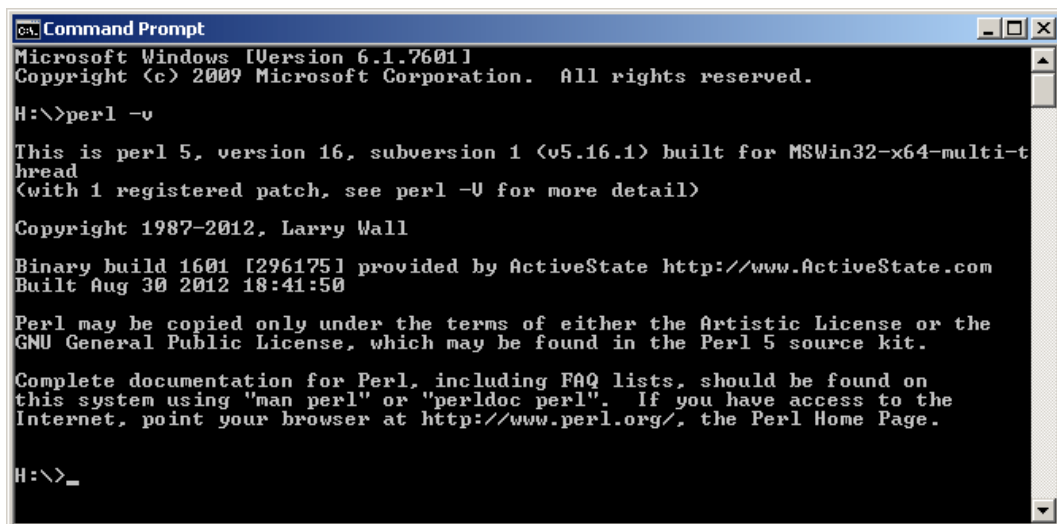


Figure 6: Perl version information.

Changing drives and folders

In this particular case, the current drive is the *H:* drive, as indicated by the *H:\>* prompt in Figure 6. To be able to run your Perl scripts, you have to change to the drive and folder containing them.

This same folder should contain the files you want to process with the scripts. It is recommended that you do not move the original cleaned up files to this folder, but instead copy them to the folder (see **text preparation.doc** for instructions on how to copy files).

In the following example, the scripts and files to be processed are located at *C:\Digital Flora Data and Utilities\Perl Procedures\Flore du Gabon*, which means they are in a folder called "Flore du Gabon" in another folder called "Perl Procedures" in yet another folder called "Digital Flora Data and Utilities" on the *C:* drive.

- 1) First, you type `c:` , and then press the "Enter" key. This will change to the *C:* drive, and is reflected in the change in prompt, which is now *C:\>* (Figure 7 **Error! Reference source not found.**).

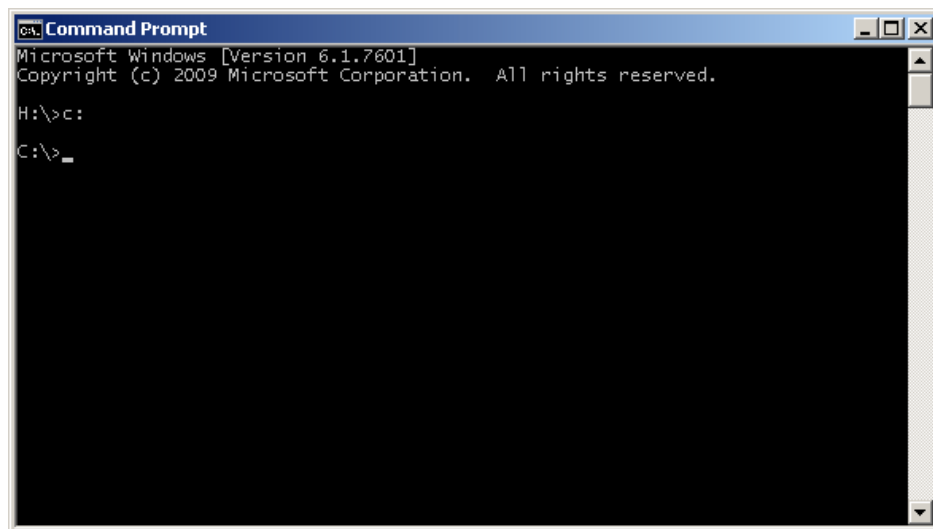


Figure 7: Changing from one drive to another.

- 2) You then type `cd digital flora data and utilities` and presses the "Enter" key. "`cd`" stands for "change directory" (directory = folder). Again, the prompt changes to reflect that the folder change was performed successfully (Figure 8).

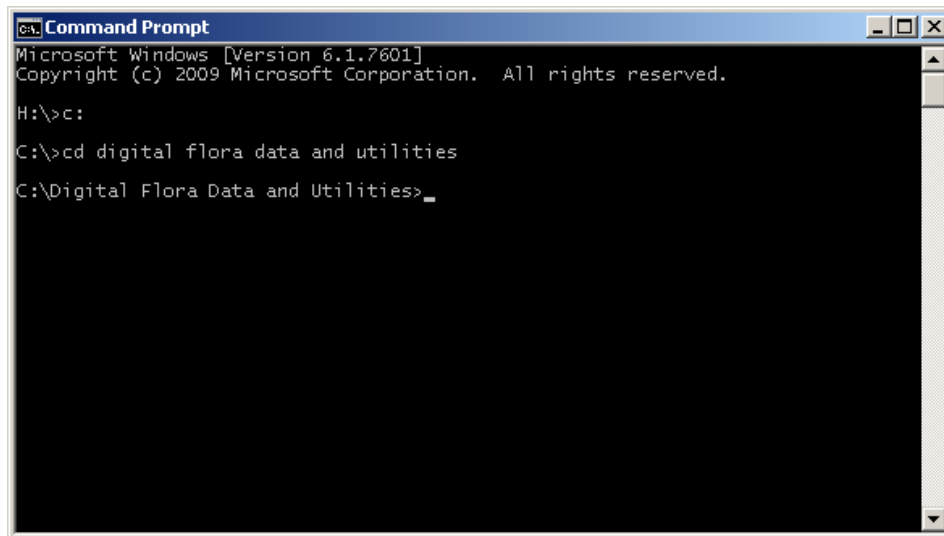


Figure 8: Changing to another folder.

- 3) You then repeat step 2 with the other folder names until you reach the destination folder. In this case you first change to the "Perl Procedures" folder, then to the "Flore du Gabon" folder (Figure 9). Each time the prompt shows in what folder you are.

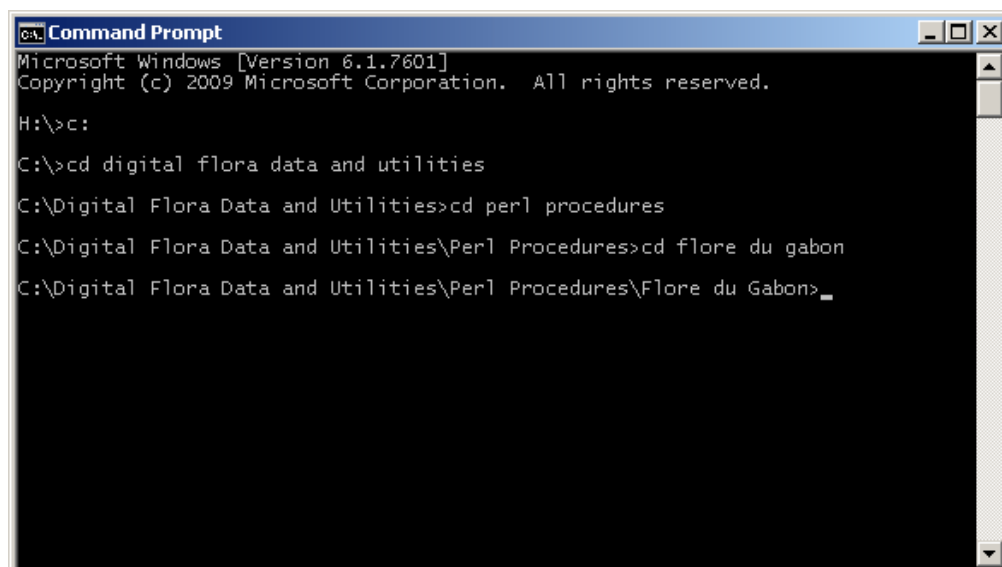


Figure 9: Changing to the destination folder.

(Note that the Command Prompt is not case sensitive, so uppercase letters are interpreted as lowercase letters, and inversely)

You can now try this on your own computer using your own folders. If you are unsure what your folders are called or on which drive they are located, start Windows Explorer and use that as a reference. For example, in Figure 10 the Windows drive has drive letter C:.

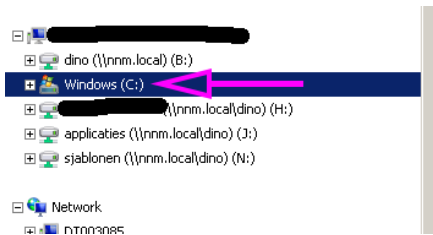


Figure 10: Drive letters in Windows Explorer.

Tip: Should you accidentally change to the wrong folder, you can type in `cd . .` (`cd` followed by 2 dots) in the Command Prompt window and press the "Enter" key to move back to the previous folder.

The Command Prompt window can be closed like any other Windows program, by clicking on the cross in the upper right corner. Doing this while running a script will abort the script.

Running FlorML Perl scripts

Basics

As was mentioned before, you need to be in the folder your Perl scripts are located in before being able to run them.

Furthermore, the files that the scripts will modify also need to be in the same folder. This is because the scripts do not use a fancy file handling system that allows files to be opened from and saved to other locations.

To run a Perl script, the following syntax is used:

```
perl [script_name] [input_file] [output_file]
```

In this:

`perl` starts the perl interpreter.

`[script_name]` is the file name of the script, including file name extension (`.pl` or `.plx`).

`[input_file]` is the file name of the input (source) text file to which the script will apply its modifications, including extension (`.txt` or `.xml`).

[output_file] is the file name of the output (destination) file that the script will write its modifications to, including extension (.txt or .xml).

An example is shown in Figure 11. Here a script called "atomiseur.plx" is run over an input file called "fdg9to11_10.xml", outputting to a file "fdg9to11_11.xml".

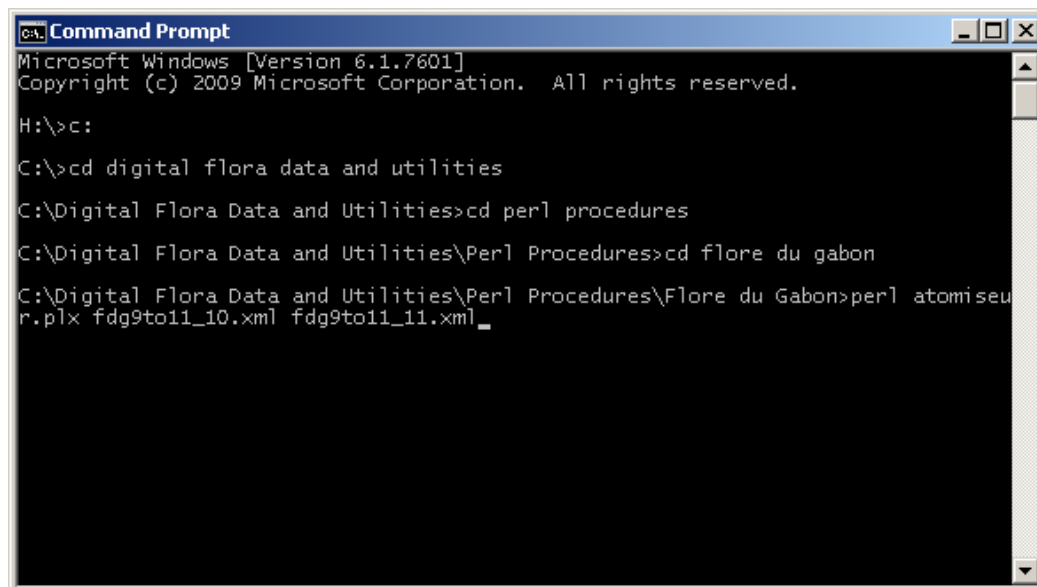


Figure 11: Script running example.

This is followed by pressing the "Enter" key, after which the script will run. Normally, there is no on-screen indication that the script is running except for a momentary pause, after which the prompt shows up again.

If there are errors in the script or improper syntax was used in the Command Prompt window, however, an error message will show up. In the case of script errors this error message will indicate where the error can be found.

Generally, the output file from one script is used as the input file for the next script.

Because each script performs one or more specific tasks, and subsequent scripts often are dependent on each other, the scripts need to be run in a specific order. The script orders for Flora Malesiana, Flore du Gabon, and Flora of the Guianas are given later on in this document.

Important notes on file names

When running scripts from the Command Prompt, it is useful to keep in mind that the scripts use rather primitive file handling. This means that you must exercise some care when typing in file names:

- Do not use the same file name for both the source and destination files.

- Do not use the same file name for the destination file as another already existing file.
- In both cases, the original file will be destroyed. Contrary to Windows programs, there will be no warning that an existing file is being overwritten.

If a file cannot be found, an error message will be displayed.

Furthermore, the Perl interpreter and the Command Prompt itself also put some limitations on file names. Therefore, the following conventions must be followed:

- There cannot be any spaces in the file names. The same applies to punctuation other than the dot preceding the file extension.
- It is therefore best that you only use alphanumeric characters (a-z0-9) and underscores (_) for file names.

This applies to all file names (scripts and input/output files). Not adhering to these conventions will cause problems, such as files not being found or files without an extension being created.

Finally, it is useful to take care to name files in such a way that you can easily recognise what they are:

- Scripts can best have names that describe what they do or what XML tags they insert. For example, a script used for atomisation can be called "**atomisation.pl**".
- Input and output files are best named to reflect their contents. Because of the aforementioned limitations with regards to file names and the risk of typing errors in long file names, it is suggested that you use an abbreviation for the title of the taxonomic work combined with the volume number. For example, a text file containing Flora Malesiana volume 10 could be called "**fmvol10.txt**".

Keeping track of the semi-automated mark-up process

To keep track of the stage of the semi-automated mark-up process you are at, it is suggested that you either give each output file a name that reflects the changes that were made to it, or use incremental file names.

In the latter case, using the Flora Malesiana volume 10 example above, you would do the following:

1. Start with **fmvol10.txt**
2. Run first script, name output file **fmvol10_1.txt**:

```
perl [script_name1] fmvol10.txt fmvol10_1.txt
```
3. Run second script, name output file **fmvol10_2.txt**:

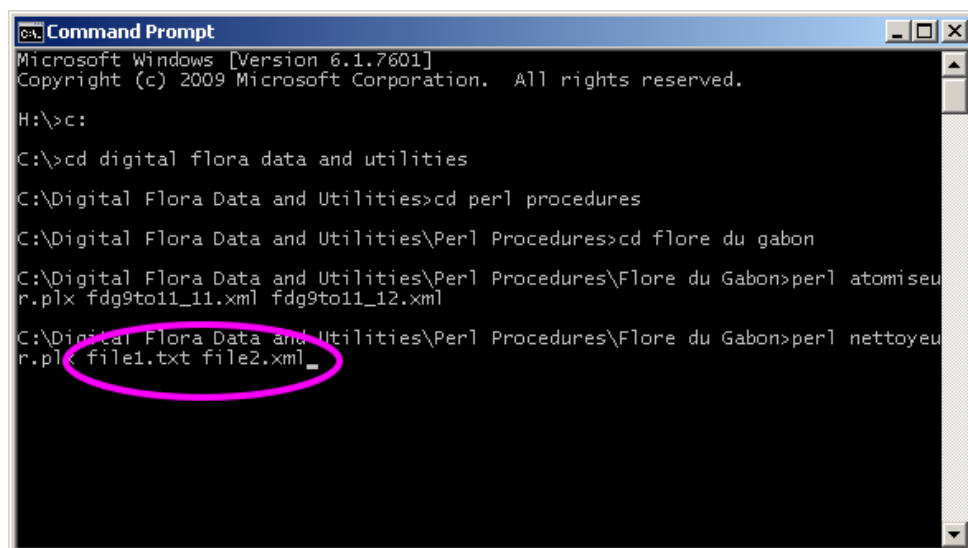
```
perl [script_name2] fmvol10_1.txt fmvol10_2.txt
```

4. Run third script, name output file **fmvol10_3.txt**:
`perl [script_name3] fmvol10_2.txt fmvol10_3.txt`
5. etc. (until you reach the last script)

You may want to use pen and paper to assist yourself in recalling the mark-up stage you are at.

Changing the file extension of files that are processed

Since both the input and output files are plain text-based files, and XML files are also a type of plain text file, it is very easy to change the file extension from .txt to .xml when running one of the first scripts. In Figure 12 the clean-up script for Flore du Gabon is run (using `perl nettoyeur.plx`) using input file `file1.txt` and output file `file2.xml`.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

H:\>cd c:

C:\>cd digital flora data and utilities

C:\Digital Flora Data and Utilities>cd perl procedures

C:\Digital Flora Data and Utilities\Perl Procedures>cd flore du gabon

C:\Digital Flora Data and Utilities\Perl Procedures\Flore du Gabon>perl atomiseu
r.plx fdg9to11_11.xml fdg9to11_12.xml

C:\Digital Flora Data and Utilities\Perl Procedures\Flore du Gabon>perl nettoyeu
r.plx file1.txt file2.xml
```

Figure 12: Changing the file extension in a simple manner.

Checking the output and/or making manual changes

You may want to check the output file now and then after running a certain script. This is very useful to do when you know you can expect errors (e.g. due to a bad OCR), you know a certain part of the document is problematic, or when debugging scripts during script development. Furthermore, in some cases there are manual steps between scripts due to contents that is difficult to automatically mark up.

In such cases, you can open the files in an advanced text editor, such as Notepad++ (<http://notepad-plus-plus.org/>). Do not use Microsoft Word for this, as it will try to interpret the XML tags, which you do not want.

You can see an example in Figure 13 **Error! Reference source not found..** Shown is a volume of Flore du Gabon during manual insertion of the elements for tables, lists, and line breaks.

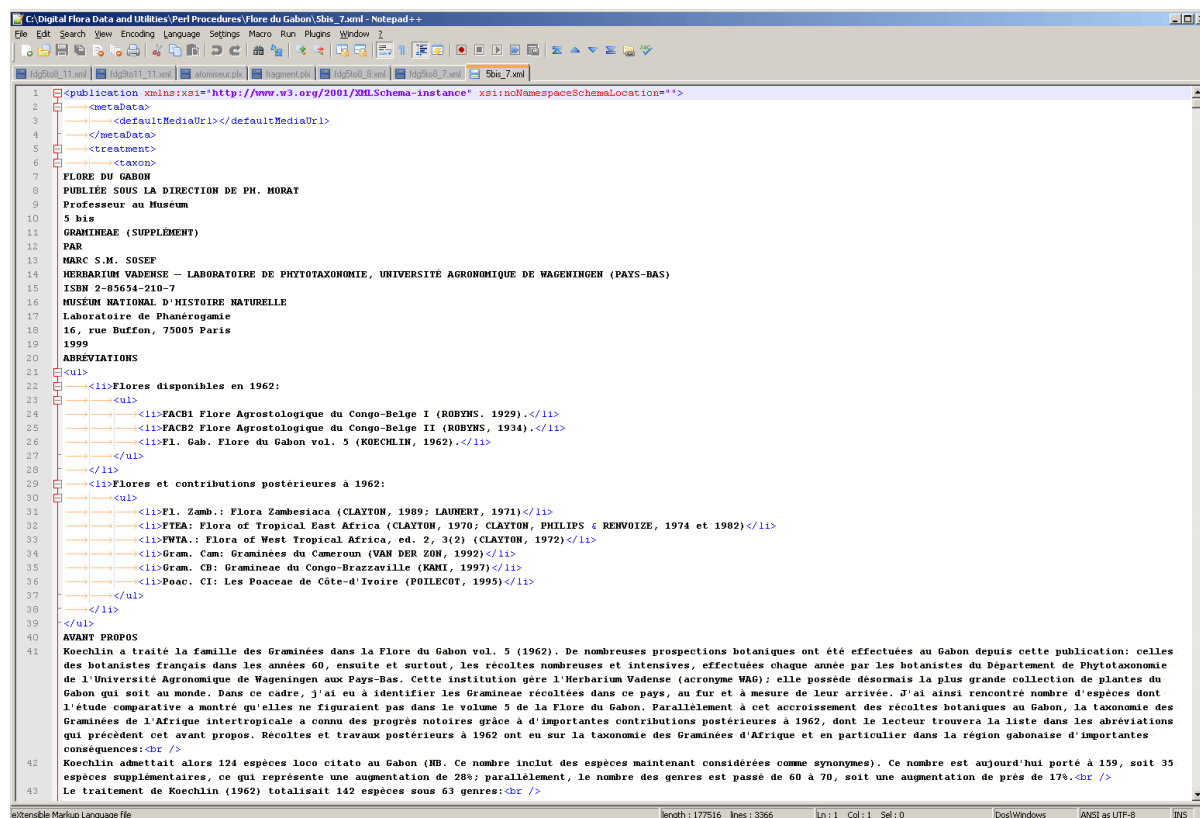


Figure 13: A volume of Flore du Gabon during manual insertion of table, list and line break elements.

For more information on XML elements that usually have to be inserted manually, please see the FlorML reference manual (**FlorML reference.doc**).

If your copy of Notepad++ is not showing the tab symbols and other white space or the text lines run off your screen, go to Notepad++'s "View" menu and switch on the following three options (see also Figure 14):

- "Word wrap", which wraps the text if a line is longer than your screen is wide.
- Under "Show Symbol", choose
 - "Show White Space and TAB"
 - "Show Indent Guide"

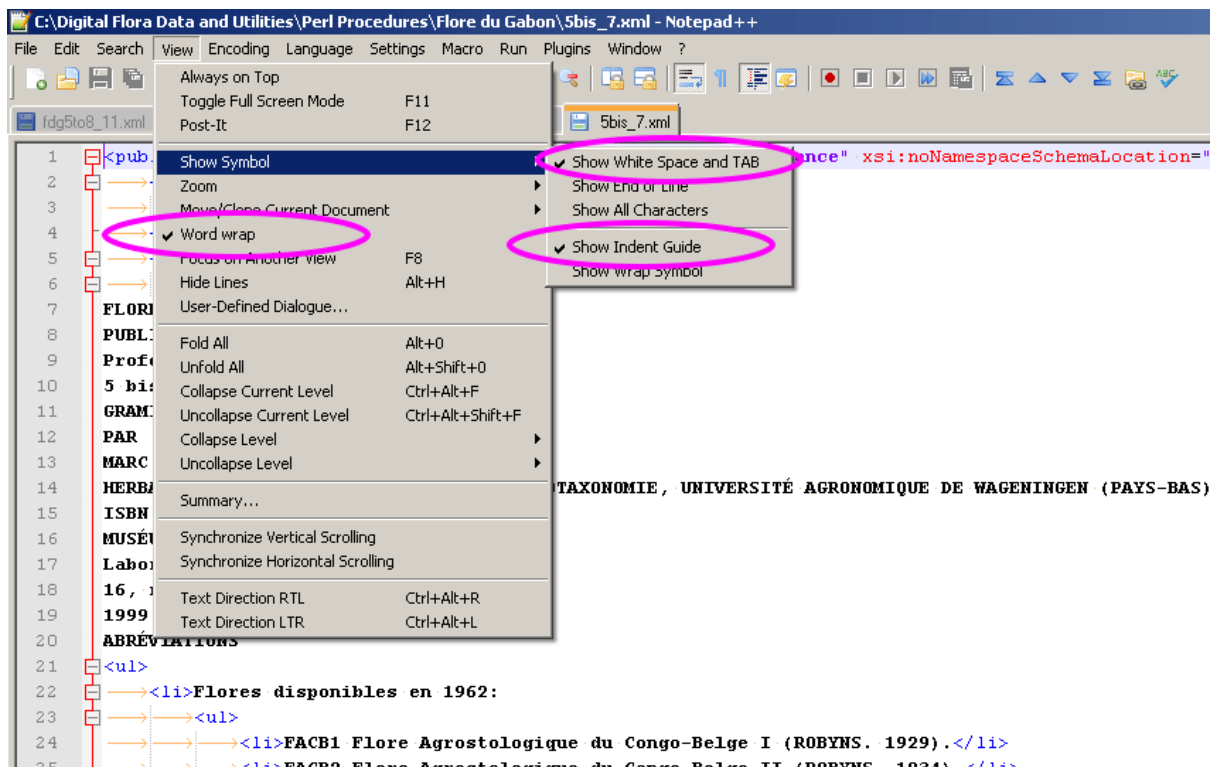


Figure 14: Helpful options in Notepad++.

Further actions

Once all of the scripts have been run, the next step is proofreading the final output file and making the required corrections. See **[XML proofreading.doc](#)**.

Appendix I: FlorML Script running orders

These are the preferred running orders for the scripts written for Flora Malesiana, Flore du Gabon, and Flora of the Guianas.

In all cases, for additional information refer to:

- **script creation.doc** for details on modifying the scripts and why a particular order of scripts was chosen.
- Appendix I in **text preparation.doc** for details on the purpose of the clean-up scripts.
- **FlorML reference.doc** for details on the use of the various XML elements.

If you are not working on these three floras, you will have to create your own scripts and script order. The processes required for this are described in **script creation.doc**.

Running order for Flora Malesiana

The following list shows the running order for the Perl scripts for Flora Malesiana. You need to use the scripts for Flora Malesiana (available at <https://github.com/thoha/FlorML>, folder Example_Scripts/Flora_Malesiana_scripts).

- 1) **filecleaner.pl** (clean-up)
- 2) **ocrfix.pl** (fixing OCR-errors)

At this point, insert empty lines between taxa.

- 3) **pubtags.pl** (first and last publication and taxon tags, metadata base tags)
- 4) **taxontags.pl** (other taxon tags)
- 5) **keytags.pl** (keys)

At this point, manually insert tags for tables, lists and line breaks (see **FlorML reference.doc**) and remove newlines after line breaks in descriptions and distributions.

- 6) **footnotes.pl** (footnotes)
- 7) **featbasics.pl** (features)

(If many descriptions are missed by the previous script, consider adding the basic description mark-up before running the next script)

- 8) **figures.pl** (figures)
- 9) **nombasics.pl** (basic nomenclature tags)
- 10) **characters.pl** (character atomisation in descriptions)
- 11) **nomatomizer.pl** (nomenclature atomisation)
- 12) **atomizer.pl** (new script for all remaining atomisation, incl. distributions)

13) **entities.pl** (special symbols)

Now proofread the whole document, make corrections, and ensure it validates with the latest version of FlorML.

(Compared to old mark-up scripts, several FM scripts have been merged into each other; order improved for better efficiency)

Flora Malesiana bug fixing scripts

There exist two sets of scripts to fix two separate issues with the mark-up of Flora Malesiana:

Wrong mark-up of series and volume numbers

The first set of bug fixing scripts fixes a mark-up issue where the literature and reference atomisation failed to properly recognise series and volume numbers. In this case, when Roman numerals were used for the series, followed by a volume number in regular numerals, these were accidentally marked up as volume and issue numbers, respectively. Run the following two scripts in order to fix this:

- 1) **serfixA.pl** (run on file with errors, output to new file)
- 2) **serfixB.pl** (run on output file of serfixA.pl, output to new file)

The resulting file may have some <refPart>-tags that were not put back on a separate line. If you care about that, fix it manually.

Wrong mark-up of older misidentified names

The second set of bug fixing scripts fixes a mark-up issue caused not by the nomenclature atomisation script itself, but by a divergent text format used for misidentified names in the earlier Flora Malesiana volumes. Instead of using the “auct. non AUTHOR A: AUTHOR B”-format used in later volumes, these volumes use “(non AUTHOR A) AUTHOR B”. This causes a mismatch during nomenclature atomisation. These scripts convert the earlier format to the later format in XML files that are fully marked up and proofread:

- 1) **parautfixA.pl** (run on file with errors, output to new file)
- 2) **parautfixB.pl** (run on output file of parautfixA.pl, output to new file)

Running order for Flore du Gabon

The following list shows the running order for the Perl scripts for Flore du Gabon. You need to use the scripts for Flore du Gabon (available at <https://github.com/thoha/FlorML>, folder Example_Scripts/Flore_du_Gabon_scripts).

- 1) **nettoyeur.pl** (clean-up)
- 2) **reparocr.pl** (fixing OCR-errors)

At this point, insert empty lines between taxa.

- 3) **pubtags_fr.pl** (first and last publication and taxon tags, metadata base tags)
- 4) **taxontags_fr.pl** (other taxon tags)
- 5) **clestags.pl** (keys)
- 6) **planches.pl** (figures)
- 7) **notesbaspages.pl** (footnotes)

At this point, manually insert tags for tables, lists and line breaks (see **FlorML reference.doc**) and remove newlines after line breaks in descriptions and distributions.

- 8) **featuresfr.pl** (basic feature tags)

(If many descriptions are missed by the previous script, consider adding the basic description mark-up before running the next script)

- 9) **nomenclaturefr.pl** (nomenclature, types)

At this point, manually insert other feature tags including string tags and headings (see **FlorML reference.doc**), etc.

- 10) **atomiseur.pl** (atomisation, step 1: nomenclature, types, distribution data, vernacular names, descriptions, etc.)
- 11) **annotationsfr.pl** (annotations)
- 12) **entities.pl** (special symbols)

Now proofread the whole document, make corrections, and ensure it validates with the latest version of FlorML.

Running order for Flora of the Guianas

The following list shows the running order for the Perl scripts for Flora of the Guianas. You need to use the scripts for Flora of the Guianas (available at <https://github.com/thoha/FlorML>, folder Example_Scripts/Flora_of_the_Guianas_scripts).

Important!!! Due to some specific contents in Flora of the Guianas, some additional preparation steps are required before you run any of the scripts. These steps are described in Appendix II in **text preparation.doc**.

- 1) **filecleaner.pl** (clean-up)
- 2) **ocrfix.pl** (fixing OCR errors)

At this point, insert empty lines between taxa.

- 3) **pubtags.pl** (first and last publication and taxon tags, metadata base tags)
- 4) **taxontags.pl** (other taxon tags)
- 5) **keytags.pl** (keys)
- 6) **figures.pl** (figures + gatherings listed in the figures' text)
- 7) **footnotes.pl** (footnotes)

At this point, manually insert tags for tables, lists and line breaks (see FlorML reference.doc) and remove newlines after line breaks in descriptions and distributions.

- 3) **featbasic.pl** (basic feature tags).

At this point, manually insert other feature tags including string tags and headings (see **FlorML reference.doc**), etc. Furthermore, if many descriptions are missed by the previous script, consider adding the basic description mark-up before running the next script.

- 4) **nombasics.pl** (basic nomenclature tags).
- 5) **atomizer.pl** (first atomisation step).
- 6) **entities.pl** (entities).

Now proofread the whole document, make corrections, and ensure it validates with the latest version of FlorML.

- 7) **atom2.pl** (second atomisation step)
- 8) **namefinder.pl** (remaining names).

Now proofread the whole document a second time, make corrections, and ensure it validates with the latest version of FlorML.

- 9) Optional: **recodechar.pl** (for adding additional contextual information to character trait data; no proofreading required).