Project Organization

Overview

Teaching: 15 min
Exercises: 15 min
Questions

- How can I organize my file system for a new bioinformatics project?
- How can I document my work?

Objectives

- Create a file system for a bioinformatics project.
- Explain what types of files should go in your `docs` , `data` , and `results` directories.
- Use the `history` command and a text editor like `nano` to document your work on your project.

Getting your project started

Project organization is one of the most important parts of a sequencing project, and yet is often overlooked amidst the excitement of getting a first look at new data. Of course, while it's best to get yourself organized before you even begin your analyses, it's never too late to start, either.

You should approach your sequencing project similarly to how you do a biological experiment and this ideally begins with experimental design. We're going to assume that you've already designed a beautiful sequencing experiment to address your biological question, collected appropriate samples, and that you have enough statistical power to answer the questions you're interested in asking. These steps are all incredibly important, but beyond the scope of our course. For all of those steps (collecting specimens, extracting DNA, prepping your samples) you've likely kept a lab notebook that details how and why you did each step. However, the process of documentation doesn't stop at the sequencer!

Genomics projects can quickly accumulate hundreds of files across tens of folders. Every computational analysis you perform over the course of your project is going to create many files, which can especially become a problem when you'll inevitably want to run some of those analyses again. For instance, you might have made significant headway into your project, but then have to remember the PCR conditions you used to create your sequencing library months prior.

Other questions might arise along the way:

- What were your best alignment results?
- Which folder were they in: Analysis1, AnalysisRedone, or AnalysisRedone2?
- Which quality cutoff did you use?
- What version of a given program did you implement your analysis in?

Good documentation is key to avoiding this issue, and luckily enough, recording your computational experiments is even easier than recording lab data. Copy/Paste will become your best friend, sensible file names will make your analysis understandable by you and your collaborators, and writing the methods section for your next paper will be easy! Remember that in any given project of yours, it's worthwhile to consider a future version of yourself as an entirely separate collaborator. The better your documenation is, the more this 'collaborator' will feel indebted to you!

With this in mind, let's have a look at the best practices for documenting your genomics project. Your future self will thank you.

In this exercise we will setup a file system for the project we will be working on during this workshop.

We will start by creating a directory that we can use for the rest of the workshop. First navigate to your home directory. Then confirm that you are in the correct directory using the `pwd` command.

Bash

```
$ cd
$ pwd
```

You should see the output:

Output

```
/home/dcuser
```

Tip

If you aren't in your home directory, the easiest way to get there is to enter the command `cd` , which always returns you to home.

Exercise

Use the `mkdir` command to make the following directories:

- `dc_workshop`
- `dc_workshop/docs`
- `dc_workshop/data`
- `dc_workshop/results`

Solution

Bash

```
$ mkdir dc_workshop
$ mkdir dc_workshop/docs
$ mkdir dc_workshop/data
$ mkdir dc_workshop/results
```

Use `ls -R` to verify that you have created these directories. The `-R` option for `ls` stands for recursive. This option causes `ls` to return the contents of each subdirectory within the directory iteratively.

Bash

```
$ ls -R dc_workshop
```

You should see the following output:

Output

```
dc_workshop/:
data  docs  results

dc_workshop/data:

dc_workshop/docs:

dc_workshop/results:
```

Organizing your files

Before beginning any analysis, it's important to save a copy of your raw data. The raw data should never be changed. Regardless of how sure you are that you want to carry out a particular data cleaning step, there's always the chance that you'll change your mind later or that there will be an error in carrying out the data cleaning and you'll need to go back a step in the process. Having a raw copy of your data that you never modify guarantees that you will always be able to start over if something goes wrong with your analysis. When starting any analysis, you can make a copy of your raw data file and do your manipulations on that file, rather than the raw version. We learned in a previous episode (http://www.datacarpentry.org/shell-genomics/03-working-with-files/#file-permissions) how to prevent overwriting our raw data files by setting restrictive file permissions.

You can store any results that are generated from your analysis in the `results` folder. This guarantees that you won't confuse results file and data files in six months or two years when you are looking back through your files in preparation for publishing your study.

The `docs` folder is the place to store any written analysis of your results, notes about how your analyses were carried out, and documents related to your eventual publication.

Documenting your activity on the project

When carrying out wet-lab analyses, most scientists work from a written protocol and keep a hard copy of written notes in their lab notebook, including any things they did differently from the written protocol. This detailed record-keeping process is just as important when doing computational analyses. Luckily, it's even easier to record the steps you've carried out computational than it is when working at the bench.

The `history` command is a convenient way to document all the commands you have used while analyzing and manipulating your project files. Let's document the work we have done on our project so far.

View the commands that you have used so far during this session using `history`:

Bash

```
$ history
```

The history likely contains many more commands than you have used for the current project. Let's view the last several commands that focus on just what we need for this project.

View the last n lines of your history (where n = approximately the last few lines you think relevant). For our example, we will use the last 7:

Bash

```
$ history | tail -n 7
```

Exercise

Using your knowledge of the shell, use the append redirect `>>` to create a file called `dc_workshop_log_XXXX_XX_XX.sh` (Use the four-digit year, two-digit month, and two digit day, e.g. `dc_workshop_log_2017_10_27.sh`)

Solution

Bash

```
$ history | tail -n 8 >> dc_workshop_log_2017_10_27.sh
```

Note we used the last 7 lines as an example, the number of lines may vary.

You may have noticed that your history contains the `history` command itself. To remove this redundancy from our log, let's use the `nano` text editor to fix the file:

Bash

```
$ nano dc_workshop_log_2017_10_27.sh
```

(Remember to replace the `2017_10_27` with your workshop date.)

From the `nano` screen, you can use your cursor to navigate, type, and delete any redundant lines.

Navigating in Nano

Although `nano` is useful, it can be frustrating to edit documents, as you can't use your mouse to navigate to the part of the document you would like to edit. Here are some useful keyboard shortcuts for moving around within a text document in `nano`. You can find more information by typing Ctrl-G within `nano`.

key	action
Ctrl-Space	to move forward one word
Alt-Space	to move back one word
Ctrl-A	to move to the beginning of the current line
Ctrl-E	to move to the end of the current line
Ctrl-W	to search

Add a date line and comment to the line where you have created the directory, for example:

Bash

```
# 2017_10_27
# Created sample directories for the Data Carpentry workshop
```

`bash` treats the `#` character as a comment character. Any text on a line after a `#` is ignored by bash when evaluating the text as code.

Next, remove any lines of the history that are not relevant by navigating to those lines and using your delete key. Save your file and close `nano`.

Your file should look something like this:

Output

```
# 2017_10_27
# Created sample directories for the Data Carpentry workshop

mkdir dc_workshop
mkdir dc_workshop/docs
mkdir dc_workshop/data
mkdir dc_workshop/results
```

If you keep this file up to date, you can use it to re-do your work on your project if something happens to your results files. To demonstrate how this works, first delete your `dc_workshop` directory and all of its subdirectories. Look at your directory contents to verify the directory is gone.

Bash

```
$ rm -r dc_workshop
$ ls
```

Output

```
shell_data      dc_workshop_log_2017_10_27.sh
```

Then run your workshop log file as a bash script. You should see the `dc_workshop` directory and all of its subdirectories reappear.

Bash

```
$ bash dc_workshop_log_2017_10_27.sh
$ ls
```

Output

```
shell_data      dc_workshop dc_workshop_log_2017_10_27.sh
```

It's important that we keep our workshop log file outside of our `dc_workshop` directory if we want to use it to recreate our work. It's also important for us to keep it up to date by regularly updating with the commands that we used to generate our results files.

Congratulations! You've finished your introduction to using the shell for genomics projects. You now know how to navigate your file system, create, copy, move, and remove files and directories, and automate repetitive tasks using scripts and wildcards. With this solid foundation, you're ready to move on to apply all of these new skills to carrying out more sophisticated bioinformatics analysis work. Don't worry if everything doesn't feel perfectly comfortable yet. We're going to have many more opportunities for practice as we move forward on our bioinformatics journey!

References

A Quick Guide to Organizing Computational Biology Projects (http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000424)

Key Points

- Spend the time to organize your file system when you start a new project. Your future self will thank you!
- Always save a write-protected copy of your raw data.

(../05-writing-scripts/index.html)

(../)