Computational Biology

A Practical Introduction to BioData Processing and Analysis with Linux, MySQL, and R

Bearbeitet von
Röbbe Wünschiers

1. Auflage 2013. Buch. xxix, 449 S. Hardcover
ISBN 978 3 642 34748 1
Format (B x L): 15,5 x 23,5 cm
Gewicht: 877 g

Chapter 2
Content of This Book

This book aims at the total beginner. However, if you know something about computers but not about programming, the book will still be useful for you. After introducing the basics of how to work in the Linux environment, some great tools will be presented. Among these are the stream line editor `sed`, the script-oriented programming languages `awk` and `perl`, the data analysis and visualization tool R, and the relational database system MySQL. These utilities are extremely helpful when it comes to formatting and analyzing data files. After you have worked through all the chapters, you can use this book as a reference. The learning approach is absolutely practically oriented. Thus, you are invited to run all examples, printed in so-called Terminals, on your own! *If you face any problems: contact me!* Of course, I cannot help you if your non-unix-like-operating-system driven computer crashes continuously. However, if things connected to this book confuse you—or you even find errors—please let me know (Email: rw@biowasserstoff.de). Further information about this book, including lists with internet links and known errors, can be found at my homepage (http://www.hs-mittweida.de/wuenschi). You are very much welcome to supply me with good ideas for examples!

2.1 The Main Chapters

This book contains several chapters that focus on one particular concept, e.g. a programming language. Though, there is a progression—chapters usually depend on previous chapters. However, if you are already a command line geek, you might skip early chapters.

2.1.1 Linux

Linux is a multi-user multi-task operating system, originally based on *Minix*, which is an operating system similar to Unix. Linux was initially developed by Linus Torvalds

R. Wúnschiers, *Computational Biology*, DOI: 10.1007/978-3-642-34749-8_2,

in 1991. It is an *open source* operating system. This means everybody who has programming knowledge can modify and improve the system; but it also means that everybody can download and install it. This is a main reason to choose Linux: you need invest no money except for the book itself. Still, the content of this book is valid for any Unix-like operating system like Mac OS X or CygWin, the free Unix emulator for Windows.

2.1.2 Shell Programming

The shell, also called terminal or console, will be our playground. Everything we do in this book is done in the shell. The shell can be seen as a command interpreter: we enter a command and the shell takes care of its execution; but we can also combine a number of commands and programs, including programming structures like decisions, in order to generate new functionality. Typical shell programs handle files and directories rather than file contents. A common task would be to convert all file extensions from *.txt* to *.seq*, make specific files executable or archive all recently changed files. Shell programming resembles DOS's programming language for *batch files*.

2.1.3 Sed

No, this is not the about the Socialist Unity Party of Germany (German: Sozialistische Einheitspartei Deutschlands, SED) that was governing the German Democratic Republic from October 1949 until March 1990. `sed` (**s**tream **ed**itor) is used to perform basic text editing on an input text file (or data stream) and was written by Lee E. McMahon in 1973. `sed` does not allow for any interactions.
Figure 2.1 shows a sketch for an example of what `sed` can do. Here, the stop codon of a DNA sequence is replaced by the text "!STOP!". `sed` is well suited to perform small formatting tasks like converting RNA to DNA, commas to points, tabs to semicolons and the like.

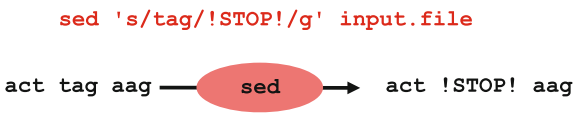

Fig. 2.1 Stream EDitor. What does Sed do?

2.1.4 AWK

AWK is a programming language for handling common data manipulation tasks with only a few lines of code. It was initially developed by Alfred V. **A**ho, Peter J. **W**einberger and Brian W. **K**ernighan in 1977. This is where the name comes from. AWK is really a great tool when it comes to analyzing the content of data files. With AWK you can perform calculations, draw decisions, read and write multiple files. What is best is that AWK can be extended with your own designed functions. A typical task would be to fuse the content of files having one common field. Another typical task would be to extract data matching certain criteria. AWK forms the kernel of this book. After you have finished the chapter on AWK you should be able to (a) program basically anything you need and (b) learn any other programming language. The example shown in Fig. 2.2 shows one basic function of AWK. All enzyme names in the file *enzymes.file* are printed, if the corresponding Km value (do you remember enzyme kinetics and Michaelis-Menten?) is smaller than 0.4.

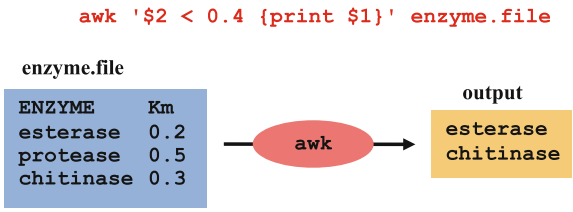

Fig. 2.2 AWK. What does AWK do?

2.1.5 Perl

Practical **e**xtraction and **r**eport **l**anguage (Perl) arose from a project started in 1987 by Larry Wall. It combines some of the best features of the programming language C, Sed, AWK and shell programming and was optimized for scanning arbitrary text files, extracting information from those text files and printing reports based on that information. Perl is intended to be practical rather than being a beautiful language. It offers you everything a programming language can offer. Perl is often used to program web-based applications (known as CGI scripts), it provides database connectivity and there is even a Bioperl project which provides many tools biologists need. This book can introduce you to only the very basics of Perl. Still, you will get to the point where you learn how to write your own modules and generate small database files.

2.1.6 MySQL

When you happen to have several, possibly cross-linked, Excel tables, then you should consider to save them in a MySQL database. Imagine you have one table (or tab-delimited file) containing gene expression data. Another table contains gene annotations (Fig. 2.3).
Thus, there is a relationship between data in these tables. The power of MySQL (and other relational database systems) is to query over such relations. Furthermore, MySQL is a good storage place for such data because everything is available in one database. Many programming languages and software packages—e.g. R—can connect to MySQL.

Table: *annotation*

gene	function	metabolism
alr2938	iron superoxide dismutase	Detoxification
alr4392	nitrogen-responsive regulator	Nitrogen assimilation
alr4851	preprotein translocase subunit	Protein and peptide secretion
alr3395	adenylosuccinate lyase	Purine biosynthesis
alr1207	uridylate kinase	Pyrimidine biosynthesis
alr5000	CTP synthetase	Pyrimidine biosynthesis
all3556	succinate-dehydrogenase	TCA cycle

Table: *expression*

gene	expr_level
alr1207	8303
alr2938	10323
alr3395	1432
all3556	8043
alr4392	729
alr4851	633
alr5000	5732

```
mysql> SELECT a.gene, a.function, e.expr_value
    -> FROM annotation as a, expression as e
    -> WHERE a.gene = e.gene AND
    -> a.metabolism REGEXP
    -> "(Purine|Pyrimidine) biosynthesis" AND
    -> e.expr_value < 5000;
+---------+-------------------------+------------+
| gene    | function                | expr_value |
+---------+-------------------------+------------+
| alr3395 | adenylosuccinate lyase |       1432 |
+---------+-------------------------+------------+
1 row in set (0.00 sec)

mysql>
```

Fig. 2.3 MySQL. The query shown at the bottom right displays gene names and gene functions for all genes, which are involved in either purine or pyrimidine metabolism and which have an expression value less than 5000

2.1.7 R

There are a broad range of software tools available to perform data analysis and visualization. R is a completely free software that emulates S-Plus. S-Plus in turn was initially developed by AT&T's Bell Laboratories to provide a software tool for professional statisticians who wanted to combine state-of-the-art graphics with powerful model-fitting capability.
R is a very well established platform for scientists in general and computational biologists in particular. Besides from being a programming environment for statisti-

cal computing, R is also a data visualization tool. Several subject specific packages are available to analyze and visualize experimental data, e.g. for evolutionary biology, the evaluation of biochemical assays, nucleotide and amino acid sequence analysis, microarray data interpretation and more. Bioconductor is the name of an international project that brings developments from various research teams together and that provides tools for the analysis and comprehension of high-throughput genomic data.
The introduction given in this book shall set you on the right track to develop more and more sophisticated skills in using and combing available packages and apply them to your own scientific problems.

2.1.8 Worked Examples

This might be the most important part of this book. In Part VI on p. 343 I present four chapters that can be used for teaching and training. All are based on real-world problems and come with a detailed instruction of how to solve the problem. All exercises include data processing and visualization of some sort.
Most programmers would agree that one learns a programming language by using and adapting programs of others. This is learning by mimicking. You should go the same way with the worked examples. Start by following the instructions—then leave the path and start playing with both the data and data processing. Maybe you develop nice modifications to the examples and like to send them to me—I would be very happy. Last but not least, the worked examples shall give you a glimpse of where you could employ command line data processing in your own projects.

2.2 Prerequisites

In order to perform the exercises shown in this book—and there are lot—you need to have access to a computer running either Linux, Unix or Mac OS X (the newest Apple operating system) or the free Windows Unix emulator CygWin (see below). As you will learn soon, these systems are very similar. Thus, all the things we are going to learn will work on all Linux, Unix and Mac OS X computers. On a normal installation all required programs should be installed. Otherwise, contact the system administrator.
My personal recommendation is to install Linux as a virtual machine. This operating system independent approach is explained in Sect. 4.1.2 on p. 40 (see also Sect. 3.8.1 on p. 30).
Alternatively, you can install the free *Cygwin* Unix emulator on a computer running the Microsoft Windows operating system (from Win95 upwards, excluding WinCE) (see Sect. 3.8.2 on p. 30) or to start with *Knoppix* Linux (see Sect. 3.9 on p. 31).

Knoppix runs from a CD-ROM and requires no installation on the hard disk drive. Neither your operating system nor the data on your computer will be touched.

2.3 Conventions

What you see on your computer screen is written in `typewriter` style and boxed. I will refer to this as the *Terminal*. The data you have to enter are given behind the $ character. Key labels are written in a box. For example, the key labelled "Enter" would be written as (Enter). Commands that appear in the text are written in `typewriter`, too. When necessary, space characters are symbolized by "␣" in the text. Thus, "␣␣␣" means that you have to type three consecutive spaces. In the following example you would type `date` as input and get the current date as output. Thereafter, from lines 3-5 I indicate how I mark commands or output that spans several lines.

Terminal 1: The Command *date*

```
1 $ date
2 Thu Feb 13 18:53:26 CET 2003
3 $ # this is a very long comment line that does nothing but indicating how I      +
4   highlight commands or output that spans two or more lines; NOTE the plus signs +
5   at the end of the lines
6 $
```

In most cases you will find some text behind the terminal which describes the terminal content: in Terminal 1, line 1, we check for the current date and time.
Boxes labelled "Program" contain script files or programs. These have to be saved in a file as indicated in the first or second program line: `# save as hello.sh`. You will find the program under the same name on the accompanying website. As terminals, programs are numbered.

Program 1: *hello.sh* - Our First Shell Script

```
1 #!/bin/bash
2 # save as hello.sh
3 # This is a comment
4 echo "Hello World"
```

Finally, there are "Files", which usually contain text files that are processed. At the end of most chapters you will find exercises. These are numbered, too. The solution can be found in Sect. A.7 on p. 431.

2.4 Additional Resources

This book is accompained by a website (http://www.staff.hs-mittweida.de/~wuenschi/doku.php?id=rwbook2) and a blog (Fig. 2.4). Suggestions and comments are highly welcome.

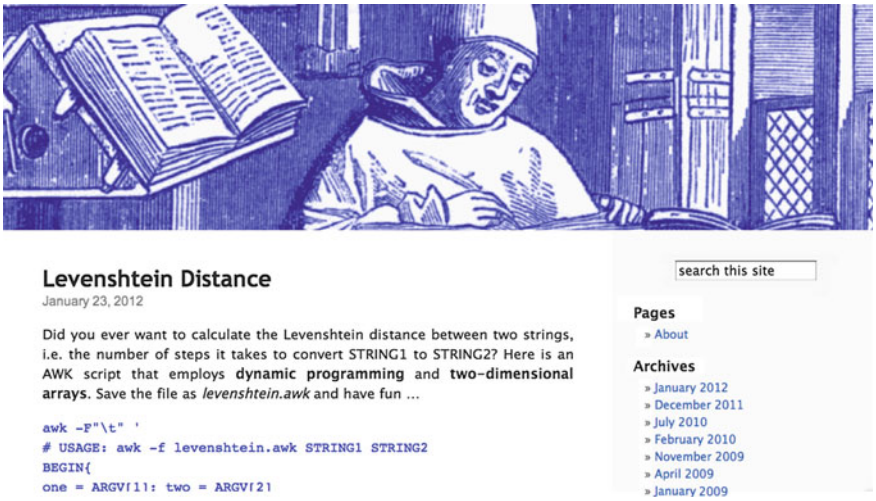

Fig. 2.4 Blog about AWK. Take a look or participate at my blog at http://www.awkologist.com