Statistical Applications in Genetics and Molecular Biology

Volume 4, *Issue* 1 2005 *Article* 2

Reproducible Research: A Bioinformatics Case Study

Robert Gentleman, *Harvard University*

Recommended Citation:
Gentleman, Robert (2005) "Reproducible Research: A Bioinformatics Case Study," *Statistical Applications in Genetics and Molecular Biology*: Vol. 4: Iss. 1, Article 2.
DOI: 10.2202/1544-6115.1034

Reproducible Research: A Bioinformatics Case Study

Robert Gentleman

Abstract

While scientific research and the methodologies involved have gone through substantial technological evolution the technology involved in the publication of the results of these endeavors has remained relatively stagnant. Publication is largely done in the same manner today as it was fifty years ago. Many journals have adopted electronic formats, however, their orientation and style is little different from a printed document. The documents tend to be static and take little advantage of computational resources that might be available. Recent work, Gentleman and Temple Lang (2003), suggests a methodology and basic infrastructure that can be used to *publish* documents in a substantially different way. Their approach is suitable for the publication of papers whose message relies on computation. Stated quite simply, Gentleman and Temple Lang (2003) propose a paradigm where documents are mixtures of code and text. Such documents may be self-contained or they may be a component of a *compendium* which provides the infrastructure needed to provide access to data and supporting software. These documents, or compendiums, can be processed in a number of different ways. One transformation will be to replace the code with its output -- thereby providing the familiar, but limited, static document.

In this paper we apply these concepts to a seminal paper in bioinformatics, namely *The Molecular Classification of Cancer*, Golub et al (1999). The authors of that paper have generously provided data and other information that have allowed us to largely reproduce their results. Rather than reproduce this paper exactly we demonstrate that such a reproduction is possible and instead concentrate on demonstrating the usefulness of the compendium concept itself.

KEYWORDS: computational science, reproducibility, literate programming

Author Notes: I would like to thank D. Temple Lang, S. Dudoit, P. Tamayo, V. Carey and T. Rossini for many helpful discussions. I would also like to thank two referees for their insight and helpful comments.

Introduction

The publication of scientific results is carried out today in much the same way as it was fifty years ago. Computers rather than typewriters bear the brunt of the composition and the internet has largely replaced the mail as the transport mechanism but, by and large the processes are unchanged. On the other hand, the basic tools of scientific research have changed dramatically and technology has had a deep and lasting impact. In this paper we examine the implications of a new method for publishing results that rely on computation that was proposed by Gentleman and Temple Lang (2003).

We have termed this method of publication reproducible research because one of the goals is to provide readers (and potentially users) with versions of the research which can be explored and where the contributions and results can be reproduced on the reader's own computer. The general approach is reported in Gentleman and Temple Lang (2003) and is based on ideas of literate programming proposed by Knuth (1992) with adaptations to statistical research as proposed by Buckheit and Donoho (1995). This report uses an implementation based on R (Ihaka and Gentleman, 1996) and Sweave (Leisch, 2002).

Gentleman and Temple Lang (2003) refer to the distributable object as a *compendium*. In its most simplistic form a compendium is a collection of software, data and one or more navigable documents. A navigable document is a document that contains markup for both text chunks and code chunks. It is called a *navigable document* because a reader, equipped with the appropriate software tools, can navigate its contents and explore and reproduce the computationally derived content.

Navigable documents are transformed in a variety of different ways to produce outputs and it is the different outputs that are of interest to the readers. One transformation of a navigable document is to evaluate the code chunks (with appropriate software) and to replace them with the output from the evaluation. For example, rather than including a graphic or plot directly into a document the author includes the set of commands that produce the plot. During the transformation of the document the plot is created and included in the output document. Using this method the reader can read the finished document but she can also refer to the untransformed navigable document to determine the exact details of how the plot was constructed and which data were used. In many cases the transformations that are made will rely on specific data or computer code and these resources are stored in, and made available through, the compendium.

Our primary motivation for this research came from attempts to understand the large number of interesting papers related to computational biology. While the authors are, by and large, generous with their data and their time the situation is not satisfactory for them or their audience. The research reported here represents one approach that has potential to allow authors to better express themselves while simultaneously allowing readers to better comprehend the points being made.

Adoption of these mechanisms will have substantial side benefits, such as allowing a user to explore and perturb the original work thereby increasing their comprehension. And for those involved in collaborative research the compendium concept can greatly simplify some aspects. For example, computations carried out by one investigator (possibly at one site) can be easily reproduced, examined and extended by an investigator at another site. There is also substantial benefit to be gained in a single laboratory. In that setting, as post-docs, students and collaborators come and go, compendiums can provide guidance. New researchers are able to quickly and relatively easily determine what the previous investigator had done. Extension, improvement, or simply use will be easier than if no protocol has been used.

In this document we will use the term publication in a very general sense and often the phrase *make available* may be more apt. Examples of publication include the usual form of publication in a scientific journal, it may mean sending a compendium to a specific set of colleagues or the compendium may be internal to the lab group, where the compendium is a method of transferring knowledge from one generation to the next. In this last case, we envisage the situation where a post-doc, scientist or student has finished their employment but their project will form the basis for other initiatives; in this setting publication is really the process of constructing one or more compendiums and leaving them for the next investigator.

We put the ideas proposed in Gentleman and Temple Lang (2003) into practice. The application of these ideas is demonstrated on a particular well known example: the Molecular Classification of Cancer, Golub et al. (1999). This is one of the important papers in the field of bioinformatics and the authors have generously provided data and other information that have allowed the reproduction of their results. We have also relied on Slonim et al. (2000) and Dudoit et al. (2002) for some guidance.

There are two basic points that we would like to make in this article. First that a practical, easy to use set of tools exists for creating compendiums and second that both the author and the reader benefit from the approach. To achieve the first of these goals a portion of the analysis reported in Golub et al. (1999) is implemented using available tools. It would violate copyright laws and would be rather boring for those familiar with that work to replicate it in its entirety. Rather, we demonstrate that such a replication is possible, by reproducing a portion of their analysis.

Achieving the second goal is harder. The author benefits by being able to better describe the analysis since the code segments supplement the written document and make it much easier to reconstruct the analysis and to improve the exposition. I can return to this analysis at any time in the future and still understand it – a feat that would require considerably more effort with some of my other works. To demonstrate a benefit to the reader your help is needed. The reader must examine both the transformed version of this document and the untransformed one. In the untransformed document you will have to do a little work to locate and comprehend the code. But the benefits can be substantial and we hope that you will choose to

DOI: 10.2202/1544-6115.1034

explore both the typeset version and the compendium.

Motivation

These ideas have been guided by a number of pioneering works. Buckheit and Donoho (1995), referring to the work and philosophy of Claerbout state the following principle:

> An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and that complete set of instructions that generated the figures.

It is hard to argue with that sentiment. There are substantial benefits that will come from enabling authors to publish not just an advertisement of their work but rather the work itself. The technology needed to support the publication of the computational science exists. A paradigm that fundamentally shifts publication of computational science from an advertisement of scholarship to the scholarship itself is needed and the research reported here is a step in that direction.

More recently, Green (2003), has drawn direct attention to the inadequacies of the current situation.

> Now that methodology is often so complicated and computationally intensive that the standard dissemination vehicle of the 16-page refereed learned journal paper is no longer adequate.

He goes on to note that,

> Most statistics papers, as published, no longer satisfy the conventional scientific criterion of reproducibility: could a reasonably competent and adequately equipped reader obtain equivalent results if the experiment or analysis were repeated?

We will demonstrate how, a compendium provides the explicit computational details which can be easily comprehended, modified and extended. A competent and adequately equipped reader will easily be able to reproduce the results.

There were many reasons for choosing Golub et al. (1999) to exemplify the principles being proposed here. Golub et al. (1999) is a well written paper, it is highly regarded and the major points made rely heavily on computation. Further, the data are publicly available many of the details of their analysis are reported in Golub et al. (1999), Slonim et al. (2000) and Dudoit et al. (2002). It is a testament to their scholarship that few inquiries were needed to establish the explicit details of the computations. On the other hand, the reader of this paper can explore the untransformed document and should, in principle, need no explicit guidance from

the author. The exact nature of the computations, the order in which they were applied and the data used to produce any graphic, table or statistic can readily be obtained from the compendium.

This paper itself is written in the format being proposed and is provided as a compendium with all data, software, and documents (transformed and untransformed) available for exploration. The reader has a very simple task if they want to recover any of the specific details; they need simply find the appropriate section of the navigable document. Moreover, readers will be able to easily interact with the details of the computations, they will be able to make different choices at different steps in the process (although this may require some programming skill). In the implementation being presented this can be done by altering the code in the navigable document and then transforming to obtain a new output. In the future we envisage an interactive viewer with controls that can be manipulated by the user.

A compendium constitutes reproducible research in the sense that the outputs presented by the author can be reproduced by the reader. It does not, however, constitute an independent implementation. However, the compendium can provide sufficient information that verification of the details of the analytic process is possible. Such a mechanism can help improve the substance of scientific publications by exposing more of the details to scrutiny both by the reviewers and by the audience.

Background and Alternative Approaches

As was noted above the ideas presented here have historical antecedents. We will consider some of them and demonstrate why each of them is incomplete with regard to the specific task that we are addressing. The basic role of a compendium is to provide support, substantiation and reproducibility to the author's work. The reader of any paper is being asked to believe that given the original data and the set of transformations described by the author that the figures and tables of the resultant paper can be obtained. The role of the compendium is to remove all doubt from the reader's mind as to whether the tables and figures can be produced and to provide a complete and algorithmic description of how they were obtained. When provided in this format, the reader is able to verify these claims by reproducing them on their own computer.

Auditing Facilities

For the S language, one of the early papers that touches on some of the issues being raised here is Becker and Chambers (1988). The authors describe their system for *auditing* data analyses. Their task itself is slightly different although not unrelated and the authors make the point that one of the purposes of the audit is to validate published results. However, unless the audit and the data are made publicly available then no real validation is possible. And if both the audit and the data are made

DOI: 10.2202/1544-6115.1034

available then there are strong similarities to what is being proposed here. Except, of course, that our proposal calls for a much tighter integration that provides direct connections between the data, the computations, and the reported tables and graphics.

It is also important at this point to indicate one of modes of failure for any auditing system is for certain calculations or transformations to be carried out in a system other than the one for which auditing is carried out. Examples of such manipulations abound. There is no simple mechanism, for example, to track changes made to data in an Excel spread sheet. Such manipulations break the audit trail and if the output (the published document) is not tightly linked to the supposed set of computations it is difficult to detect such defects in the analysis.

Since we have considered Becker and Chambers (1988) it is worth pointing out that this article itself exemplifies precisely the problem we are trying to solve. The authors are careful competent scientists, but their first code chunk appears to be in error. The S language statement `body<-m[,2]` is missing and without it the third statement in their first code chunk will fail. This error would have been detected immediately in our system yet was apparently missed by theirs. This demonstrates how difficult the task of publishing verifiable computation is and the need for software tools that reduce the complexity for authors. It also suggests that a much tighter integration between the software, data processing, and the finished paper is needed.

An auditing system performs a slightly different, but no less valuable, role than that of the compendiums we are proposing. And, the ability to capture code used to perform an analysis into a specified code chunk, in a navigable document, could be a valuable tool. However, the lack of tight integration with auxiliary software and the data means that auditing facilities can at best perform only part of the role of a compendium.

Data and Script Repositories

Some of the current practices that attempt to address the situation are for journals (and authors) to make their data publicly available and in many cases to provide the scripts that they used to perform their analyses. Now the scripts could in fact be produced by the auditing facilities described above, but they do not need to be.

Such solutions fall short of the required levels of reproducibility. One of their short comings is that each author selects a different set of conventions and strategies and the user is faced with the complexity of unraveling the sequence of steps that led to the different pages and figures. The adoption of a set of widely used conventions would make the situation better, but in a sense that is the equivalent of our proposal here. At least on one level, the compendium is merely a set of conventions. But, we need tight integration of the data, the transformations and the outputs and this is not achieved by providing data on a web site together with the scripts which are purported to carry out the reported analysis. An example of some of the difficulties

that can be encountered is reported in Baggerly et al. (2004).

Authoring Tools

The importance of easy to use authoring tools cannot be overemphasized. In our prototype we propose using the Sweave system. In large part because it is integrated with R and there are many tools in R to help with some of the basic manipulations (it is worth noting that these tools were largely developed with this particular application in mind).

Sweave itself has a historical precedent – the REVWEB system (Lang and Wolf, 1997–2001). The system is apparently quite advanced and there exists software for use with the WinEdt (`www.winedt.com`) editor for creating what they term *revivable* documents. Much of the documentation and discussion is in German, thereby limiting access to the important ideas. The substance is very similar to that of the Sweave system.

REVWEB has a mechanism that allows users to step through the code chunks that were recorded. In our system this functionality is provided by the `vExplorer` function in the *tkWidgets* package from the Bioconductor Project.

The two systems, REVWEB and Sweave, are suitable for producing navigable documents from which versions of the research can be produced. But neither has a model for controlling the data, for ensuring that all auxiliary software is available, for versioning or for distribution. Now granted, all such issues can be dealt with in different ways and we anticipate adopting any such innovations. But currently neither of these systems would be a good replacement for the compendium concept - although both could play a substantial role within that paradigm.

Authors using the Sweave system tend to rely on the Emacs Speaks Statistics system Rossini et al. (2004). Rossini (2001) discusses the related concept of literate statistical analysis using many of the same tools, however the emphasis is different.

Other related work of Weck (1997) considers a document-centric view of publishing mathematical works. The author raises a number of points that would need to be addressed in a complete solution to the problem. However, the problems of tight integration with the data and the outputs exists here as well and no functioning system appears to be available.

The work of Sawitzki (Sawitzki, 2002) is also in a similar vein. The notion of a document with dynamic content is explored. However, the emphasis there is on real time interaction. The document is live in the sense that the user can adjust certain parameters and see the figures change in real-time. While clearly an interesting perspective such documents could quite naturally fit within the compendium concept. The compendium would provide them with a natural location for data, auxiliary code, documentation as well as tools for versioning and distribution.

DOI: 10.2202/1544-6115.1034

Methods

For the purposes of this paper a compendium is an R package together with an Sweave document. The package provides specific locations for functions, data and documentation. These are described in the R Extensions Manual (R Development Core Team) which is available with every distribution of R. The R package satisfies our requirements for associating data and software with the navigable documents. This document and the research it embodies is provided as a compendium and therefore, as an R package. The package is titled *GolubRR*, named after the first author of Golub et al. (1999), with the RR suffix conveying both the notion of reproducible research and the reliance on the R language. *GolubRR* contains code that implements the functions needed, manual pages for all supplied functions, data and a navigable document which can be processed in a number of ways. This document that you are reading is one of the transformations of that navigable document.

While our proposed prototype comes in the form of an R package it is important that we distinguish the compendium concept from that of a software package or module. We have adopted R's packaging mechanism because it gave us the structure that we needed (for containing code, data and navigable documents) together with tools to manipulate that document, to provide version information, distribution and testing. But the purpose of a compendium is different from that of a software package. It is not intended to be reusable on a variety of inputs nor does it provide a coherent set of software tools to carry out a specific and well-defined set of operations. A compendium provides support for the claims that its author has made about their processing of the data and about the reproducibility of that processing. Compendiums are designed to address a single specific question and that distinguishes them substantially from software packages – it is only the medium used for management that is the same.

We further note, emphatically, that the compendium concept does not rely on R but is completely general and language neutral. It does require the implementation of a certain amount of software infrastructure and currently only the R language supports the production and use of compendiums. However, the compendium concept could easily be extended to include other languages such as Perl and Python. The concepts are general, the implementations must be specific. A prototype of a navigation system for Sweave documents is available from the Bioconductor project in the *tkWidgets* package as `vExplorer`.

You, as a reader have several choices in how you would like to interact with the compendium. You can simply read this document, which is largely complete. You can obtain the compendium (in this case from `http://www.bioconductor.org/Docs/Papers/2003/Compendium`) and save it on your computer. There you can explore the different folders and files that it contains. You can obtain R, install the compendium as a package in R, start R and use it to explore the compendium using the tools mentioned above. To examine the code chunks you will need to either

open the navigable document in an editor or use some of the functionality available in R.

The text you are reading is contained in an Sweave document named `Golub.Rnw` within the compendium. This document contains an alternating sequence of text (called *text chunks*) and computer code (called *code chunks*). The text describes what procedures and methods are to be performed on the data and the code is sequence of commands needed to carry out those procedures. When the document is processed, *woven*, the code is evaluated, in the appropriate language, and the outputs are placed into the text of the finished document. However, it is important to note that the compendium is a unit. One cannot expect to extract components (even if they look like familiar LaTeX documents) and have them function without the supporting infrastructure. The outputs and transformations (such as PDF documents) are distributable.

Code chunks do not need to be explicitly printed in the finished document. They are often *hidden* since the reader will not want to see the explicit details but rather some transformation of them. For example, when filtering genes, the author might explain in the text how that was done and may or may not want to explicitly show the code. But in either case, the code exists and is contained in the untransformed document. It can therefore be examined by the reader. Some of the outputs, such as the number of interesting genes, may be placed directly in the text of the output document.

The text below is a code chunk that demonstrates how the data were Windsorized (the low values were moved up to 100 and the high values down to 16,000) during the data cleaning process described in Golub et al. (1999). Below is the code chunk that is needed to carry this out.

```
<<windsorize, results=hide>>=
X <- exprs(golubTrain)
Wlow <- 100
Whigh <- 16000
X[X<Wlow] <- Wlow
X[X>Whigh] <- Whigh
@
```

The first line, `<<windsorize, results=hide>>=` indicates the start of a code chunk. The first argument, `windsorize` is a label for that code chunk and is useful when debugging or performing other operations on the document. The second argument for the code chunk is `results=hide`. This command indicates that when the document is processed the output should not be visible in the transformed document. The code chunk consists of five statements, in the R language, followed by a line with an at symbol, `@`, located in the first position. All subsequent lines will be treated as text chunks until another code chunk is encountered. Many more details of the syntax and semantics of the Sweave system are available in the appropriate documentation provided with R and in Leisch (2002).

DOI: 10.2202/1544-6115.1034

Authors may also want to include the results of some computations within the text chunks. For example the author might want to report the values that were used for windsorizing the data. The following construct may be used, at any point following the definitions of the variables `Wlow` and `Whigh`.

```
The data were Windsorized with lower values \Sexpr{Wlow} and upper
value \Sexpr{Whigh}.
```

When the document is processed the markup `\Sexpr{Wlow}` will be replaced by the value of the R expression contained in the call to `\Sexpr`; which in this case would be the value of `Wlow`. In this case no code chunk is required.

Producing figures in the Sweave model is also quite straightforward. The following code snippet is used to reproduce part of Figure 3 in Golub et al. (1999).

```
\begin{figure}[htbp]
\begin{center}
<<imageplot, fig=TRUE, echo=FALSE>>=
  image(1:38, 1:50, t(exprs(gTrPS)), col=dChip.colors(10),
       main="")
@
\caption{Recreation of Figure 3B from \citet{Golub99}.}
\end{center}
\end{figure}
```

In this example we intermingle the usual LaTeX commands used to produce figures with Sweave markup. At the time that this segment appears all necessary variables and functions (e.g. `gTrPS` and `dChip.colors`) must be defined.

The evaluation model for these documents is linear. Any variable or function created in a code chunk is available for use after the point of its creation. As research in this area progresses it will become important to consider different models for controlling the scoping of variables within the document. Both Weck (1997) and Sawitzki (2002) raise this issue and it is of some importance. In the current implementation variable and function scope is global. However, one can easily imagine cases where restricting scope to specific code chunks would be beneficial.

The author of the navigable document has a number of options for controlling the output produced by any code chunk. We reiterate the fact that all details exist in the untransformed document, whether or not they are presented in the finished document, and the reader has access to them. The reader can determine what values were used and exactly when in the data analytic process a step was carried out. The existence of the code and the sequential nature of an Sweave document provide the necessary details and typically further explanation is not required.

Another way in which an Sweave document can be processed is by tangling. When an Sweave document is tangled the code chunks are extracted. This can be done either to a file or into R itself. This process separates the processing from the

narrative and can be quite helpful to those who want to examine the sequential data processing steps.

We have exposed more code and raw results in this document than would normally be the case. The reason for this is to convince you, the reader, that there is no artifice involved. All computations are being carried out during document processing. If you have downloaded the compendium you can, of course, check that for yourself. In normal use the code would be suppressed and the outputs would be confined to tables, figures and in-line values.

The Details

In this section we provide specific examples and code based on the analysis reported in (Golub et al., 1999). To avoid the rather constant citation of this work we use *Golub* to refer to the paper in this section. Our analysis was also aided by the details reported in (Dudoit et al., 2002) and Slonim et al. (2000) regarding their understanding of the analysis. The analysis is intentionally incomplete; the goal here is not to reproduce *Golub* but rather to convince the reader that that paper could have been authored using the tools being described here. Any author contemplating a new paper would simply use this system, as we do, to produce their work in the form of a compendium.

The data, as provided at `http://www.genome.wi.mit.edu/MPR` in January 2002, were collected and assembled into an R package. This was done to make it easier for readers to access the data. The package is named *golubEsets*. This package and other software will need to be assembled by the reader of this document if they want to interact with it. Much of this process should be automated and the user should only need to obtain this compendium and load it into an appropriate version of R. They will subsequently be queried regarding the downloading and installation of the other required software libraries.

The first code chunk loads the necessary software packages into R. The code chunk is labeled `setup` and its output is suppressed. The reader does not need to be distracted by these details in the processed document. Including these steps in the untransformed document is essential for reproducibility.

Preprocessing

The analyses reported by *Golub* involved some preprocessing of the data. In all microarray experiments it is important to filter, or remove, probes that are not informative. A probe is non-informative when it shows little variation in expression across the samples being considered. This can happen if the gene is not expressed or if it is expressed but the levels are constant in all samples.

While the exact processing steps are not reported in *Golub* the data were Windsorized to a lower value of 100 and an upper value of 16,000. Next the minimum

DOI: 10.2202/1544-6115.1034

and maximum expression values for that probe, across samples, was determined. A gene was deemed non-informative (and hence excluded) if the ratio of the minimum to the maximum is less than 5 or the difference between the minimum and the maximum is less than 500 (Tamayo, 2003). We have incorporated this processing in the function `mmfilt` which makes use of functionality incorporated in the Bioconductor package *genefilter*.

At this point we begin processing the data. The code chunk presented previously is evaluated here and the filtering and gene selection process is carried out. The output below comes from evaluating the expressions in R. A reader could easily edit the untransformed document to change these criteria and examine what happens if a different set of conditions were used for gene selection.

```
> X <- exprs(golubTrain)
> Wlow <- 100
> Whigh <- 16000
> X[X < Wlow] <- Wlow
> X[X > Whigh] <- Whigh
```

The details of the filtering process are suppressed but the filtering process has selected 3051 genes that seem worthy (according to the criteria imposed) of further investigation. The value printed in the previous sentence (it should be 3051) was computed and inserted into the text using the `Sexpr` command. Changing the processing instructions would change the value reported in that sentence.

The interested reader will find the software instructions for carrying out these computations in the untransformed document. They are in the code chunks labeled `windsorize` and `filter`.

The next step is to produce a subset of the data that will be used in our subsequent computations. This code chunk is displayed below. As you can see the commands are printed out as are the results (if any). We first subset the expression data and then check to see if we have obtained the *correct* number of probes. The command `dim(X)` asks R to print out the dimensions of the matrix `X`; hopefully we see that this is 3051.

```
> X <- X[sub, ]
> dim(X)

[1] 3051   38

> golubTrainSub <- golubTrain[sub, ]
> golubTrainSub@exprs <- X
```

The data have been stored in an `exprSet` object. An `exprSet` is a data structure designed to hold microarray data. More details can be found through the on-line help system in R and in the *Biobase* package. Next the test set is reduced to the

same set of genes as selected for the training set. The code to do this is contained in the code chunk labeled `testset`, but is not displayed here.

In this analysis the genes were selected according to their behavior in the training set. If the same selection criteria were applied to the test set a different set of genes would be selected. This is a point where an interested reader could benefit from the compendium and simply determine which genes would be selected from the test set if the same criterion were applied. This could lead to a different decision about which genes to use in the remainder of the analysis. Readers can explore various scenarios using the compendium for guidance.

Neighborhood Analysis

The first quantitative analysis reported was called *neighborhood analysis*. The basic idea is to determine if a set of genes had a correspondence or association with a particular grouping variable, such as the ALL–AML classification. The test statistic used was reported in Note 16 as:

$$P(g, c) = \frac{\mu_1(g) - \mu_2(g)}{\sigma_1(g) + \sigma_2(g)}, \tag{1}$$

where μ_i denotes the mean level of log expression in sample i and σ_i denotes the standard deviations of the expression levels in sample i. The two groups (labeled 1 and 2) are determined by the supplied variable c. Statisticians will notice a similarity to the two-sample t-test (except for the denominator). Here, a reader might want to replace this measure of *correlation* with another choice. We will use the terminology *correlation* here since that is what was used in *Golub* but emphasize that this usage does not reflect the usual statistical interpretation.

The code for this is quite simple to write in R. It is included in the *GolubRR* package as the function `P` and is an example of the need to include auxiliary software in the compendium. An idealized expression pattern can be created using the training data. In their paper *Golub* used a variable that is one when the sample is from the ALL group and zero otherwise. We set the R variable `c` to have these values in the next code chunk.

```
> c <- ifelse(golubTrain$ALL == "ALL", 1, 0)
```

pattern. The function P is then applied to the data data in `golubTrainSub` to obtain the *correlations*.

The fact that roughly 1100 genes were more highly correlated with the ALL-AML classification than would be expected by chance is reported at the top of page 532. Further details are given in their Figure 2 and in their supplemental Notes 16 and 17. *Golub* report using random permutations of the idealized expression pattern to determine whether the number of genes correlated with the idealized expression pattern was larger than one might expect by chance. They reported using 400 permutations to assess the significance of the results.

DOI: 10.2202/1544-6115.1034

At this point the author of the compendium has some choices to make. By including all of the computed permutations the size of the compendium, which is already fairly large, would be about 400 times larger. An alternative would be to provide sufficient documentation for the reader to reconstruct the simulations. That might simply involve a description of the random number generator and the seed used to start it or perhaps a complete implementation of the random number generator would be supplied in the compendium. The reader would then have to create the permutation data sets and using them carry out the calculations reported. A third alternative for the author of the compendium would be to make the permuted data sets available for download. In the first and third situations the amount of data that the reader needs to obtain is increased substantially while in the second the processing time may substantially increase. The nature of the trade-offs would probably need to be evaluated in each specific situation by the author of the compendium.

For the purposes of this report we supply a function, `permCor` that can be used to perform the computations and have made no further investigations into these aspects. If you have R available for exploring the compendium you can find out more about this function by either typing its name at the R prompt (in which case you will see the code) or by typing `?permCor` at the R prompt to get the manual page for `permCor`.

Class Prediction

Starting on page 532, *Golub* begin a discussion of the procedures they used for class prediction. Details are given in the caption for their Figure 1, and in their Notes 19 and 20, part of which is repeated next:

> The prediction of a new sample is based on "weighted votes" of a set of informative genes. Each such gene g_i votes for either AML or ALL, depending on whether its expression level x_i in the sample is closer to μ_{AML} or μ_{ALL}...

The magnitude of a vote is $w_i v_i$ where w_i is a weighting factor that depends on how well correlated gene g_i is with the idealized expression. In Note 19 w_i is denoted a_i (or a_g) and is stated to be simply $P(g, c)$. Whereas, v_i is given by,

$$v_i = |x_i - \frac{\mu_{AML} + \mu_{ALL}}{2}|.$$

The total votes for each of the two classes are tallied to yield V_{AML} and V_{ALL}. Then the prediction strength (PS) is computed as,

$$PS = \frac{|V_{AML} - V_{ALL}|}{V_{AML} + V_{ALL}}.$$

Each sample is labeled with the class that corresponds to the larger of V_{AML} and V_{ALL}, provided that the prediction strength, PS, is larger than some prespecified limit; *Golub* chose to use 0.3 as their prespecified limit.

The algorithm is quite explicit and requires only a determination of how to select the *informative genes*. *Golub* chose to use 50 informative genes. These were the 25 genes most highly correlated with the idealized expression value (the 25 nearest to one and the 25 nearest to minus one). The code needed to find these best 25 genes is contained in the code chunk labeled `getBest25`.

The 50 genes selected using this criterion did not precisely coincide with those reported in *Golub*. There were three genes that were selected by the methods described here that were not in the lists presented by *Golub*. The disagreements were minor and likely due to rounding or similar minor differences or a misunderstanding on our part. Since those reported by *Golub* were used for all their subsequent analyses and since the goal here is to reproduce their published results we provide those probes reported by *Golub* as data sets named `amlgenes` and `allgenes`. In most of the subsequent analyses reported here we use these data to be comparable. But before leaving this, we consider briefly how a user might choose to study the genes found using our interpretation of the method used by *Golub*.

In the code chunk below we first read in the data sets for genes as determined by *Golub* and count how many are in common with the lists we selected (ours are in variables named `AML25` and `ALL25`). We then determined where those selected by *Golub* fall in our ordered list of genes. The values can be found by exploring the variables `wh.aml` and `wh.all`. The genes selected in *Golub* were very close (their ranks were just outside the set selected here, suggesting that the difference is likely to be just in how ties or near ties were handled). There were 23 in common for AML and 24 for ALL. Finally the symbols for the three genes we selected that they did not, are printed by the following code chunk.

```
> data(hu6800SYMBOL)
> unlist(mget(wh.leftout, hu6800SYMBOL))

     J05243_at      M11147_at M21551_rna1_at
      "SPTAN1"          "FTL"          "NMB"
```

Returning to the main analysis, Dudoit et al. (2002) report that some further processing of the data occurred at this point. The data were log transformed and then standardization was performed. For each gene *Golub* subtracted the mean and divided by the standard deviation (mean and standard deviation were taken across samples). These details were not contained in the original paper, but reproducibility depends on using the same transformation at the same point as was done in the original analysis. We note that such details would easily be available from any compendium-like version of the analysis. The means and standard deviations are saved since those from the training set were also used to standardize the test set. The code is contained in a code chunk labeled `standardize`.

To compute the prediction strength we use two functions. These are supplied in the compendium and are documented more fully there. The first is called `votes`. It

DOI: 10.2202/1544-6115.1034

computes the matrix (samples by genes) of votes as well as the average of the two means, and which of the two means is closer to the observed expression value for that gene and sample.

The function to compute prediction strength is called `PS`. This function takes the the class vector and computes both the group assignment and the vote.

```
> gTr.votes <- votes(gTrPS, c)
> names(gTr.votes)

[1] "closer" "mns"    "wts"    "vote"

> C <- ifelse(c == 1, "ALL", "AML")
> vsTr <- vstruct(gTrPS, C)
> PSsamp1 <- dovote(exprs(gTrPS)[, 1], vsTr)
> allPS.train <- vector("list", length = 38)
> for (i in 1:38) allPS.train[[i]] <- dovote(exprs(gTrPS[, i]),
+     vsTr)
```

The cross-validation component is implemented using `PScv` and the testing component is implemented using `PStest`. These can be applied to the data to obtain the values used in Figure 3 (A) of *Golub*. We can then use the training set to provide predictions for the test set.

The code to produce the table comparing the predicted classes to the observed classes is given below. We indicate to the document processor that we do not want the commands to be echoed and that the output from the command will be valid LaTeX. This ensures that there will be no markup around the output that would interfere with the usual LaTeX processing of the document. In producing the table we rely on the R package *xtable*. The set of commands are presented next, to demonstrate the relative simplicity with which we can produce tables in the output of our document that are based on computations made during the evaluation of code chunks found earlier in the document.

```
<<PStable, echo=FALSE, results=tex>>=

y<-unclass(table( tsPred, gTePS$ALL))
dny <- dimnames(y)
dimnames(y) <- list(paste(dny[[1]], "Obs"),
                    paste(dny[[2]], "Pred"))

xtable.matrix(y, caption="Predicted versus Observed",
        label="Ta:PreObs")
@
```

The output of these commands produces Table 1. As noted, the values in the table are recomputed each time the document is processed. They are not obtained by cutting and pasting and hence are not subject to the sorts of errors that that style of document construction is prone to. They are of course subject to other sorts of errors.

	ALL Pred	AML Pred
ALL Obs	19.00	1.00
AML Obs	1.00	13.00

Table 1: Predicted versus Observed

Next we look at a table of the results. The number of samples where the prediction strength exceeded 0.3 was 29. And the table of predicted class versus observed class is given in Table 2).

	ALL	AML
ALL	19.00	0.00
AML	0.00	10.00

Table 2: Predicted versus Observed (with high prediction strength)

We can produce the false-color image, replicating Figure 3 B of *Golub* using the `image` function in R. We could in fact reproduce the plot almost identically, but that would require some additional amount of effort that would only be warranted in a production run. We have, however, provided both the false-color image as shown in *Golub* and beside it a *heatmap* of the sort that has become quite popular. The reader may want to further examine the groupings suggested by the dendrograms (both columns and rows).

Discussion

A compendium constitutes reproducible research in the sense that the outputs presented by the author can be reproduced by the reader. It does not, however, constitute an independent implementation. That would require a second, independent experiment and analysis which would result in a second independent compendium. However, it provides sufficient information to enable verification of the details of the scientific results being reported.

A compendium enables new and different levels of collaboration on scientific work based on computation. Each of the authors has available to them complete details and complete data, during the authoring process. It is easier to see, understand and possibly extend the work of your collaborators. A compendium helps to ensure

DOI: 10.2202/1544-6115.1034

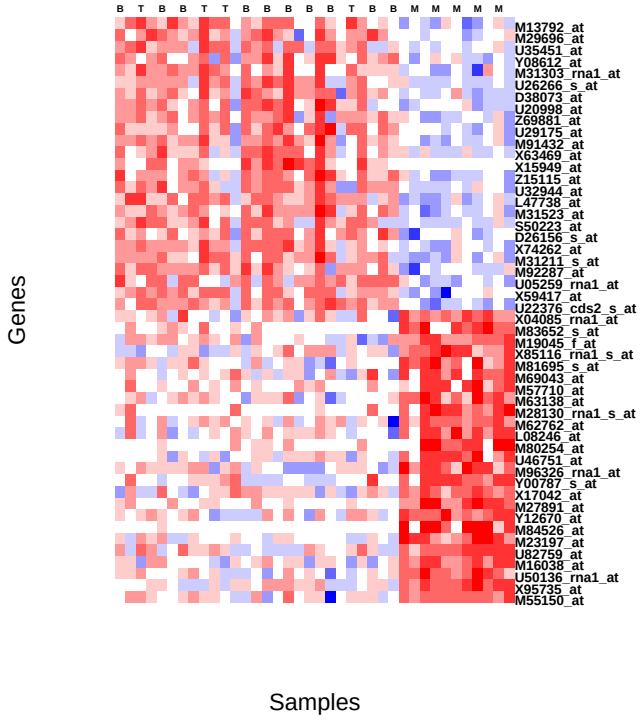

Figure 1: Recreation of Figure 3B from *Golub*.

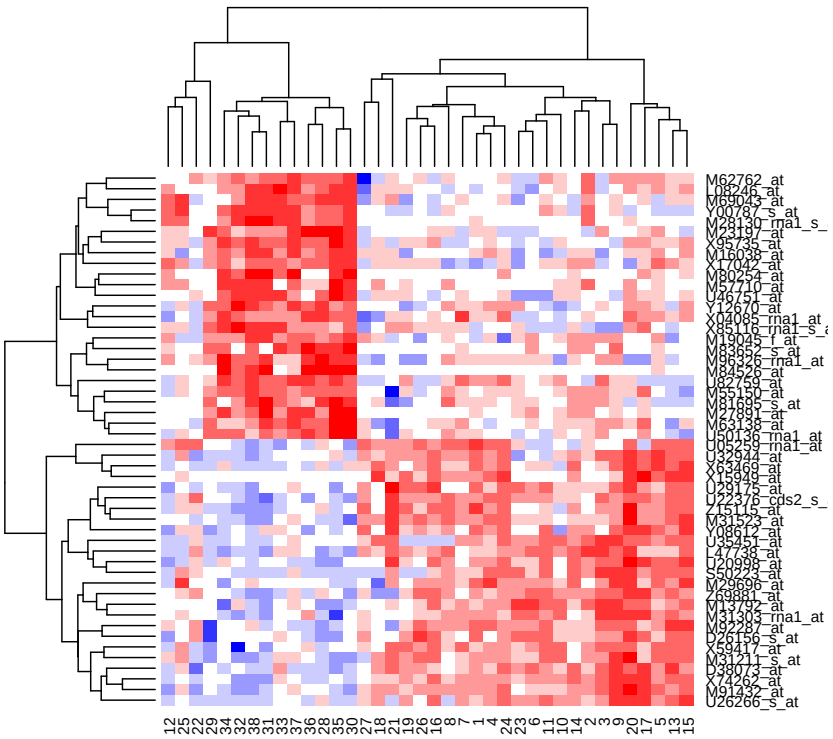

Figure 2: The data from Figure 3B as a heatmap.

DOI: 10.2202/1544-6115.1034

continuity and transparency of computational work within a laboratory or group. When a post-doc, student or scientist leaves the group, their work is still accessible to others and generally the time required for someone new to grasp and extend that work will be shorter if a compendium is available.

Finally, there is of course the notion of general publication, or publication in a scientific journal. Again, we argue that compendiums merely increase the options available for both publication and refereeing. In neither case is the compendium essential but, if it is available it can make it much easier for the reader to comprehend the computations involved. We have heard arguments made about the problems of finding referees for these compendiums and can only answer them by saying that if the results being published are computationally based then it is essential that they be refereed by individuals that are computationally literate. Having access to, and knowledge of the specific details of the computations provides invaluable information to a referee or critical reader.

In most areas of research the scientific process is one of iterative refinements. A hypothesis is formed and experiments or theorems devised that help to refine that hypothesis. Works based on scientific computation have not generally benefited from this approach. Since the works themselves (i.e. the explicit computations) are seldom explicitly published it is difficult for others to refine or improve them. The compendium has the potential to change this. A compendium provides the work in a format that is conducive to extension and refinement.

In situations where the research being reported relies mainly on simulations or other *in silico* experiments then the compendium can be largely independent of the original data. If the random number generators are included and other constraints met then the user will have access to the entire experimental process. In other situations, such as bioinformatics or computational biology there must be some point at which the data are captured electronically. There is no way that the compendium concept can provide validity prior to that point. But rather compendiums provide a mechanism for comprehending and exploring the *reported* data and the *reported* analyses of it.

While the examples and discussion presented are based on a set of prototypes that have been written for the R language we must once again stress the fact that the concepts and paradigm are completely general and language neutral. All aspects that we have considered could be made available and implemented in any one of the many computer languages now popular, e.g. Java, Perl or Python. Of course a great deal of software infrastructure will be needed, but the results speak for themselves. We must make computational science more accessible to the forces of openness, reproducibility and iterative refinement.

Appendix A: R Packages Used

The following provides a description and some details of the R packages used to produce this document.

annotate R. Gentleman, *Using R environments for annotation.*

Biobase R. Gentleman and V. Carey, *Bioconductor Fundamentals.*

genefilter R. Gentleman and V Carey, *Some basic functions for filtering genes.*

geneplotter R. Gentleman, *Some basic functions for plotting genomic data.*

golubEsets T. Golub, *A representation of the publicly available Golub data.*

GolubRR R. Gentleman, *A package demonstrating the benefits of reproducible research. With a reanalysis of Golub et al 1999.*

hu6800 J. Zhang, *Annotation data file for hu6800 assembled using data from public data repositories..*

tkWidgets J. Zhang, *R based tk Widgets.*

xtable D. Dahl, *Coerce data to LaTeX and HTML tables.*

Appendix B: Creating a compendium

The creation of one of the proposed compendiums is quite straight forward. The first step is the creation of an R package. Once that is done, the author creates a folder in that package named `inst` and within the `inst` folder a second folder named `doc`. Within the `doc` folder they can create all of the documents that they would like following the directions given in the Sweave manual.

The Anatomy of an R Package

An R package is a set of files and folders, some of which have specific names. Many more details are given in the R Extension Manual (R Development Core Team) which is the definitive reference for packages and many aspects of R. We list the most important set of these below:

- `DESCRIPTION`: a file in the main folder that provides various declarative statements about the package. These include its name, the version number, the maintainer and any dependencies on other packages (as well as several other things).
- `R`: a folder that contains all of the R code for the package.

DOI: 10.2202/1544-6115.1034

- `man`: a folder that contains the manual pages for the functions in the package.
- `src`: a folder that contains the source code for any foreign languages (such as C or FORTRAN) that will be used.
- `inst`: a folder that contains components that will be made available at install time.
- `inst/doc`: a folder, within the `inst` folder that contains all navigable documents.

The authoring cycle begins by creating the structure of a package, often using the R function `package.skeleton`. They then create the `inst` and `doc` folders and begin filling in the different components. If they have special R functions that they will use then these should be put in the `R` folder and documented. The data that the compendium is using should be put into the `data` folder and it to should be documented appropriately. Manual pages are stored in a special R language markup that is also described in the R Extension Manual and their creation is often facilitated by the use of the function `prompt`.

Once the author is comfortable that their document is ready they should engage in unit testing and checking. The R system has a sophisticated, although sometimes cryptic, software verification system. Issues such as consistency between the code (in the `R` folder) and the documentation (in the `man` folder) is evaluated. At the same time all examples that have been provided are run as are all navigable documents in the `inst\doc` folder and any problems are reported. The author should fix the defects and run the checking system until no errors are reported.

References

Keith A. Baggerly, Jeffrey S. Morris, and Kevin R. Coombes. Reproducibility of seldi-tof protein patterns in serum: comparing datasets from different experiments. *Bioinformatics*, 20:777–85, 2004.

R. A. Becker and J. M. Chambers. Auditing of data analyses. *SIAM Journal on Scientific and Statistical Computing*, 9:747–760, 1988.

J. Buckheit and D. L. Donoho. Wavelab and reproducible research. In A. Antoniadis, editor, *Wavelets and Statistics*. Springer-Verlag, 1995.

S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457):77–87, 2002.

R. Gentleman and D. Temple Lang. Statistical analyses and reproducible research. 2003.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M.L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

Peter J. Green. Diversities of gifts, but the same spirit. *The Statistician*, pages 423–438, 2003.

R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.

D. Knuth. *Literate Programming*. Center for the Study of Language and Information, Stanford, California, 1992.

Lorenz Lang and Hans Peter Wolf. The REVWEB manual for S-Plus in windows, 1997–2001.

Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physika Verlag, Heidelberg, Germany, 2002. URL `http://www.ci.tuwien.ac.at/~leisch/Sweave`. ISBN 3-7908-1517-9.

R Development Core Team. *Writing R Extensions*. R Foundation, Vienna, Austria, 1999.

A. Rossini. Literate statistical analysis. In K. Hornik and F. Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, March 15-17, 2002.* http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings, 2001.

A. J. Rossini, Richard M. Heiberger, Rodney A. Sparapani, Martin Maechler, and Kurt Horniki. Emacs speaks statistics: A multiplatform, multipackage development environment for statistical analysis. *Journal of Computational and Graphical Statistics*, 13:247–261, 2004.

Gunther Sawitzki. Keeping statistics alive in documents. *Computational Statistics*, 17:65–88, 2002.

Donna K. Slonim, Pablo Tamayo, Jill P. Mesirov, Todd R. Golub, and Eric S. Langer. Class prediction and discovery using gene expression data. In *RECOMB, Tokyo, Japan*, pages 575–580. ACM 2000 1-58113-186-0/00/04, 2000. URL `http://www.ci.tuwien.ac.at/~leisch/Sweave`.

P. Tamayo. Personal communication. 2003.

DOI: 10.2202/1544-6115.1034

Wolfgang Weck. Document-centered computing: Compound document editors as user interfaces, 1997. URL citeseer.ist.psu.edu/weck97documentcentered.html.