Building-in quality rather than assessing quality afterwards: A technological solution to ensuring computational accuracy in learning materials

Abstract

Quality encompasses a very broad range of ideas in learning materials, yet the accuracy of the content is often overlooked as a measure of quality. Various aspects of accuracy are briefly considered, and the issue of computational accuracy is then considered further. When learning materials are produced containing the results of mathematical computations, accuracy is essential: but how can the results of these computations be known to be correct? A solution is to embed the *instructions* for performing the calculations in the materials, and let the computer calculate the result and place it in the text. In this way, quality is built into the learning materials by design, not evaluated after the event. This is all accomplished using the ideas of literate programming, applied to the learning materials context. A small example demonstrates how remarkably easy the ideas are to apply in practice using the appropriate technology. Given the technology is available and is easy to use, it would

seem imperative the approach discussed is adopted to improve quality in learning materials containing computational results.

Introduction

Pressure exists everywhere for universities to incorporate quality into every component of their operations, including management, operations, assessment, technology, administration, community involvement, teaching and research. Entire journals are devoted to the study of various components of quality in the higher education context, universities are judged by academics and students alike on the perceived measures of quality even to the extent of ranking universities (e.g., O'Leary et al., 2007), and Universities are often even funded on the basis of their performance on certain quality indicators (for example, in the Australian context see Office of Legislative Drafting and Publishing, 2006). Obvious questions emerge, such as "How is quality defined?" and "How is quality measured?" Lee and Green (1993) give a thorough yet broad discussion of various definitions of quality in the higher education context. At the other extreme, Tanur (1982) states most educators would agree that "a minimum criterion of quality in education is that students be given no false information". As a referee pointed out, sometimes information is given in the belief it is true and is later found to be false. In this vein, perhaps the minimum criterion spoken of by Tanur is that students been given no false information knowingly.

The onus for producing quality outcomes for the university ultimately rests on the people within the university, including academics. For the academic, the preparation

of quality courses, and consequently the related learning materials (such as assignment solutions, study books, tutorial solutions, either on-line or hardcopies), is non-trivial and is usually very time-consuming,; consequently, computational tools to assist in the process are highly regarded. The difficulties arise partly because there are so many components to the job description of an academic, but also because of the many facets and definitions of quality to consider; as examples, consider O'Leary et al. (2007); Ottewill and Macfarlane (2004) list six components of quality in a UK context; Castle and Kelly (2004) discuss various issues of quality in the Australian offshore context; Hosie and Schibeci (2005) give tables of topics to consider regarding quality learning materials. Some topics under which quality is assessed in the above references include: accessibility; appropriate use of media; clear goals; currency of information; ease of use; flexibility; inclusivity; mode of delivery; organisation; pedagogy; pluracy (variety of teaching media); student interaction; style; and support. Surprisingly absent is the *accuracy* of the content

Presumably accuracy of content is either assumed to be correct; is considered too difficult to assess; or can only be judged by human comprehension of the text. But content should not—and cannot—be *assumed* to be correct; consider the number of errata associated with published books and journal articles.

Closer examination of such errata show that content itself may be inaccurate in different ways. As an example, the errata for a text I used recently (Bolstad, 2004) listed errors in mathematical equations, numerical answers, cross-references, in-text citations, references, grammar, graphics, computer instructions, entries in tables, formatting of tables, exercises, and answers to exercises (http://www.stats.waikato.ac.nz/publications/bolstad/IntroductionToBayesianStatistics/students/ accessed 27 Nov 2006).

This paper focuses on the quality of learning materials, and particularly on the quality of the content where it pertains to computational results. Methods for maintaining quality in the results of computations are given. To place in an appropriate framework, the various aspects of quality of content are discussed next.

Aspects of quality

As discussed, others have debated the definition of quality in the higher education context. Here "content" itself is separated into various components, each of which can fall under the scrutiny of "quality".

The *format and presentation of the content* is crucial. For example, graphics must be accurate, but also must be correctly chosen and well prepared (Tufte, 1983). Likewise, tables must present the information appropriately without misleading (Tufte, 1983). Even the typography and placement of text and the organisational structure of the document contribute to overall quality of the content (Priestly, 1991).

The *logic of the content* is crucial. Arguments and mathematical proofs should be coherent.

In this paper, the *accuracy of the content* only is the focus. Accuracy of content itself encompasses many components. Accuracy of content requires instructions (for example, to make a compound in a laboratory; to achieve a task at a computer; to conduct a survey, to access information; to conduct an experiment) to be comprehensive, unambiguous, clear and generally applicable; formulae to be correct and correctly formatted; factual information to be correct and correctly cited when relevant; terminology and definitions to be clear, correct and consistent; and cross-referencing (including indexes and tables of contents) correct and accurate. Many

facets may need to be considered in each components; for example, references must be consistently formatted, the reference list comprehensive and accurate, and each reference must be correct, relevant and appropriate. The first two components are easily assessed, but the third is much harder.

An issue related to factual errors is noted by Hand *et al*. (1994). Often, teachers use artificial data or scenarios in exercises and problems, which may be unrealistic and misleading. Artificial data or scenarios that mislead students must also be considered a failure of content. Hand et al. (1994) uses this argument as a justification for their book of real data sets; interestingly, their Dataset 49 is artificial (the famous correlation data of Anscombe, 1973), and serves as a counter-example that real data is always better than artificial data.

Quality embeds itself deeper also; consider the accuracy of the procedures or equipment used to generate the presented factual information, such as the misuse of *statistical* procedures; for example, see Glantz (1980) and references therein.

The component of quality in content in this article concerns the accuracy of the *computational results*, discussed further in the next section. An unfortunate example appears in the table on page 20 of Student (1908), the seminal paper introducing the ubiquitous *t*-test.

Computational accuracy

Accuracy of computational results is the main focus of this paper. Conceptually, this is clearly a task ideally suited to a computer: computers excel at quickly performing

many calculations. The question is how this can be easily accomplished in practice, especially when the content contains large amounts of computational results. Because of the assistance proffered by computers, the quality of computational results should be one of the easiest aspects of quality to ensure. For example, it seems difficult to comprehend any automated way of assessing the quality of the various types of factual information.

Two ways exist to assess any aspect of quality: The quality can be assessed after the event, or it can be built into the system. The former is the traditional method: The materials are produced and proof-read to determine the quality of the result. Because no quality is built in to the system, every aspect of quality must be checked by people, which is difficult and time-consuming since the many and varied aspects of quality must all be considered. Building quality into the system means the human proof-reader can concentrate on those aspects of quality that remain to be assessed.

Building the quality into the materials often uses technology. Obviously, no learning materials—or any other materials—can be the exclusive domain of technology. For example, if a computer spell-checker is used, the proof-reader does not have to ensure the spelling is correct, but still must confirm the actual word itself is the correct word. This paper considers building quality into the learning materials at the level of computational results using technology.

Many aspects of quality already use technology to lend assistance; an obvious example is the spell checker present in most word and document processors. Notice the phrase “lend assistance”. For any facet of quality, technology is only ever an assistant; see the example in the previous paragraph. As another example, the text Levin and Rubin (1993) was obviously prepared using technology, yet Chapter 5 is called Nonparametric Methods in the Table of Contents, while the Chapter itself is

called, and is really about, Probability II: Distributions. For many, spelling and grammar checkers are almost a panacea, but true quality in these areas is not the exclusive domain of technology. Some facets of quality cannot be assessed with technological assistance, and must be assessed by careful and considered reading by humans.

With this in mind, understand the quality of the computational results as discussed in this paper relate to the results being *accurate* rather *correct*. Comparable to the role of spell checkers, the technology discussed in this paper only ensures accurate answers to the given computations appear in the learning materials, not that the computations themselves are the correct computations to be performing. However, this technology does eliminate one source of quality-deficiency in the learning materials.

The accuracy of computations also assumes the software performing the computations is doing so correctly. This assumption may appear obvious, but it is not necessarily true; McCullough and Wilson (2002), McCullough (1998), McCullough (1999) and Altman and McDonald (2001) discuss how many popular software programs do not produce accurate computational results. In recognition of this, the National Institute of Standards and Technology, an agency of the U.S. Commerce Department's Information Technology Laboratory, established the Statistical Reference Datasets project. The purpose of this project is “to improve the accuracy of statistical software by providing reference datasets with certified computational results that enable the objective evaluation of statistical software” (Statistical Reference Datasets).

Accuracy of computational results

The quality facets discussed above are not all equal: the quality of some facets is more difficult to assess than others. For example, it is relatively easy to assess if a reference is *formatted* correctly, but far more time-consuming to check if the reference is the *correct* reference, and if the given page numbers for the reference are correct. Likewise, patience, skill and time are needed to check the result of every calculation in the text. If the learning materials contain hundreds of calculations, plots and tables, assessing the quality is tedious in the extreme.

The learning materials discussed in this paper refer to a final-semester climatology course at the University of Southern Queensland, Australia (USQ). USQ is a young university (established 1967, gaining University status in 1990), with a significant distance education program (in 2005, about 19 500, or 77%, of the total 25 378 enrolments were external). The course in question is an applied course using many real (and often large) data sets, available in traditional on-campus mode as well as through distance education. Using a statistical software package is essential for this course; indeed, *learning* to use the software is an essential outcome. The learning materials include a paper-based study book available to all students enrolled in the course, plus on-line supplements. The materials contain large amounts of mathematical computation and their results (including 70 plots and numerous tables), and contain a large number of software instructions for performing various tasks (2858 lines of instructions). While the course learning materials must embrace quality on every facet, the final quality of the learning materials crucially depends on the accuracy of the content in two particular areas: the results of computations must be free from *computational* errors, in quoted figures, in graphics and in tables of figures; and the materials must be free from *instructional* errors: all output, pictures, and tables must be produced exactly as shown by the supplied software instructions.

Related to these quality issues is the *maintenance of* these materials: All instructions, pictures, and tables must be easily updated when the software version changes, or when data files are updated.

The technological ideas and tools for achieving these goals are presented next.

General ideas

The general concept is simple: To avoid checking the result of every calculation and the accuracy of every instruction, actually *embed* the instructions for producing the results in the text, and let the computer place the result in the text in place of the instructions. This ensures the instructions in the text are the instructions actually used to produce the output, and the output is the output actually produced by the instructions.

Consider three examples to understand how this may arise. Suppose the text contains the following:

> The mean of the SOI is 0.575.

To check this computation, the data must be loaded into software and the mean computed: tedious indeed if many such computations appear. Notice the *result* of the calculation appears in the text; instead, embed the *computation* in the text, perhaps as follows:

> The mean of the SOI is *mean(soi).*

The *result* of the computation is replaced by commands to *compute* the result.

A related problem occurs when data is updated. For example, the course in question uses climatological data which may be updated periodically. When data files are updated, all computations and plots based on the data file must be updated also.

Suppose the materials contain a graphic plotting the monthly average SOI from 1990 to the end of 2002 (Figure 1).

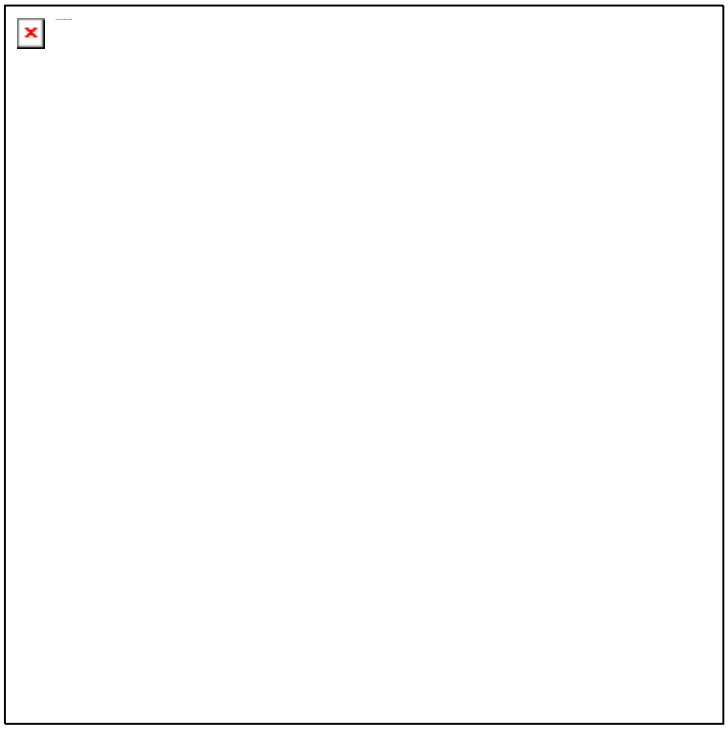


Figure 1: The SOI plotted from 1990 to the start of 2002

As more data is collected, an updated picture is needed. Updated data files are made available, and normally the updated graphic is made manually and imported into the text. However, consider letting the computer make the graph automatically by embedding the *instructions* to *make* the plot in the text:

```
plot( soi )
```

Three years later, when the materials are reproduced, the graphic is dynamically updated, and an updated graphic appears automatically (Figure 2).

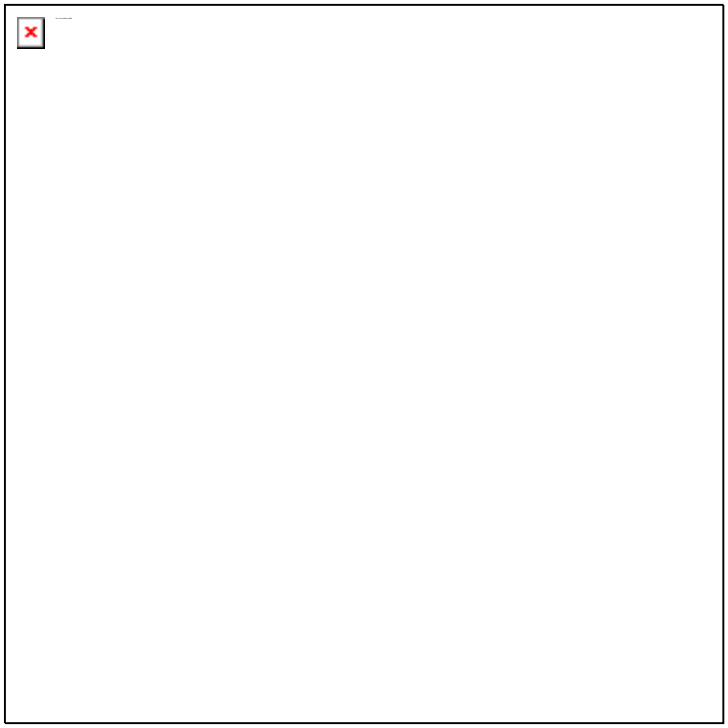


Figure 2: The SOI plotted from 1990 to the start of 2005

The problem is more acute when the learning materials are teaching students *how* to drive the software: the instructions must (should!) produce exactly the results shown. In addition, updated versions of software often produce differences in the results from the same instructions: formatting, language and information may differ in the output. Students need to be shown the correct output. Sometimes, the instructions themselves change when software is updated; embedding the instructions in the text ensures the correct instructions are used (otherwise the given instructions cannot be executed).

Suppose the text contains the following:

> Fit the model using the command
>
> arima(SOI, order=c(3,0,0))
>
> This command produces the output:
>
> ```
> Call:
> ```

```
arima(x = SOI, order = c(3, 0, 0))

Coefficients:
                  ar1     ar2     ar3  intercept
                0.5000  0.1648  0.0836    -0.1085
      s.e.      0.0256  0.0283  0.0256     0.7446
```

If the instructions ever change, if the output information or formatting ever changes, or if the data ever changes, this part of the learning materials is out-of-date. Better is to embed the instructions in the text, ensuring the instructions in the text are the instructions actually used to produce the output, and the output is the output actually produced by the instructions. Embed the instruction

arima(SOI, order=c(3,0,0))

Then allow the computer to place the resulting output in the text. This also ensures the instructions in the text are correct: if the instructions contain any errors, the instructions cannot—and will not—produce correct output, and may instead warn the user with an error message.

Three advantages are immediately apparent: The results of the computations are accurate, since the computer generated the results and placed them directly in the text; the output (such as graphs and tables) must be correct, and must be the output claimed to come from the given instructions, since the output of the instructions is the actual output in the text; and the instructions must be free of syntax errors, since the instructions in the text are the actual instructions used.

Of course, incorrect (rather than erroneous) instructions will produce incorrect answers, just as a spell checker will not find an error if "their" is used in place of "there". Again, the technology is an *aid* to human intervention in the quality process. In other words, this process can only ensure the results are accurate, not necessarily correct.

Details

The ideas presented here stem from the concept of literate programming (Knuth, 1984), implemented by Ramsey (1994) in the tool noweb. In literate programming, one file combines a computer program intermingled with the necessary documentation. Various applications are discussed by others: Rossini (2001) discusses the use of these tools in preparing statistical reports, while Carey (2001) discusses Ramsey's ideas for documenting software programs. The ideas are also related to the idea of reproducible research (Gentleman, 2005); one example is Vandewale *et al*. (2006), who use Matlab for their computational software; their results are reproduced at http://lcavwww.epfl.ch/reproducible_research/VandewalleSV05/index.html (accessed 20 Sept 2006). Vandewale *et al*. (2006) give a web page where their results can be reproduced. In this paper, we use the ideas and relevant technology to build quality directly into the learning materials that contain calculations.

The basic idea is as follows:

1. Create a document file in a word or document processor as normal.

2. Wherever software output (plots, tables, computation results) appear, place the software instructions there instead. These instruction need to be identified as such in some manner.
3. Let the computational software process the document, replacing the instructions with the requested output.
4. The result is the original document with the instructions replaced by the software output requested.

In the current context, the document processor used was LaTeX (Lamport, 1994), with the software package R (R Development Core Team, 2006). The actual software used is, in some sense, irrelevant since the principles apply to any combination of word or document processor and software, provided the necessary tools exist for those programs. In the R–LaTeX context, see Leisch (2002) and Leisch (2005). Currently, the only interpreted computational software I know using this approach is R; because of this, R is used as the computational software throughout this paper. However, there are no reasons why other computational software (such as MATLAB) cannot adopt the same ideas.

R works with three important document formats; LaTeX is identified above. In addition, html (web) pages can be created using R2HTML (Lecoutre, 2006), and with more commonly-used word processors provided they can use the Open Document format (Kuhn, 2006). Given USQ's high proportion of distance students, the ability to produce materials for the web and paper is a distinct advantage.

The most commonly-used format for creating documents is with conventional word processors; consequently, in this paper the application using familiar word processors is discussed. However, the ideas are the same and the implementation very similar for all document formats.

Any conventional word processor that can use the international standard OASIS (2005) Open Document Format for Office Applications can be used. In 2006, this format was approved and released as an ISO and IEC International Standard for office applications. Numerous word processors already support the OASIS OpenDocument format, including:

- The free OpenOffice.org office suite and many derivative products (NeoOffice; NextOffice 9.0; IBM Workplace Documents 2.6+); http://www.openoffice.org/
- The StarOffice office suite; http://www.staroffice.org
- The free KOffice office suite; http://www.koffice.org
- Google's new, free, web-based word processor Google Docs (formerly Writely); http://docs.google.com/
- The free word processor Abiword; http://www.abisource.com
- TextEdit for the Mac OSX.; http://www.apple.com/support/mac101/work/23/.

WordPerfect appears certain to adopt the ODF format eventually since Corel (owner of WordPerfect) is an original member of the OASIS Technical Committee proposing the Open Document Format, while Microsoft has made some claims of moving towards this industry standard (see the report at the Microsoft website http://www.microsoft.com/presspass/press/2006/jul06/07-06OpenSourceProjectPR.mspx, accessed 20 Sept 2006). However, until the next generation of Microsoft Office is released, Word users will need to wait to know for sure, but pressure may come to bear to adopt the international standard format.

Implementation

Consider how the four steps above are implemented in practice. In the usual manner, create a standard document in your favourite word processor, and save it using the international standard OpenDocument format (usually with an odt extension). Wherever R code or output is desired, the equivalent instructions are placed in the file. These instructions are identified as instructions in two ways.

First, if the instructions produce a small snippet of scalar output, the instructions are enclosed in \Sexpr{…} as follows:

```
The mean of the data is \Sexpr{signif( mean( soi ), 3)}.
```

This command computes the mean of the variable soi, and rounds it to three significant figures.

Secondly, if the instructions are lengthier or produce more than a single numerical output, enclose in <<>>=…@ as follows (in this example, note that the hash # is a comment character in R: all text after the # is ignored):

```
<<>>=
soi <- read.table("soidata.txt")        # Reads the data from a file
plot( soi )                             # Plot soi
@
```

Inside the double angle brackets, various options can be placed (for example, to control the size of the figure produced). Then, the file is processed by R, producing the expected output in the expected places. After processing, a new file (in the Open Document format) is produced.

Consider now a specific example of a document containing instructions to produce output (named example.odt in Figure 3). The R commands themselves are incidental; they are simply commands for the chosen software. This file is then processed in R using these commands:

library(odfWeave)

odfWeave("example.odt", "exampleout.odt")

The first command loads the required R package; the second converts the R instructions in the file example.odt into output, and produces the file exampleout.odt (Figure 4).

An simple analysis of tooth growth in guinea pigs

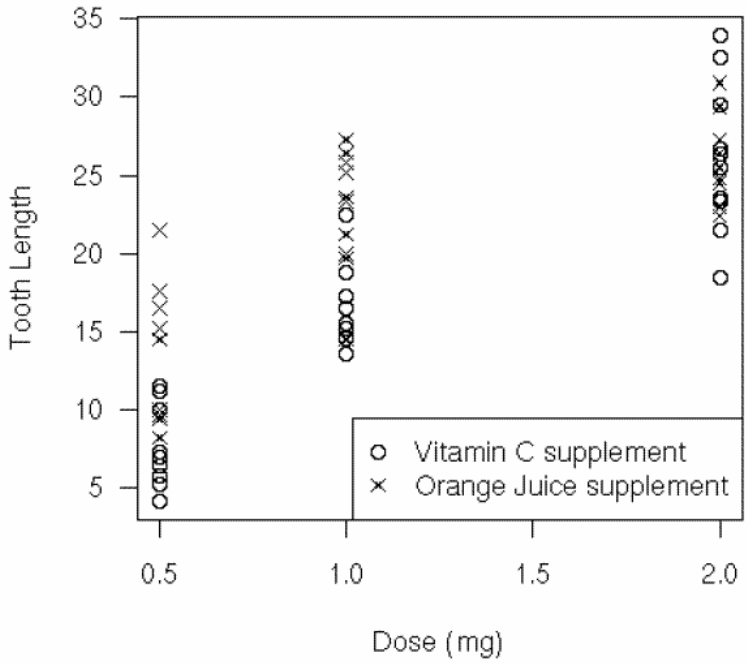
```
<<echo=FALSE>>=
# Some preliminaries
data(ToothGrowth)    # Load the R dataset
ToothGrowth$SuppType <- factor(ToothGrowth$supp)
attach(ToothGrowth)  # Make this the default data set
@

A study measured the length of teeth in guinea pigs at various doses of vitamin C (0.5, 1.0 and 2
mg) and using two different methods of giving the vitamin C (orange juice, or ascorbic acid).
There are \Sexpr{length(len)} observations.  There are two predictors:  the dosage (dose) is
quantitative; supplement type (Supptype) is categorical (also called a factor).

The following plot shows the amount of vitamin C is important for predicting length of teeth.

<<fig=TRUE,echo=FALSE,width=5,height=3>>=
# Plot of the data
plot(len ~ dose, type="n", ylab="Tooth Length", xlab="Dose (mg)", las=1)
points(len[supp == "VC"] ~ dose[SuppType == "VC"], pch = 1)
points(len[supp == "OJ"] ~ dose[SuppType == "OJ"], pch = 4)
legend("bottomright", pch = c(1, 4),
    legend = c("Vitamin C supplement", "Orange Juice supplement"))
@

The main effects model is fitted in R as follows:
<<echo=TRUE>>=
tg.lm <- lm( len ~ SuppType + dose ) # Fits the main effects model
@
The parameter estimates are shown below:
<<echo=FALSE,results=xml>>=
coef(summary(tg.lm))
odfTable( formatC(coef(summary(tg.lm)), digits=3)) # Display the coefficients
@
Clearly, both the supplement type and the dosage levels significantly effect tooth growth, though
the assumption of a linear relationship is perhaps unsupported.
```

Figure 3: The odf document named `example.odt` containing embedded computer instructions mixed with the text.

An simple analysis of tooth growth in guinea pigs

A study measured the length of teeth in guinea pigs at various doses of vitamin C (0.5, 1.0 and 2 mg) and using two different methods of giving the vitamin C (orange juice, or ascorbic acid). There are 60 observations. There are two predictors: the dosage (`dose`) is quantitative; supplement type (`supp`) is categorical (also called a *factor*).

The following plot shows the amount of vitamin C is important for predicting length of teeth.



The main effects model is fitted in R as follows:

```
> tg.lm <- lm(len ~ SuppType + dose)
```

The parameter estimates are shown below:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	9.27	1.28	7.23	1.31e-09
SuppTypeVC	-3.7	1.09	-3.38	0.0013
dose	9.76	0.877	11.1	6.31e-16

Clearly, both the supplement type and the dosage levels significantly effect tooth growth, though the assumption of a linear relationship is perhaps unsupported.

Figure 4: The same odf document as in Figure 1 after the instructions are evaluated. This is the document exampleout.odf.

Notice the difference types of interfacing between the input and output documents.

The first chunk of R instructions is executed, but is invisible in the output file (echo=FALSE). The command \Sexpr{length(len)} instructs R to compute the length of the variable len, and place the answer in the document. The next chunk of R instructions produces a plot which is inserted into the document (fig=TRUE). The commands are not shown in the output document (echo=FALSE), and the size of the graphic produced by R is also specified (width=5, height=3). The next chunk of R instructions is shown in the output document and is also executed (echo=TRUE). The final chunk of R instructions are not shown (echo=FALSE), but the resulting table appearing is correctly formatted in the output document (results=xml).

To some, perhaps it appears fanciful that a file containing instructions is so easily converted into a file containing output. But the ideas are not fanciful: I have used this method to produce the study book and examination for a USQ climatology course, and am using it to produce a statistics textbook for Springer. In both cases, accuracy of computational content is assured and maintenance time reduced. Other facets of quality in the final product still need to be addressed, but accuracy of results from computations is one less concern.

Conclusion

Assessing and ensuring the quality of learning materials is difficult when the materials contain more than a small number of calculation results. One solution is to embed instructions in text for the creating graphs, tables and computational results. This approach offers many significant advantages.

The first is an increase in quality: By embedding the software instructions in the text, the results of computations must be free of computational errors, since the

computations are performed by the instructions in the text and results placed directly in the text; all output, pictures, and tables must be produced exactly as shown by the supplied instructions; and all the instructions in the text *must* be, and have to be, free of syntax errors. In other words, quality is built into the learning materials, not assessed afterwards.

In addition, the materials are easier to maintain: all instructions, pictures, and tables are easily updated when the software version changes, or when data files are updated.

The technology to embed computer instructions in the text is demonstrably available. Importantly, it is also easy to use in practice. Given these facts, it would seem imperative to use the appropriate tools and technology in the production of learning materials that contain the results of computations; it is hard to find any compelling reason not to, apart from one.

The only impediment standing in the way of extensive adoption of this approach is the software. The embedding process is already possible using the R software package, in conjunction with LaTeX, html, and word processors that can use the international standard Open Document format. Extensions to other command-line based languages (such as Matlab, SPSS, Minitab, Octave, STATA, SAS) are not only suggested, but strongly encouraged. Such developments also are a strong motivation for encouraging Microsoft and other makers of proprietary word processor software to adopt the international standards of the Open Document format. Apart from the usual advantages, open file formats permit and encourage developments such as those given here. With the tools and technology evidently available, it would be approaching dishonourable if uptake of the ideas by makers of proprietary software stood between educators and one of their primary goals: quality in learning materials.

Acknowledgements

Thanks to Mrs Christine McDonald and Mr Tim Passmore for providing feedback on early drafts, and to a referee for helpful comments.

References

Anscombe, F. J., (1973). Graphs in statistical analysis. *American Statistician*, *27*, 17–21.

Altman, M. and McDonald, M. P. (2001). Choosing reliable statistical software. *Political Science and Politics*, *34*, 681–87.

Bolstad, W. M. (2004). *Introduction to Bayesian statistics.* New Jersey: Wiley-Interscience.

Carey, V. J. (2001). Literate statistical programming: concepts and tools. *Chance*, *14*, 46–50.

Castle, R and Kelly, D. (2004). International education: quality assurance and standards in offshore teaching: exemplars and problems. *Quality in Higher Education*, *10*, 51–7.

Gentleman, R. (2005). Reproducible Research: A Bioinformatics Case Study. *Statistical Applications in Genetics and Molecular Biology*, *14*, Article 2. Retrieved October 10, 2006, from http://www.bepress.com/sagmb/vol4/iss1/art2/

Glantz, S. A. (1980). Biostatistics: how to detect, correct and prevent errors in the medical literature. *Circulation*, *61*, 1–7.

Hand, D. J., Daly, F., Lunn, A. D., McConway, K. J. and Ostrowski, E. (1994). *A Handbook of Small Data Sets.* London: Chapman and Hall.

Hosie, P and Schibeci, R. (2005). Checklist and context-bound evaluations of online learning in higher education. *British Journal of Educational Technology*, *36*, 881–95.

Knuth, D. E. (1984). Literate programming. *The Computer Journal*, *27*, 97–111.

Kuhn, M. (2006). *odfWeave: Sweave processing of Open Document Format (ODF) files*, R package version 0.4.4.

Lamport, L. (1994). *LaTeX: A Document Preparation System*, 2nd edn. Reading, Mass: Addison Wesley Professional.

Lecoutre, E. (2006). *R2HTML: HTML exportation for R objects* (Version 1.57) [Computer software]. Retrieved 20 September 2006, from http://www.feferraz.net/en/R2HTML.html

Leisch, F. (2002, August). Sweave: dynamic generation of statistical reports using literate data analysis. *Compstat 2002—Proceedings in Computational Statistics*, Berlin, Germany.

Leisch, F. (2005). Sweave user manual. Retrieved 20 September 2006, from http://www.ci.tuwien.ac.at/~leisch/Sweave/

Levin, R. I. and Rubin, D. S. (1993). *Statistics for Management.* New Jersey: Prentice-Hall.

McCullough, B. D. (1998). Assessing the reliability of statistical software: Part I. *The American Statistician*, *52*, 358–66.

McCullough, B. D. (1999). Assessing the reliability of statistical software: Part II. *The American Statistician*, *53*, 149–59.

McCullough, B. D. and Wilson, B. (2002). On the accuracy of statistical procedures in Microsoft Excel 2000 and Excel XP. *Computational Statistics and Data Analysis*, *40*, 713–21.

Office of Legislative Drafting and Publishing (2006). *Higher Education Support Act 2003 as amended*. (Attorney-General's Department, Act No. 149). Canberra: Government Printing Office.

O'Leary, J., Kingston, B. and Hindmarsh, A. (2007). *The Times Good University Guide 2007*. Times Books.

Organization for the Advancement of Structured Information Standards (OASIS) (2005). Open document format for office applications (OpenDocument) v1.0. Retrieved 20 September 2006, from http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf

Ottewill, R and Macfarlane, B. (2004). Quality and the scholarship of teaching: learning from subject review. *Quality in Higher Education*, *10*, 231–41.

Priestly, W. (1991). Instructional typographies using desktop publishing techniques to produce effective learning and training materials. *Australian Journal of Educational Technology*, *7*, 153–63.

R Development Core Team (2006). R: A Language and Environment for Statistical Computing R Foundation for Statistical Computing (Version 2.3.1) [Computer software].Vienna, Austria. Retrieved from 20 September 2006, from http://www.r-project.org/

Ramsey, N. (1994). Literate programming simplified. *IEEE Software*, *11*, 97–105.

Rossini, A. (2001). Literate statistical analysis, paper presented at the *2nd International Workshop on Distributed Statistical Computing*, Vienna, Austria, March 15–17.

Statistical Reference Datasets. (n.d.) Retrieved 9 October, 2006, from http://www.itl.nist.gov/div898/strd/

Student (1908). The probable error of a mean. *Biometrika*, *6*, 1–25.

Tanur, J. M. (2005). Quality of statistical education: should ASA assist or assess? *The American Statistician*, *36*, 90–3.

Tufte, E.R. (1983). *The Visual Display of Quantitative Information.* Chesire, Conn.: Graphics Press

Vandewalle, P., Süsstrunk, S., and Vetterli, M. (2006). A frequency domain approach to registration of aliased images with application to super-resolution. *EURASIP Journal on Applied Signal Processing*, *2006*, 14 pages.