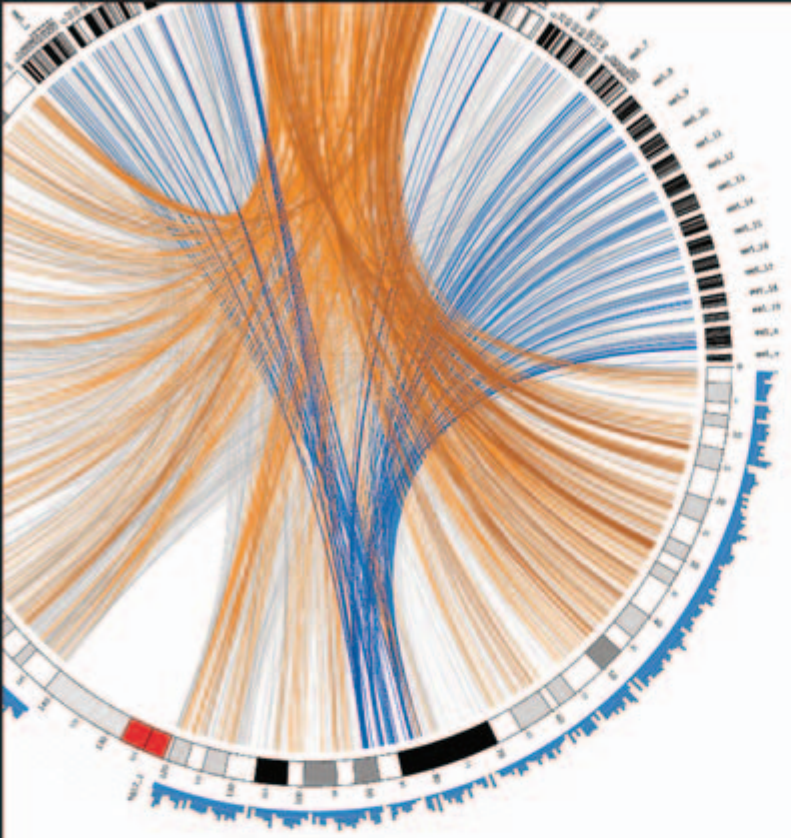


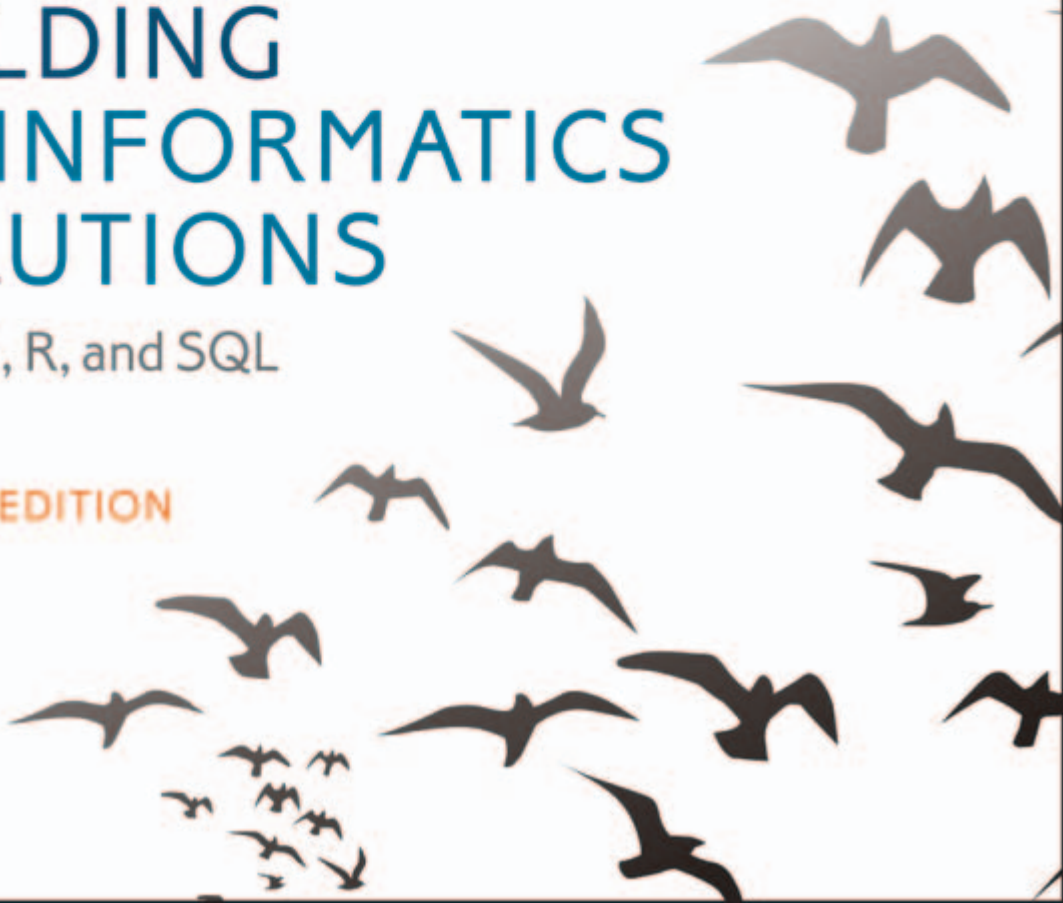
OXFORD

CONRAD **BESSANT**
DARREN **OAKLEY**
IAN **SHADFORTH**

BUILDING BIOINFORMATICS SOLUTIONS

with Perl, R, and SQL

SECOND EDITION



Building Bioinformatics Solutions

Building Bioinformatics Solutions

with Perl, R, and SQL

Conrad Bessant
Darren Oakley
Ian Shadforth

Second Edition

OXFORD
UNIVERSITY PRESS

Building Bioinformatics Solutions. Second Edition. Conrad Bessant, Darren Oakley and Ian Shadforth.

OXFORD
UNIVERSITY PRESS

Great Clarendon Street, Oxford OX2 6DP,
United Kingdom

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide. Oxford is a registered trade mark of
Oxford University Press in the UK and in certain other countries

First Edition published in 2009
Second Edition published in 2014

Impression: 1

Published in the United States of America by Oxford University Press
198 Madison Avenue, New York, NY 10016, United States of America

British Library Cataloguing in Publication Data

Data available

Library of Congress Control Number: 2013943290

ISBN 978–0–19–965855–8 (hbk.)
ISBN 978–0–19–965856–5 (pbk.)

Printed in Great Britain by
Clays Ltd, St Ives plc

Acknowledgements

We would like to take this opportunity to acknowledge the army of very smart people who have contributed to the open source software and publicly available databases used herein. Without these tools, bioinformatics would never have advanced to the state it is in today and this book would probably not exist.

We would also like to acknowledge our colleagues who have contributed to this book by providing advice, ideas, and material. Particular thanks are due to Michael Cauchi, Jenny Cham, Elena Chatzimichali, Jun Fan, Dan Klose, Fady Mohareb, and Will Stott. Thanks also to the team at Oxford University Press who have supported *Building Bioinformatics Solutions* since the beginning and have brought this second edition to the shelves.

Finally, we are indebted to our friends and family who once again provided their invaluable support as we inevitably spent longer than we should have done locked away in our offices.

Preface to the Second Edition

Bioinformatics is a fast moving subject, so while updating this book for the second edition it has been interesting to consider which aspects of the field have changed and which have remained the same. As the commissioning of a second edition indicates, bioinformatics skills continue to be in high demand, as the acquisition of complex multivariate data sets becomes the norm across many areas of biology. Although many new tools have emerged since the first edition of this book was published, and many existing tools have been further developed, there is still a need for the development of new tools, and the creation of bespoke software that brings existing tools together, increasingly across the different 'omics' disciplines.

So the need to build bioinformatics solutions remains. The general approach to building these solutions also remains the same, typically involving a database element, programming, some quantitative analysis, and possibly the creation of a web front end. We have therefore kept this structure within the book. The core tools around which we wrote the first edition—MySQL, Perl, and R—are all still popular within the bioinformatics community. The choice of programming language was a tough one, with Python and Ruby both gaining ground in the bioinformatics community. Ultimately, however, we decided that Perl was still the right choice for this book as it remains the single most widely used language in bioinformatics, and has the benefit of a thriving ecosystem that gives a head start when developing most types of bioinformatics applications. However, in recognition of the increased diversity of programming languages used in the community, we have added new material explaining how to get started in Python, Ruby, and Java to help you experiment with these languages so you can make up your own mind which best suits your needs. Having said that, we should note that all these languages are underpinned by the same fundamental programming concepts, so time spent learning Perl will give you a head start in any of the other languages.

Something else we have witnessed over recent years is a gradual increase in the size and complexity of bioinformatics projects. More complex projects call for more systematic coding practices, such as those proposed by the discipline of software engineering. We have added a new chapter dedicated to this subject, covering version control, documentation, user-centred design, and unit testing. These techniques can help streamline the development of your software—especially

when working as part of a team—and make it more reliable, easier to use, and easier to maintain and extend.

The technical area that has changed the most since the first edition is web development. Perl CGI's dominant position in bioinformatics web development has been usurped by web frameworks that are far more powerful while, at the same, make the development and debugging of web applications much easier. At the same time, the arrival of HTML5—the most significant update to the core web standard for over a decade—together with increasingly sophisticated JavaScript libraries such as jQuery, has ushered in a new era of more engaging web applications. Chapter 5 has been substantially revised to take account of these latest developments.

We sincerely hope that this book provides the overview of bioinformatics software development that you are looking for, and that it gives you the knowledge and confidence to go ahead and build your own bioinformatics solutions. We very much welcome your comments and questions—the best way to find us is via `www.bixsolutions.net`.

Conrad, Darren, and Ian
November 2013

Contents

CHAPTER 1

Introduction

1.1 From data to knowledge: the aim of bioinformatics

The term 'bioinformatics' can mean different things to different people. For the purpose of this book, we adopt the broad definition that bioinformatics is the process of extracting novel biological knowledge from bioanalytical data. Such knowledge has immense value because it may be used to better understand biology, to combat disease, and to mitigate environmental catastrophes, but what we actually have is lots of data—genome sequences, protein structures, metabolomic profiles, and more.

Bioinformatics bridges this gap between data and knowledge. It is neither tied to a particular type of experimental data nor to a particular biological application. Indeed, the data may come from any of today's plethora of bioanalytical methods, such as high throughput sequencing, nuclear magnetic resonance, or mass spectrometry, and the knowledge sought can be as varied as the identity of a new disease biomarker, a phylogenetic tree, or a system-wide understanding of a particular biological process. Similarly, the tasks involved in bioinformatics range from simply organizing data for future use, through to sophisticated analysis, visualization, and sharing of that data and the results derived from it. Bioinformatics is therefore a truly interdisciplinary subject, requiring an understanding of at least some biology, analytical science, mathematics, statistics, and information technology.

There are, of course, many generally available bioinformatics tools that can be used to analyse data with a view to extracting new knowledge, and many of these tools are free of charge. However, due to the high complexity and bespoke nature of biological data sets, it is often necessary to produce in-house software to organize, analyse, and visualize data. In this book, we introduce some of the main tools and general approaches employed to produce such software.

The book is primarily aimed at readers with a background in the life sciences who have some bioinformatics knowledge, but little or no experience in the development of software and databases. A typical reader will most likely have used online databases such as GenBank or Ensembl (if not, these are introduced briefly later in this chapter), and experienced the power of tools such as BLAST (Basic Local Alignment Search Tool) or analysis platforms such as Galaxy, and now want to take the next step and develop their own bioinformatics tools, either for their

Building Bioinformatics Solutions. Second Edition. Conrad Bessant, Darren Oakley and Ian Shadforth.

own use or for sharing with their collaborators or with the life science community at large. Our aim in writing this book was to fill the gap between texts that introduce the field of bioinformatics, such as *Introduction to Bioinformatics* (Lesk, 2008), and software development books, such as those published by O'Reilly (e.g. Laurie & Laurie, 2013; Christiansen *et al.*, 2012; Tahaghoghi & Williams, 2009). We therefore cover computing material from a fairly elementary level (for example, Appendix A explains how to use a command-line interface), while the biological background necessary to understand the applications is assumed.

1.2 Using this book

The book has been written with the intention of being read linearly from beginning to end, with a structure that generally mirrors a typical bioinformatics project. First, it is necessary to assemble the data, be it from a laboratory or from online resources, typically into a structured database using a relational database management system (RDBMS) such as MySQL (Chapter 2). Then, due to the nature of bioinformatics data sets, some programming is usually required to automate various data manipulation tasks—Perl (covered in Chapter 3) is the tool of choice for this among bioinformaticians. Often, advanced numerical data analysis is then required to extract useful information from the data gathered, which is where R comes in (Chapter 4). In Chapter 5, we bring everything together and the real power of integrating Perl, R, and MySQL becomes apparent as we show how to combine these ingredients with HTML5 and web frameworks to make complex bioinformatics tools available via the web. Finally, Chapter 6 introduces some good software engineering practice—essential when working on larger multi-developer projects—and introduces alternative programming languages that are purported to have some technical advantages over Perl. Having said all that, Chapters 2, 3, and 4 have been written to be reasonably standalone, so if you have an urgent need to go straight to a particular chapter, that is possible. Where necessary, cross-references are provided to help locate explanations of concepts that appeared earlier in the book. Similarly, if you are already experienced in one of the three main tools covered, you should be able to skip that chapter without any problems.

One thing that this book is definitely not is a reference manual. The index should help you find a relevant passage in the book when you need it, but don't expect to see every single Perl function, R package, or Apache configuration option listed there. A simple reason for this is that there just is not enough space, but more importantly these tools have excellent online documentation and we find such documentation to be a much more convenient format for reference materials because it is easy to search and frequently updated. This book is a companion to those reference materials, and a starting point in terms of knowing where the reference materials are and how to use them efficiently.

1.2.1 About the coverage of this book

Deciding what to include and what not to include in this book was not easy. Whole books have been written about MySQL, Perl, and R, not to mention all

the various types of bioinformatics applications to which they can be applied. It is, therefore, impossible to claim that this book provides exhaustive coverage of all the subjects covered. Selecting the topics to cover was very much like writing a tourist guide to a particular country—it is not feasible to cover everything of interest, so information needs to be carefully selected so that it helps newcomers orientate themselves, outlines the practicalities of survival, covers a few highlights that give a flavour of what is possible, and tells you where you can go to find out more. This is the approach we have taken in this book.

1.2.2 Choice of tools

The toolkit for building bioinformatics solutions that we put forward in this book is loosely based on the so-called LAMP toolkit, which has been widely used by developers in all sorts of domains, not just bioinformatics, for many years. LAMP is an acronym for four popular open source software packages: Linux (the operating system), Apache (web server), MySQL (database), and Perl[1] (for programming). Together, these packages make a powerful combination for gathering, storing, and serving up data over the Internet. To this mix we add R, which brings with it the sophisticated data analysis and visualization capabilities frequently needed in bioinformatics applications. A key factor in the appeal of this suite of tools is that they are open source and freely available. To be frank, the technical benefit of these tools being open source is minimal for the typical user. Although one is able to view and edit the source code of the tools one is using, the chances are that one will never need to do so and, if one did, the complexity of these packages is such that making worthwhile modifications would be very time consuming. Even the fact that the packages can be obtained free of charge is of little direct relevance if one has a software budget. What really makes the use of open source tools appealing is that they are ubiquitous and this widespread uptake has two very important consequences. First, there are a lot of people around who know how to use them, so getting help should not be a problem. Second, if someone wants to produce a piece of software or an add-on, they are most likely to do it with these tools to allow maximum exposure. This latter benefit is really the most important, as it means that there is, for example, a constant stream of add-on modules for Perl and packages for R that perform common bioinformatics tasks. We refer to the collection of community activities and third-party add-ons for a particular tool as the tool's *ecosystem*. Choosing to use a development tool that has a well-established and dynamic ecosystem saves a huge amount of development effort and allows one to concentrate on the science, which is, after all, the main priority in bioinformatics.

1.2.3 Choice of operating system

Throughout this book we have worked to ensure that everything is operating system independent as far as possible. All examples have been tested on Windows

1 Depending on who you ask, the P might actually stand for a similar language called PHP, or even just programming in general. It doesn't really matter.

8, Mac OS 10.8.3, and Ubuntu Linux 12.04. This is not just a ploy to maximize sales of the book, but a reflection of the fact that, in real-world bioinformatics, all three operating systems are commonly used. In general, Linux is the operating system of choice in bioinformatics, because it is well suited to running servers and is very scalable, allowing solutions to be developed for anything from individual desktop PCs through to multi-processor supercomputers. However, the familiarity, usability, and availability of Windows and Mac OS ensure they are also widely used, particularly by those starting out in bioinformatics, and among the biologists using the software that we produce. Indeed at the time of writing, some of the most popular freely available software tools for proteomics run exclusively on Windows.

Where there are differences between operating systems, typically when installing software, we have provided instructions for all three. Clearly, we have not been able to test the material in this book on every available Linux distribution, or on versions of Windows or Mac OS that were released after the book went to press. If you have any problems getting the examples in the book to work, we recommend that you head over to the book's website (`www.bixsolutions.net`) in search of a solution.

1.2.4 www.bixsolutions.net

To help you as you work through this book, we maintain a companion website, `www.bixsolutions.net`, where you can find the main example programs and data from the book, as well as up-to-date lists of recommended reading. The site also has a discussion forum, which is monitored by the authors, who are happy to help if you have any difficulties while working through the examples. Obviously, we have checked the examples very carefully and believed them to work correctly when the book went to press, but they may not work forever. This is because the tools and, indeed, the operating systems, used in this book are constantly evolving—new versions are released frequently, so functions may be deprecated, database schemas may change, and new features may become available. As we become aware of any changes that affect material in the book we will post updates or workarounds on the website.

For readers familiar with code repositories, we also maintain a GitHub repository (`github.com/dazoakley/bbs-v2`) containing the main example programs from the book. For those new to code repositories, this topic is covered in Chapter 6.

1.3 Principal applications of bioinformatics

Since its origins in genomic sequence analysis, bioinformatics has spread across the whole of molecular biology, in its broadest sense. There are already many fine texts explaining the various applications of bioinformatics (e.g. Lesk, 2008), so we don't replicate such material here. However, we include below a brief outline of the main areas of contemporary bioinformatics with particular emphasis on the application to those areas of the tools introduced in this book.

1.3.1 Sequence analysis

Sequence analysis is a massive field, covering all manner of analysis of textual sequences representing genomes (DNA) and proteins (sequences of amino acids). Applications within genomics are wide ranging and include sequence assembly, prediction of coding regions (i.e. genes), determination of genomic structure, research into the purpose of non-coding DNA, translation of DNA into protein sequences, comparison of sequences to infer evolutionary relationships, rates of evolution, the study of variation between individuals, and prediction of gene function and regulation. DNA sequence analysis is also used in the design of experiments that employ techniques such as PCR and microarray technology.

Protein sequence analysis has a similarly wide range of applications, including protein structure prediction, inference of protein function based on sequence similarity, determination of protein similarity for building protein families and understanding protein evolution, and identification of structural or functional subsequences (*motifs*). Protein sequences are also used when designing and interpreting the results from proteomic (protein expression) experiments.

From a technical point of view, a distinguishing feature of sequence analysis applications is the frequent use of relatively large data sets such as whole genome sequences (for example, the human genome, which consists of about 3,000,000,000 bp), or multiple genomes when studying variation or metagenomics. As well as having to deal with the sequences themselves, it is also necessary to handle *annotations*, without which the sequence data is meaningless. Such annotations include things like the species from which a selected sequence originated, its location in the genome, and any functions or gene products that have been associated with it. A RDBMS, such as MySQL, is essential to store such data in a useable way. Another distinguishing feature of sequence analysis is the need to efficiently process textual data, essentially long strings of As, Cs, Gs, and Ts, or the 20-letter alphabet used to represent amino acids. Typical tasks include looking for motifs within much larger sequences and looking for similarities between sequences. Handling large amounts of textual data is something at which Perl excels, as we will see in Chapter 3.

1.3.2 Transcriptomics

With the invention of DNA microarrays it became possible, for the first time, to monitor the expression level of all known genes in an organism (e.g. approximately 22,000 genes for humans) in a single analysis. Such genome-wide expression analysis has become known as *transcriptomics*. This yields large sets of highly multivariate quantitative data, as studies tend to involve multiple samples so that differential expression or behaviour over time can be observed.

More recently, the high throughput sequencing-based RNA-seq method has become a popular alternative for transcriptomic studies. A key advantage of RNA-seq is that it sequences cDNA generated from the transcribed mRNA, whereas the fundamental principle of microarray analysis is the hybridization of cDNA to pre-fabricated DNA probes. RNA-seq is therefore a more open technique, particularly useful for non-model organisms or studies where multiple species or sequence variation are of particular interest.

Just storing expression data and associated information about that data (the so-called *metadata*, which describes the sample and experimental conditions) can be an issue, but the major challenge lies in the processing and statistical analysis of gene expression data. In microarray experiments, the raw data is in the form of an image that needs to be processed and cross-referenced with metadata describing the array design (i.e. which spot relates to which gene) to yield numerical expression values. The raw data in RNA-seq studies consists of millions of short sequence reads, which are typically mapped to a reference genome for identification purposes, and the abundance of which are used to achieve quantitation.

Once the expression values have been extracted, assigned to genes, and tabulated, we can begin statistical analysis to identify biologically important features, such as differentially expressed genes in a comparative study or co-regulated genes in a temporal study. Due to the large data sets and sometimes sub-optimal experimental designs (the number of genes monitored usually exceeds the number of samples analysed) the statistical methods required can be complex. This is where R is extremely valuable, as the common processing algorithms and even some of the more exotic methods have already been implemented for us in R packages, such as Bioconductor. Chapter 4 will get you started with R and, having worked through it, you should be in a position to start working with packages like Bioconductor.

1.3.3 Proteomics

Gene expression studies are essential in understanding gene regulation and related phenomena, but to gain a deeper insight into how a biological system functions it is arguably more useful to look at the expression of the functional molecules themselves: the proteins. Proteomics is the science of identifying and, where possible, quantifying proteins in a sample. At the time of writing, the majority of proteomics methods are based on mass spectrometry (MS), coupled with prior protein digestion and separation steps to help ensure that peptides are delivered to the mass spectrometer individually. Due to the complexity and variety of proteomic protocols, handling the data from these experiments can be a major challenge. Some laboratories separate proteins using two-dimensional gel electrophoresis, after which image analysis is employed to identify and define gel spots, prior to excision and MS analysis. Today it is more common to use liquid chromatography (LC) instead of gels as this supports a much higher throughput, resulting in large data sets comprising several thousand spectra per sample.

Whichever separation technique is used, data analysis is required to identify the peptide represented by each mass spectrum. The most common techniques are peptide mass fingerprinting (PMF) or, if peptides have been subjected to secondary fragmentation in a subsequent MS stage, a search against simulated spectra derived from a database of known protein sequences for the species being studied. Identified peptides then need to be assigned to proteins, which is not easy due to the relatively short length of each peptide and the many proteins that could potentially be present in a sample—in human tissue this could be several hundred thousand if splice variants and post-translational modifications are

taken into account. Due to the large data sets and the experimental metadata needed to make use of this, a well-designed relational database (perhaps implemented in MySQL) is essential to organize the information, and the increasingly high throughput nature of proteomics necessitates analysis pipelines to be built (often in Perl) if data analysis is to keep pace with data acquisition. Having said that, certain steps of the processing have been shown to be amenable to large-scale, non-RDBMS (NoSQL) methods, which we touch on briefly in Chapter 2.

Having identified proteins from all the samples in a given study, statistical analysis then needs to be performed across samples to extract the biologically significant information from the acquired experimental data. Some proteomics protocols are only qualitative—they aim to determine which proteins are present in a sample, but not how much of each protein is present. However, a raft of quantitative protocols is now in use, which can reveal not just the identities of proteins, but also their abundance. Statistical techniques similar to those used for gene expression analysis are clearly applicable to this data and, as these protocols are adopted more widely, relevant R packages are emerging.

1.3.4 Metabolomics

Metabolomics deals with the identification and quantification of small molecules in biological samples. Analysis of metabolites is one of the most well-established bioanalytical techniques that we come across in bioinformatics. The primary technologies used—nuclear magnetic resonance (NMR) and mass spectrometry following gas chromatography (GC) or liquid chromatography (LC)—have been around considerably longer than high throughput sequencing, microarrays, or proteomic MS. Indeed, it is even possible to buy pocket-sized devices for personal monitoring of medically important metabolites, such as glucose and cholesterol. However, metabolomics brings a new emphasis to high throughput analysis and the desire to quantify as many analytes as possible in each sample. The desire for such a global view of the metabolome is being driven mainly by the search for diagnostic biomarkers and the growth of systems biology. As in proteomics, increasing data volumes are encouraging people to put together relational databases and software pipelines for metabolomic data analysis. As metabolite analysis is well established, so are the core algorithms for dealing with the data from such analysis—such methods often come under the banner of *chemometrics*, about which some very good introductory texts have been written (e.g. Brereton, 2007; Otto, 2007). R is a perfect environment for such analysis, as demonstrated by some of the examples in Chapter 4.

1.3.5 Systems biology

Traditionally, bioanalytical science, and even the bioinformatics that supports it, is broken down into the areas outlined previously—genomics, transcriptomics, proteomics, and metabolomics. In terms of how organisms function, these delineations are artificial because in reality the genes, proteins, and metabolites are free to interact. Systems biology recognizes this and, in bioinformatics terms, comprises integration of both data storage and analysis to permit system-wide

analysis and modelling of living organisms without being constrained to a particular class of molecule. This is likely to be a theme of bioinformatics for some time to come, because the potential outcome of such work—cell and tissue models with application in areas such as toxicology—is so valuable, but the challenge of achieving this is immense. The core elements of database design, programming, and numerical data analysis covered in this book are all important in systems biology, and much of the effort in systems biology is focused on the design of databases for meaningful storage of heterogeneous data and novel methods for data analysis and visualization.

1.3.6 Literature mining

PubMed, which contains bibliographic information on journals primarily associated with biomedicine (see Section 1.5.1), is growing by around 3,000 citations every day. Keeping up with the work described in all these papers by reading them is clearly impossible, and that is before we consider the backlog of at least 20 million papers already out there. There is, therefore, a lot of interest in literature mining methods that help to facilitate the high throughput machine reading of papers to extract salient information, and advanced ways of searching through and annotating papers to assist human reading. Perl's text handling capabilities make it ideal for this type of work, as does its ability to automate querying of bibliographic databases and retrieval of papers from websites. There can also be a need to visualize the results of literature mining, which can be handled by Perl or R. For example, if we wrote a Perl program to extract protein–protein interactions from text, it would be convenient to display the resulting network of interactions graphically, as this is a representation with which biologists are familiar.

1.3.7 Structural biology

Structural biology is the study of the physical architecture of biological molecules—particularly proteins. Research typically focuses on topics such as the relationship between structure and function, structural similarity between proteins, simulation of interaction between proteins and other molecules, and the relationship between protein sequences and their structure (particularly the process of protein folding). Some of these topics can be tackled using the tools introduced in this book, but due to the mathematical complexity of molecular simulations, it is common for researchers to use existing modelling tools (some of them commercial products) or to produce bespoke software in lower level languages, such as C. Such tools are sometimes pipelined using Perl, with a program being written to retrieve a structure from a repository of protein structures, such as PDB (see Section 1.5.1), pass it to a modelling program, and deposit the final result of the modelled experiment into a local database.

1.4 Building bioinformatics solutions

The premise of this book is that, regardless of the particular type of data being analysed, or the scientific purpose behind the analysis, the tools and general

approaches used to solve a bioinformatics problem are often the same. This is because most bioinformatics projects share a similar aim—to bring together data (be it public or proprietary) with analysis tools (be they existing or novel) to generate new biological knowledge. We refer to this as building a bioinformatics solution because it best describes what we are doing—putting things together to solve a problem.

A schematic representation of this concept is shown in Fig. 1.1. Regardless of the particular type of data being analysed or the scientific aim behind the analysis, the general structure of a bioinformatics solution tends to follow this pattern, although not all components are required in all applications. For example, it may be possible to analyse novel data collected locally without recourse to any public databases. Similarly, we might be able to rely entirely on our own in-house analysis software, instead of sourcing tools from the public domain. Conversely, we may not have access to any novel data or novel analysis routines, and instead focus entirely on the analysis of public data using publicly available tools. The way in which results are output from our system may also vary. The figure shows two potential routes, which may be used together. These are deposition of the results in a local database, and presentation of the results to one or more users via a web browser. Unless the software is truly single user—intended only for use by the developer—some kind of user interface is required, and a web-based interface is convenient because web interfaces are familiar to users, are relatively easy to implement, and should be platform independent. The web obviously allows you to make your software available to remote users around the world, but even if you are developing a solution for local use in a single group or organization, a web interface is often the best way to go.

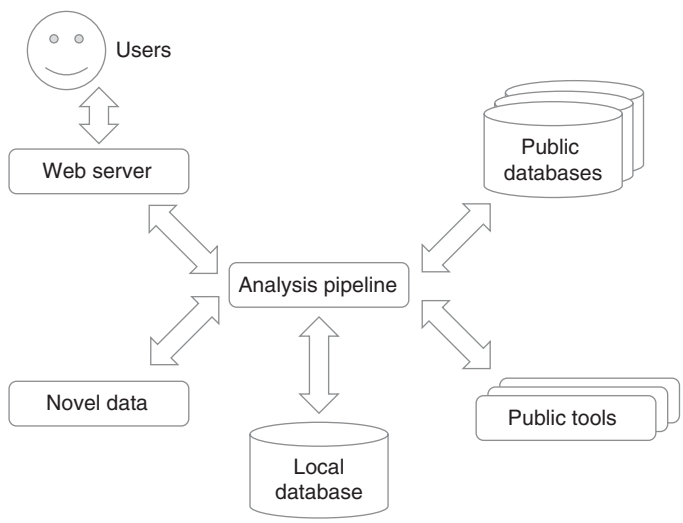


Fig. 1.1 A generic bioinformatics solution showing typical components that may be used.

So, the key components of a system like the one shown in the figure are the analysis pipeline, the local database, and the web interface. The process of building a bioinformatics solution starts with capturing the project requirements, typically by manually carrying out analysis on a small subset of the data being collected and, if appropriate, talking to users about what they want from the software. This should reveal which components are needed and what functionality they should have. These components can then be produced using the tools described in the following chapters, specifically the pipeline (with Perl and R), the local database (with MySQL), and the interface (with HTML5, a web framework and Apache). For completeness, a brief overview of other potential components—the public tools and databases—is provided in the next section.

1.5 Publicly available bioinformatics resources

We are very fortunate to have a substantial body of high quality, freely available bioinformatics resources accessible via the Internet. Most of the core resources, or services as they are sometimes called, are hosted by two major organizations—the American National Centre for Biotechnology Information (NCBI) and the European Bioinformatics Institute (EBI), which is based in the UK. Resources provided by these organizations are easily accessed via their websites (`www.ncbi.nlm.nih.gov` and `www.ebi.ac.uk`, respectively). These resources can roughly be broken down into databases and analysis tools, although the distinction is becoming ever more blurred as the websites through which databases are accessed begin to include more sophisticated integrated analysis tools. What follows is a brief overview of the key resources at these sites and on the wider Internet.

1.5.1 Publicly available data

There exists a substantial global collection of data covering a wide range of biological areas, from gene sequences to protein structures and medical information. Furthermore, this collection is growing all the time, both in terms of the number of databases and the amount of data in each database. Indeed, many of the databases are growing exponentially.

This impressive collection of data forms the basis for much bioinformatics work, especially among the many researchers who do not have laboratories or laboratory-based collaborators. Even when we are working with novel proprietary data, we often need to analyse this in the context of publicly available data. For example, if we are looking at SNP data, we may want to map this to existing genomic annotations to identify potential consequences of that SNP. In gene expression studies, it is common to use publicly available information about gene function to provide context for the analysis. In proteomics, it is common practice to identify peptides by searching acquired mass spectra against protein sequence databases.

Due to the extent and rapidly changing nature of the database landscape, we are unable to provide a thorough review of all available resources here. Instead,

we would refer you to the annual *Database Collection* published in the journal *Nucleic Acids Research* (`www.oxfordjournals.org/nar/database/c`), which has summaries of many hundreds of databases. However, it is worth taking a little space here to introduce what we consider to be the key core databases at the current time, particularly as some of these are referred to in later chapters.

Genome sequences

The core repositories for nucleotide sequence data are Genbank (at NCBI), ENA (at EBI), and DDBJ (at the Japanese National Institute of Genetics). These three databases comprise the International Nucleotide Sequence Database Collaboration (INSDC), which collectively capture all the public genome sequence data ever collected. There is even some putative—though seemingly dubious—dinosaur DNA in there if you look hard enough (accession number[2] U41319). As part of the collaboration, the contents of these three databases are automatically synchronized daily, so a sequence submitted to one will appear in the other two. All three databases therefore contain the same data. This may seem wasteful, but having the databases in different geographical locations increases data security and ensures that most researchers have the data reasonably close to them, which improves efficiency of access. Also, the three databases are distinct in that they have different user interfaces.

Although the three INSDC databases are exhaustive in terms of cataloguing all available nucleotide data, the organization of the data is fairly rudimentary and, as a result, they are not the easiest databases to browse. Consequently, many *secondary databases* have appeared, in which the primary data from repositories such as those in the INSDC has been compiled and indexed into a form that is highly structured, easy to browse, and well integrated with other resources. An excellent example of this is Ensembl (`www.ensembl.org`), which provides information about a number of completed eukaryotic genomes. As well as the original sequence data, assembled into whole chromosomes, Ensembl also provides annotations for this data, such as known genes, gene predictions, gene structure, gene products, orthologues, and SNPs. As well as being accessed via the web interface, Ensembl can also be accessed by programs directly via an application programming interface (API), or can even be downloaded in its entirety for working with locally, which can be faster and arguably offer more privacy than connecting to the database via the Internet.

Protein sequences

The most comprehensive central resource for protein sequences and functional annotation is UniProt (`www.uniprot.org`). The UniProt knowledgebase comprises two main sections: a large database called TrEMBL, which contains protein sequences produced by automatic translation of ENA nucleotide sequences, and a

2 An accession number is a unique identifier for a particular record in a database. The record can be found simply by searching for this number.

much smaller database of manually curated protein sequences called Swiss-Prot. Thanks to the extensive manual curation, which includes the addition of cross-references to many other databases, the data in Swiss-Prot is recognized as being of very high quality and it is considered to be the gold standard protein sequence database. As well as individual protein records, the UniProt knowledgebase also contains complete proteomes for an increasing number of organisms. Many other protein databases exist, providing information related to or derived from UniProt sequences. A notable example is InterPro (`www.ebi.ac.uk/interpro`), which provides information about protein families, domains, and functional sites.

Transcriptomic data

There are two main databases of gene expression data: Array Express (`www.ebi.ac.uk/arrayexpress`) and the Gene Expression Omnibus—GEO (`www.ncbi.nlm.nih.gov/geo`). These databases support data from both microarrays and RNA-seq. A key element in establishing these databases was finding agreement within the community for a common way in which to report all the relevant details of a microarray experiment. This was finally achieved by the definition of the MIAME (Minimum Information from A Microarray Experiment) reporting standard, and databases such as GEO and ArrayExpress are fully MIAME compliant. Having this consistent method of representing data is important because it means that we can automatically download expression data complete with all the metadata necessary to interpret it, regardless of its origin. Also, having a common data standard means that people are willing to spend time developing software that supports it. Indeed, we will see in Chapter 4 that there are R packages to deal directly with this type of data.

Protein expression data

Proteomics is following transcriptomics both in terms of uptake in the laboratory and in terms of data repositories. Like transcriptomic data, proteomic data is only useful if metadata is available to provide context to that data, and this issue has largely been addressed through the development of reporting guidelines and data formats by the Proteomics Standards Initiative (PSI). Substantial databases of protein identifications and associated mass spectrometry data have already emerged, most notably PRIDE (`www.ebi.ac.uk/pride`) and PeptideAtlas (`www.peptideatlas.org`).

Metabolomic data

Repositories of metabolomics data sets have taken some time to appear due to the difficulty in coherently capturing data from the many diverse protocols employed. In contrast to proteomics and transcriptomics, many metabolomics studies employ profiling methods where phenotypic differences are inferred directly from analytical data without first identifying all (or sometimes any) of the molecular entities represented by that data. This is, in part, because the analytical signatures of many metabolites are not known so they simply cannot be identified. However, the situation is improving thanks to the development of databases that catalogue small molecules and their analytical signatures. These include the

Human Metabolome Database (www.hmdb.ca), Metlin (metlin.scripps.edu), PubChem (pubchem.ncbi.nlm.nih.gov), and ChemSpider (www.chemspider.com). These databases are often used as libraries to identify compounds from mass spectra acquired during metabolomics experiments. In terms of the data and findings resulting from such experiments, MetaboLights (www.ebi.ac.uk/metabolights) leads the way at the time of writing.

Molecular structures

The pre-eminent protein structure database is the Protein Data Bank—PDB (www.pdb.org). This contains protein structures determined primarily using X-ray diffraction or NMR. The information about each structure is quite exhaustive, including information about who determined the structure and how they did it, biochemical information about the protein and, of course, the structure itself, in formats that can be viewed within PDB's web interface, and that allow the structure to be downloaded and analysed locally. Other protein structure databases, many derived from PDB, are available, as well as databases of smaller molecules.

Interactions and pathways

The databases mentioned thus far are primarily collections of experimental data from laboratory instruments, albeit augmented with manually generated annotations. Pathway databases are different in that they contain interaction data *derived* from experiments, rather than the experimental data itself. Such databases are becoming increasingly important as systems biology studies become more common. One of the most well-established pathway databases is KEGG (the Kyoto Encyclopedia of Genes and Genomes) Pathway (www.genome.ad.jp/kegg/pathway.html). KEGG comprises several databases, with the Pathway database dedicated to data pertaining to molecular interaction networks. Much of this information has been available for some time in books and papers, but having it online in electronic form facilitates easier access, new applications, and better integration with other resources.

Other useful databases in this general field include the Reactome pathway database (www.reactome.org), the IntAct database of molecular interactions (www.ebi.ac.uk/intact), and BioModels (www.ebi.ac.uk/biomodels-main), which is a collection of mathematical models of biological systems.

Literature

When we think of scientific literature, journals come to mind and, indeed, it is here that most scientific literature can be found. Despite the existence of the repositories of experimental data described above, journal articles still play an essential role, acting as the glue that links a lot of this data together and gives it context. It is also where the biological knowledge extracted from experimental data is reported. Today, the vast majority of journal articles are available online, although not all are available free of charge (open access). A large body of bibliographic information about papers, including abstracts, is available in the PubMed database (www.pubmed.gov).

Journal articles are not the only source of textual information. One notable repository is OMIM (www.ncbi.nlm.nih.gov/omim), which contains textual descriptions of all known human genetic disorders. These descriptions include links to supporting data in many of the repositories described above. Similarly, there are links to OMIM from many of these repositories.

Ontologies

In bioinformatics terms, ontologies are essentially lists of terms with strictly defined meanings that have been agreed upon by the scientific community. Where more than one word can be used to describe the same thing, synonyms are listed along with the definition. To avoid any confusion or duplication, each term has a specific accession number associated with it. The most commonly used ontology in bioinformatics is the Gene Ontology—GO (www.geneontology.org). It may sound like an ontology is little more than a dictionary, but an important additional feature is that relationships between terms are also captured. For example, in GO the term 'carbohydrate binding' is defined as a subset of the molecular function 'binding', and many specific types of binding, such as 'glucose binding', are linked to that more general 'carbohydrate binding' term.

Ontologies have many uses in bioinformatics. At their simplest, they can be used as *controlled vocabularies*—lists of terms that users are restricted to selecting when entering data. A typical way of enforcing a controlled vocabulary on users is to only allow input via a drop-down box, which contains only the permitted terms. Retrieving and analysing data using automated systems is a lot easier if it is annotated in this way, rather than with free text descriptions. For this reason, many of the standard data formats, such as the MIAME microarray standard, make use of ontologies for their metadata. Ontologies can also be used to facilitate advanced querying of data. For example, searching for documents using a normal keyword search for 'carbohydrate binding' would simply return documents containing that term, whereas a search augmented by GO could additionally return documents that do not contain the term 'carbohydrate binding', but do contain terms that GO defines as being related to this, such as 'glucose binding'. Thanks to such utility, ontologies appear in almost every area of bioinformatics.

GO is currently the most ubiquitous of biological ontologies, as it covers three key areas of interest: biological process, cellular component, and molecular function. However, there are many complementary ontologies, most of which can be found at the Open Biomedical Ontologies Foundry (www.obofoundry.org). Although there are various ontology formats, the important thing is that these ontologies are freely available, easily machine readable, and can therefore be incorporated into our own programs with relative ease.

1.5.2 Publicly available analysis tools

As well as data, there is also a wealth of data analysis tools freely available via the web. Some of these are add-ons for specific software such as Perl (Perl modules) and R (R packages), which we will deal with in the later chapters. Many other tools

are available in standalone form, either accessible via web front ends, or as programs that you can download and run locally. The most frequently used tools can be found among the services listed at the EBI (www.ebi.ac.uk/services) and via the Tools tab on the NCBI resources page (www.ncbi.nlm.nih.gov/guide/all). Due to the fundamental importance of sequence analysis, these toolboxes tend to be biased towards sequence analysis, with BLAST and the ClustalW multiple sequence analysis tool arguably being the most well known. However, other tools, such as OMSSA (at NCBI) for proteomic mass spectrometry and DaliLite (at EBI) for pairwise structure comparison, clearly cover other data types.

For occasional analyses, as done by the typical laboratory-based biologist, the web-based interfaces to these tools are very convenient. The good news for those of us seeking to build bioinformatics solutions with higher throughput is that most of these tools can also be accessed programmatically, allowing us to incorporate them into our own software and automate analysis. This can be done either by downloading versions of the tools that can be run locally and incorporated into your programs, or by connecting to a server at the EBI or NCBI, passing across the data of interest and running the tool there. The latter is achieved using an API that allows data and commands to be sent directly to the tool on the server, and results returned. The details of how to do this vary from tool to tool, but comprehensive instructions and examples are available at both the NCBI and EBI toolbox websites, and an example is given in Chapter 3.

Another suite of tools worth knowing about, especially if you are working with sequence data, is the European Molecular Biology Open Software Suite—EMBOSS (emboss.sourceforge.net). This open source suite mainly comprises tools for sequence analysis, such as sequence alignment, picking primers, and looking for sequence motifs. The programs that make up EMBOSS are well respected and easy to incorporate into your own software, making them a popular choice for bioinformatics developers.

There are many other freely available tools out there, but because these tools tend to be developed by different academic groups, they can be hard to find. In our experience, the best way to discover new tools is by monitoring relevant journals, although even that is a challenge, as tools might be announced in bioinformatics or domain-specific journals, such as those covering genomics, proteomics, or metabolomics.

1.5.3 Publicly available workflow solutions

As already mentioned, many bioinformatics applications require the connection or integration of various individual tools. Writing Perl programs is arguably the most flexible way to do this, but not everyone has the skills necessary to do that. In response to this need, a number of software platforms have emerged that allow integration of different bioinformatics tools and resources without any need for programming. These platforms act as a host environment for other tools, enabling them to be joined together into complex pipelines that can be saved for future use and shared among the scientific community. Most of these have graphical interfaces that allow workflows to be created by dragging graphical

representations of tools into position and connecting them together, and the best of them take care of optimizing the execution of the workflow on whatever computer hardware you have to hand.

Bioinformatics workflow platforms include Taverna (`www.taverna.org.uk`), Knime (`www.knime.org`), and Galaxy (`usegalaxy.org`). At the time of writing, Galaxy is most popular of the three. Indeed if you ask biologists which bioinformatics tools they use, many will say Galaxy ahead of—or even instead of—the tools within Galaxy that actually do the work. The vast majority of tools supported by Galaxy are for analysis of sequence data, but efforts are underway to add support for other data, such as mass spectra, to serve the proteomics and metabolomics communities.

Do workflow platforms such as Galaxy spell the end for programming in bioinformatics? Not at all. These platforms can do a lot of the tedious work of connecting tools and distributing compute jobs, but the development of new tools for data storage, analysis, and visualization is needed as much as ever. Indeed, environments like Galaxy provide an excellent route to generating demand and maximizing uptake of new tools that we produce. Furthermore, whenever it becomes necessary to add a newly created tool or data format to a workflow platform, there is invariably some programming to do.

1.6 Some computing practicalities

Finally, before we get into the substance of this book, we need to mention a few practical issues that apply to all of the following chapters.

1.6.1 Hardware requirements

A question we are often asked by people looking to develop their bioinformatics skills is: 'what kind of computer do I need?' For some people, the expectation is that, with all the talk about large data sets, whole genome analysis, and complex visualizations, high-end computing resources must be required. In practice, however, most bioinformaticians are able to do what they need on a standard desktop computer, and that is definitely the case for this book. Any recently produced PC or Apple Mac, in either desktop or high performance laptop format, connected to the internet, should be fine for running the examples in this book. For the avoidance of doubt, tablets, chromebooks, and similar lightweight devices are not a sensible choice for this type of work.

The alternative to using your own computer is to adopt a client–server approach, as practised in many organizations. In this scenario, a powerful computer is set up as a server and core software is installed on it. In bioinformatics, the server would typically be running a Linux operating system, with MySQL (or an equivalent such as Oracle), Perl, R, and Apache web server installed. You would connect to the server from a separate (client) computer to query databases, execute commands, and run software that you have created. This may sound like a lot of unnecessary hassle, but it has several benefits if multiple users or software developers are involved. First, because it is not necessary to physically

sit at the server to use it, it can be used simultaneously by a number of people. This makes it possible to share, for example, a single installation of MySQL or even a specific database. This considerably reduces administrative duties, because the software only needs to be installed and configured once, regardless of the number of users. This approach is therefore very popular in larger organizations, such as universities and companies that are big enough to have dedicated system administrators. The client–server approach also provides separation between the server and the personal computer on which you write program code, check your email, browse the web, and so on. This has several advantages:

- The server can be in a remote location, such as a data centre with better security and an uninterruptible power supply.
- You can run a different operating system on your personal computer to the operating system that is installed on the server.
- You can switch off, reboot, or otherwise abuse your computer without affecting the server.

All this is particularly useful if you are developing bioinformatics tools that need to be accessible to other people and therefore need a high level of reliability.

Cloud computing is a modern manifestation of the client–server approach. The main difference compared to client–server is the fact that in a cloud environment the remote computing resource may be geographically distributed across multiple servers and it may be supplied as a product, paid for according to the amount of storage and processor time that your software uses. This has the attraction that you are not burdened with the upfront costs of purchasing and commissioning large pieces of computer hardware, and brings the benefits of economy of scale through sharing running costs among many other users.

Despite the advantages of working in a server or cloud-based environment, when starting out we recommend you use a single computer for which you have administrator rights. This is what we assume throughout the book. However, if you are in an organization where you wish to, or are forced to, use a server you shouldn't have any major problems with the examples. You will just need to liaise with the server administrator to find out what software is available on your particular server and how to access it.

1.6.2 The command line

In most of the examples throughout this book we interact with the computer via command-line interfaces. In a command-line interface, instructions are issued to the computer simply by typing commands. Any feedback is returned to the screen. This may seem like a step backwards in a world of mice, touch screens, and graphical user interfaces (GUIs), but for some tasks commonly carried out in bioinformatics, such as querying databases or manipulating large data sets, command-line interfaces can actually be much more efficient than GUIs. Furthermore, if we can get the computer to do something by typing in a command, we can incorporate that command into a program as part of an automated process.

For those readers unfamiliar with command-line interfaces, we provide a basic primer in Appendix A. If you are not confident with working at the command line, you should take a look at that before moving on to the following chapters.

1.6.3 Case sensitivity

When using command-line tools or writing software, it is essential to be aware of whether the tool or language you are using distinguishes between capital and lower case letters. For example, would it consider something called 'bioinformatics' to be distinct from 'BIOINFORMATICS' or, more subtly, 'Bioinformatics'? This can lead to all sorts of problems, especially for beginners, because commands that superficially look correct may not work. To make things even more confusing it is not just the tools that differ in their case sensitivity—the behaviour of operating systems differs too. Specifically, Windows ignores case, so a file called 'DNA.TXT' could be referred to on the command line as 'dna.txt'. If you tried to do this on Linux or Mac OS you would receive an error saying that the file could not be found. Indeed, in such case-sensitive operating systems two distinct files 'dna.txt' and 'DNA.TXT' could co-exist in the same directory, although this is hardly a desirable situation.

Our recommendation throughout this book is to assume case sensitivity, regardless of the tool, language, or operating system you are using, but make sure that you avoid using names that would be identical if the case was ignored. This is the approach we have followed in the examples in the chapters that follow. Not only does this avoid confusion, it also helps ensure that any programs you write are portable between operating systems.

1.6.4 Security, firewalls, and administration rights

Ever since computers started being connecting to networks, there have been concerns about unauthorized people connecting to computers and accessing data or installing malicious software. This is clearly a particular concern for those in bioinformatics working with sensitive data, such as novel compounds, or clinical data that needs to be guarded for ethical reasons. There is also the danger of criminals taking over networked computers and using them to send out spam email, which can have serious consequences for organizations as they can end up being *blacklisted*—having all their outgoing mail flagged as spam. For these reasons, organizations have become increasingly strict in limiting what users can do with their computers, and operating systems and server software has become loaded with more security features. Basic security features include the use of password protected user accounts to access a computer or particular data, and a firewall to filter network traffic to and from the computer.

We mention this here because the material in this book goes beyond what many organizations expect you to be doing with your computer. In particular, you will need to install software on your computer if you don't already have it, and in Chapter 5 your will be using your computer as a web server. Installing software requires you to have administrator rights on the computer and running server software may require you to modify firewall settings if you want to

access the server from another computer. Neither of these things are inherently dangerous, but getting such access can be tough if your computer belongs to your organization, and is set up and administered by them. If you are working or studying in an organization, our advice is to talk to your IT support people and explain what you want to do. Depending on the organization, it is possible that the software you need may already be installed, either locally or on a server.

Even if you are using your own computer, you may need to spend a little time ensuring you have administrator rights and tweaking firewall settings. The good news is that security issues are only likely to be a problem when installing and setting up the tools used in this book, typically at the start of the chapter. Once you have successfully got MySQL, Perl, R, and so on up and running, you should be able to proceed through the chapter without further hindrance. Also, software installation and configuration is a generic issue, not specific to people using this book or even to people working in bioinformatics, so there is likely be help available on the web for almost every eventuality. Indeed, you can always head over to `www.bixsolutions.net` to report the issue and seek a solution.

References

Brereton, R. G. (2007). *Applied Chemometrics for Scientists*. Wiley: Chichester, UK.

Christiansen, T., Foy, B., Wall, L., & Orwant, J. (2012). *Programming Perl*. O'Reilly: Sebastopol, California, USA.

Laurie, B. & Laurie, P. (2013). *Apache: The Definitive Guide*. O'Reilly: Sebastopol, California, USA.

Lesk, A. M. (2008). *Introduction to Bioinformatics*. Oxford University Press: Oxford, UK.

Otto, M. (2007). *Chemometrics: Statistics and Computer Application in Analytical Chemistry*. Wiley: Chichester, UK.

Tahaghoghi, S. M. M. & Williams, H. E. (2009). *Learning MySQL*. O'Reilly: Sebastopol, California, USA.

CHAPTER 2

Building biological databases with SQL

A database is at its simplest a set of stored information, such as a filing cabinet or a computer's hard disk. Generally pieces of similar or related information are gathered together in the same place, as common sense would dictate and as you probably already do when you create folders and subfolders for information held on your computer. Database concepts provide a way of formalizing the gathering together of this data such that the relationships between pieces of information are consistent. They can therefore be more efficiently used, whether through manual or automated processes, and the structure provides a means by which data consistency may be maintained.

This chapter focuses primarily on a type of database called a *relational database*. Relational databases are powerful because they enforce a great deal of security and consistency on the data within them. The software tools that are used to create and manage relational databases are called *relational database management systems* or RDBMSs. These allow the data contained within a database to be queried in immensely powerful ways, often using very simple commands created using a special programming language called the Structured Query Language (SQL).

Also briefly introduced in this chapter are two other types of database commonly encountered in bioinformatics: flat text files, such as FASTA files, containing sequence information, and Extensible Markup Language (XML) files, which are a key component of most modern data standards, such as the HUPO-PSI standards for proteomic data. Understanding these types of database is easier than for relational databases, so they do not form the bulk of this chapter.

Finally we will also introduce the concept of NoSQL databases, a class of diverse solutions to data storage and access that have arisen in response to the increasing need to access records across very large data sets in very short time frames. There are some instances where the use of such a technology in bioinformatics applications is more appropriate than using an RDBMS.

At first glance the ordering of this chapter may seem strange—the installation of a relational database system follows extensive sections on databases and database design, and database access through SQL is covered last. This is deliberate. The hardest aspects of understanding and dealing with databases occur at the

Building Bioinformatics Solutions. Second Edition. Conrad Bessant, Darren Oakley and Ian Shadforth.

design stage, which is also the most important. Installing a RDBMS is straightforward and also unnecessary for good database design. However, a working system is needed to experiment with accessing databases, and hence installation and connection issues are discussed after database design but before database interaction.

2.1 Common database types

2.1.1 Flat text files

As stated above, a database is merely a store of data, one of the simplest forms of which would be to write the data as a set of text files, often termed *flat files*. These flat files could be any text file created in a commonly readable format, such as the `.txt` files created in text editors like Windows Notepad. A collection of text documents on a hard disk is one example of a database of flat files. In order to assist automated access and, indirectly, readability, it helps if some sort of structure is imposed upon the data within a text file. In terms of scientific papers, this order is often along the lines of Introduction, Materials and Methods, Results, Discussion and References, or variants of this type. The structure helps the reader to quickly locate information of interest by navigating first to the relevant section. Subheadings further help this cause. For automated reading, or *parsing*, of data within a file, it helps if the structure is highly consistent. The headings within scientific papers may differ due to a number of factors, such as different journal formats. On the other hand, if we consider a basic implementation of the FASTA flat file format for storing sequence files, presented below, the structure is much simpler but allows for both intuitive human and machine reading.

A simple example of a FASTA format file would be:

```
>ENSP00000630516 | a protein description
SEQUENCEAPPEARSHERE
>ENSP00000295897 | another protein description
THESEQUENCEOFTHISPROTEIN
```

The FASTA file features four structural elements:

1. Information about each protein is introduced by a greater than (>) character
2. The first piece of data to follow the > character is the protein accession number, in this case the Ensembl accession number. This is followed by a bar (|) character.
3. Following the bar, we have the protein description. There is then a newline character that is not directly visible, but results in a new line being started. This is the indication that the protein sequence follows. In terms of human readability, this is clearly indicated by the start of a new line on which text that looks like a sequence is presented, but in machine readability terms, it's the invisible newline character that is used to differentiate between protein description and protein sequence.