Distributed reproducible research using cached computations

Roger D. Peng Sandrah P. Eckel

April 30, 2008

Abstract

The ability to make scientific findings reproducible is increasingly important in areas where substantive results are the product of complex statistical computations. Reproducibility can allow others to verify the published findings and conduct alternate analyses of the same data. A question that arises naturally is how can one conduct and distribute reproducible research? We describe a simple framework in which reproducible research can be conducted and distributed via cached computations and describe tools for both authors and readers. As a prototype implementation we describe a software package written in the R language. The 'cacher' package provides tools for caching computational results in a key-value style database which can be published to a public repository for readers to download. As a case study we demonstrate the use of the package on a study of ambient air pollution exposure and mortality in the United States.

1 Introduction

The validity of conclusions from scientific investigations is typically strengthened by the replication of results by independent researchers. Full replication of a study's results using independent methods, data, equipment, and protocols, has long been, and will continue to be, the standard by which scientific claims are evaluated. In many fields of study, there are examples of scientific investigations which cannot be fully replicated, often because of a lack of time or resources. For example,

epidemiologic studies which examine large populations and can potentially impact broad policy or regulatory decisions, often cannot be fully replicated in the time frame necessary for making a specific decision. In such situations, there is a need for a minimum standard which can serve as an intermediate step between full replication and nothing. This minimum standard is *reproducible research*, which requires that datasets and computer code be made available to others for verifying published results and conducting alternate analyses.

There are a number of reasons why the need for reproducible research is increasing. Investigators are more frequently examining inherently weak associations and complex interactions for which the data contain a low signal-to-noise ratio. New technologies allow scientists in all areas to compile complex high-dimensional databases and the ubiquity of powerful statistical and computing capabilities allow investigators to explore those databases and identify associations of potential interest. However, with the increase in data and computing power come a greater potential for identifying spurious associations. In addition to these developments, recent reports of fraudulent research being published in the biomedical literature have highlighted the need for reproducibility in biomedical studies and have invited the attention of the major medical journals (Laine et al., 2007).

Interest in reproducible research in the statistical community has been increasing in the past decade with examples such as Buckheit and Donoho (1995), Rossini and Leisch (2003), Sawitzki (2002), and many others. The area of bioinformatics has produced projects such as Bioconductor (Gentleman et al., 2004) which promotes reproducible research as a primary aim (see also Ruschhaupt et al., 2004; Gentleman, 2005).

A proposal for making research reproducible in an epidemiologic context was outlined in Peng et al. (2006b). The criteria described there include a requirement that analytic data and the analytic computer code be made available for others to examine. The analytic data is defined as the dataset that served as the input to the analytic code to produce the principal results of the article. For example, a rectangular data frame might be analytic data in one case, a regression procedure might constitute analytic code, and regression coefficients with standard errors might be the principal results. Peng et al. (2006b) describe the need for reproducible research to be the minimum standard in epidemiologic studies, particularly when full replication of a study is not possible.

The standard of reproducible research requires that the source materials of a scientific investigation be made available to others. This requirement is analogous to the definition of open source software (see e.g. http://www.opensource.org/), which requires that the source code for a computer program be made available. However, by using the phrase "source materials" in the context of reproducible research, we do not mean merely the computer code that was used to analyze the data. Rather, we refer more generally to the preferred form for making modifications to the original analysis or investigation. Typically, this preferred form includes analytic datasets, analytic code, and documentation of the code and datasets.

1.1 Model for reproducible research

The model that we use to describe reproducible research is the "research pipeline" sketched in Figure 1. We model the research pipeline as beginning with "measured data" collected from na-

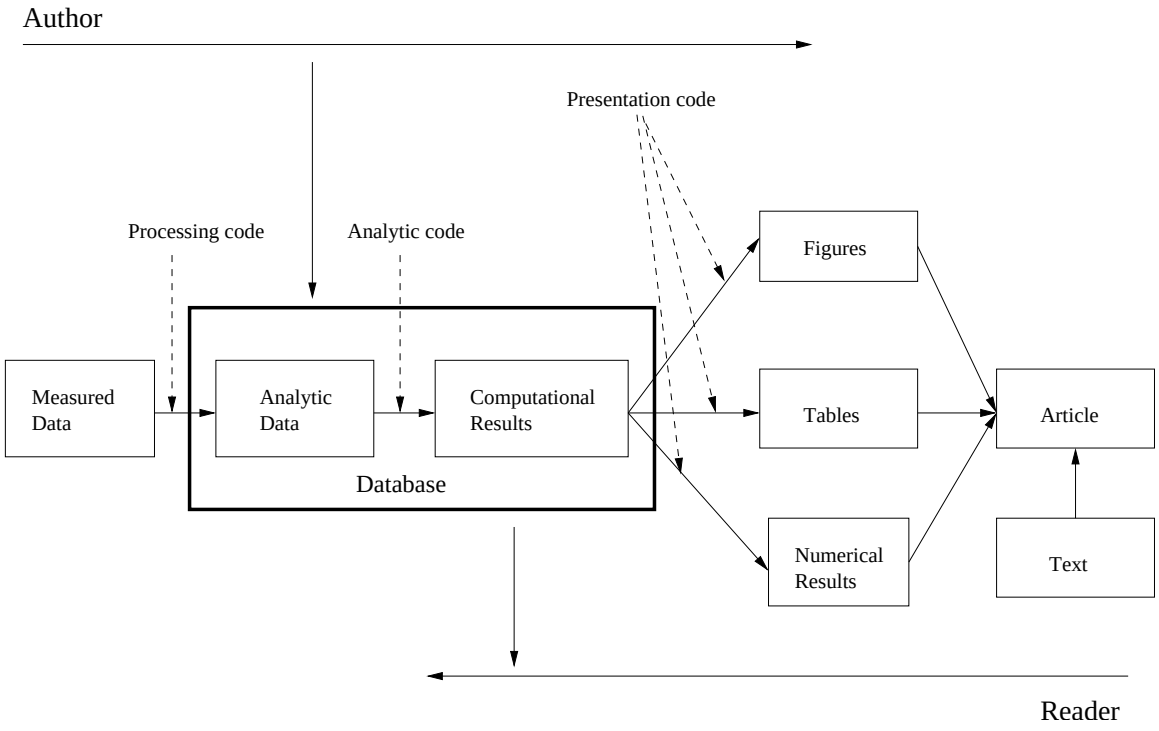

Figure 1: The research pipeline as a model for reproducible research.

ture which are then processed into "analytic data" via processing code and an associated software environment. Using a possibly different software environment (most likely a statistical analysis en-

vironment), the analytic data are then used to produce "computational results", which might be the output of regression models and various derived quantities. Computational results are then summarized in figures or tables or included as numerical results in the text, and these summarized results are then assembled with the expository text to form an article.

The principle underlying the research pipeline in Figure 1 is modularity. Modularity calls for the research process to be separated out into distinct components. We propose that a modular research pipeline lends itself more naturally to reproducible results because the existence of each component of Figure 1 in a semi-persistent state allows for the inspection of that component and the process that led to it. For example, one might be particularly interested in inspecting the analytic code that produces the computational results from the analytic data.

While a modular framework may seem reasonable, not all research is necessarily conducted in this manner. For example, software packages exist which will simultaneously process and analyze data, create a table, and embed the table in text, blurring the distinction between each of these stages. In addition, not all applications allow for the user to easily record the steps which lead from one state to another. Results from applications with graphical user interfaces are notoriously difficult to reproduce.

A modular research pipeline has implications for the way we analyze data, assemble results, and write articles. One important implication is that we must separate content from the presentation of content. This separation can only be achieved in practice if we have a reproducible means of going from one state to the other. That is why we need software that can take the results of computation and create useful summaries (e.g. figures and tables) or "views" of the data (Gentleman and Temple Lang, 2007).

One feature noted in the pipeline in Figure 1 is that authors and readers operate along opposite directions of the pipeline. Authors of papers start with the data, eventually building up to the analysis and then the paper. Readers start with the paper and, if sufficiently interested, begin to dig deeper into the details of the analysis by obtaining the relevant data and software. Given the data and software for a particular figure or table, the reader can reproduce those results and possibly conduct alternate analyses. In each direction, authors and readers need different sets of tools for

conducting reproducible research.

2 Cached Computations and the 'cacher' Package

One approach to distributing reproducible research is to use what we call "cached computations". Cached computations are intermediate results that are stored in a database as an analysis is being conducted. This database of stored results can be distributed via individual Web sites or central repositories so that others may explore the datasets and computer code for a given data analysis. Using the cached computations, readers can reproduce the original findings or take the cached results and produce alternate analyses for their own purposes.

Our implementation of this approach to using cached computations for reproducible research can be found in the 'cacher' add-on package for the R statistical computing environment (R Development Core Team, 2008). R is a widely used language for implementing statistical methods and for analyzing data in general. The software can be downloaded from the main project Web site (http://www.r-project.org/) and runs on all major computing platforms. The R system has a convenient mechanism by which users can add code to the base system in the form of packages.

The 'cacher' package provides tools for "caching" statistical analyses and for distributing these analyses to others in an efficient manner. The 'cacher' package has tools for both authors and readers of published statistical analyses. For authors, the 'cacher' package provides functions for evaluating R code and storing the results of computations in a key-value style database. There are also tools for creating "cache packages" for convenient distribution over the Web to others. For readers, there are tools for exploring a cached analysis and for evaluating selected portions of code. In addition, objects can be loaded from the database for inspection instead of evaluating the code directly to create the associated object. This feature is useful in situations where a complex statistical calculation might take a very long time to run and the reader is not interested specifically in verifying the results of the calculation. There are, however, tools for checking an analysis to see if the results that the reader gets from a given calculation match those of the original author. The internals of the 'cacher' package are described in much greater detail in Peng (2008).

The 'cacher' package can be obtained from the Comprehensive R Archive Network (CRAN)

at http://cran.r-project.org/ or at a nearby mirror (see http://cran.r-project.org/mirrors.html). The package can be installed by running the following R function

```
> install.packages("cacher")
```

which will download and install the cacher package in the default R library directory. We have also created a Web site, the Reproducible Research Archive, located at

http://penguin.biostat.jhsph.edu/

to host a number of cache packages created by the 'cacher' package. On the Web site, each cache package is assigned an identification string which is generated by the `package` function in the 'cacher' package. This identification string is generated from the contents of the cache package itself and is unique. Therefore, the ID string can be used as a global reference to the contents of the cache package. The ID string can also be used by readers to download the contents of the package to their own computers via the `clonecache` function. We will use some of these packages in the examples to follow.

3 Case Study

Estimation of the health risks of ambient air pollution is controversial for many of the reasons cited in the Introduction. The risks are inherently small (although the exposed population is large), there is a need for sophisticated computational and statistical tools, and substantive findings can play a significant role in the development of policy and regulation. These elements all conspire to make reproducibility a necessity in air pollution and health research.

The basis of our case study is the National Morbidity, Mortality, and Air Pollution Study (NMMAPS), which is a large observational study of the health effects of outdoor air pollution (Samet et al., 2000a). The purpose of NMMAPS is to investigate the short-term health effects of air pollution by 1) integrating national databases of population health, air pollution monitoring, weather, and socioeconomic variables; 2) developing statistical methods and computational tools for analyzing these databases; and 3) estimating the short-term associations between air pollution levels and mortality and their uncertainties in the largest U.S. metropolitan areas (Samet et al., 2000b,c). The

database and statistical methodology developed for NMMAPS are available from the Internet Health and Air Pollution Surveillance System [iHAPSS] (Zeger et al., 2006) website at http://www.ihapss.jhsph.edu/. The data have also been packaged separately as the 'NMMAPSdata' R package (Peng and Welty, 2004) and which can be downloaded from the iHAPSS website. We will not cover the statistical methodology used in NMMAPS, much of which has been detailed elsewhere (Peng et al., 2006a).

3.1 Exploring a cached analysis

The first analysis that we will explore is an examination of daily air pollution and mortality data in New York City. As one of the cities in the NMMAPS study, we have daily data on mortality, air pollution, and weather for the 14-year period of 1987–2000. The mortality data were obtained from the National Center for Health Statistics; the air pollution data was obtained from the Environmental Protection Agency's Air Quality System; and the weather data was obtained from the National Climatic Data Center.

The data and code for the analysis can be downloaded from the Reproducible Research Archive by using the `clonecache` function with the identification string for the cache package.

```
> library(cacher)
> clonecache(id = "7a188")
```

The full identification string is `7a188ec4e5a4af7253e202459009bce3f763ee91` but the `clonecache` function accepts an abbreviated version. Typically, only the first 7 or 8 characters of the ID string are necessary (a warning will be given if a unique match cannot be found). The `clonecache` package connects with the Archive Web site and downloads the necessary information to allow the user to explore the analysis.

Because you can cache analyses from multiple files in a single cache package, we can show which analyses have been cached in this package using the `showfiles` function.

```
> showfiles()
```

```
[1] "newyork.R"
```

In this case, there is only one analysis that has been cached and its source file is called "newyork.R".

If you want to examine an analysis, you can use the `sourcefile` function to choose that analysis and `showcode` will display the raw source file.

```
> sourcefile("newyork.R")
> showcode()
```

```
## Read in the data
classes <- readLines("colClasses.txt")
ny <- read.csv("data/ny.csv", colClasses = classes)


## Make some exploratory plots
par(mfrow = c(2, 2), mar = c(2, 4, 1, 1), pch = ".")

## PM10
with(ny, plot(date, pm10tmean + pm10mtrend, ylab = expression(PM[10])))

## Carbon monoxide
with(ny, plot(date, cotmean + comtrend, ylab = "Carbon monoxide"))

## Ozone
with(ny, plot(date, o3tmean + o3mtrend, ylab = "Ozone"))

## Mortality
with(ny, plot(date, death, ylab = "Mortality"))


## Fit a log-linear Poisson model with overdispersion
library(stats)
library(splines)

fit <- glm(death ~ dow + ns(date, 7*14) + ns(tmpd, 6) + ns(rmtmpd, 6)
           + ns(dptp, 3) + ns(rmdptp, 3) + l1pm10tmean,
           data = ny, family = quasipoisson)

## Extract log relative risk for PM10
```

```
summ <- summary(fit)
summ$coefficients["l1pm10tmean", ]
```

This analysis first loads the data for New York City and then makes a series of exploratory plots of the pollution and mortality data. The pollutants that are plotted are particulate matter less than 10 μm in aerodynamic diameter (PM_{10}), carbon monoxide, and ozone. The mortality data are daily counts of deaths in New York City from all non-accidental causes. After plotting the data, a generalized linear model is fit to the data to estimate the association between PM_{10} and mortality. A summary of the fitted model is extracted and then the log relative risk for PM_{10} is printed (along with its standard error, *t*-statistic, and *p*-value).

You can also use the `code` function to display the code in an abbreviated form. The `code` function also displays each expression's sequence number which can be used in conjunction with other functions in the 'cacher' package.

```
> code()

source file: newyork.R
1  classes <- readLines("colClasses.txt")
2  ny <- read.csv("data/ny.csv", colClasses = classes)
3  par(mfrow = c(2, 2), mar = c(2,
4  with(ny, plot(date, pm10tmean +
5  with(ny, plot(date, cotmean + comtrend,
6  with(ny, plot(date, o3tmean + o3mtrend,
7  with(ny, plot(date, death, ylab = "Mortality"))
8  library(stats)
9  library(splines)
10  fit <- glm(death ~ dow + ns(date,
11  summ <- summary(fit)
12  summ$coefficients["l1pm10tmean",
```

The `code` function truncates expressions to a single line and also shows the sequence number assigned to each expression in the order that the expression is encountered in the source file. To see the full code for each expression, you can set the `full = TRUE` option to `code`.

The first thing you might do when exploring a cached analysis is to explore the elements of the cache database itself. You can list the objects available using the `showobjects` function, which returns a character vector of the names of each object in the database. Passing an expression sequence number to `showobjects` via the `num` argument shows the objects created by that expression.

```
> showobjects()
```

```
[1] "classes" "ny"      "fit"     "summ"
```

In expression 10, a log-linaer Poisson model (with overdispersion) is fit to the data to assess the association between PM_{10} (the `l1pm10tmean` variable) and mortality, adjusting for other factors such as temperature (`tmpd` and `rmtmpd`), dew point temperature (`dptp` and `rmdptp`), and day of the week (`dow`). The fitted model is stored in a objects called `fit`.

The objects in the cache package can be loaded into the workspace using the `loadcache` function.

```
> loadcache()
> ls()
```

```
[1] "classes" "fit"     "ny"      "summ"
```

The `loadcache` function takes a `num` argument which can be a vector of indices indicating code expression sequence numbers. For example, if you want to load only the objects associated with expression 10 (i.e., the `fit` object), then you can call `loadcache(10)`. Now we can print the linear model fit (without actually fitting the model) by calling

```
> print(fit)
```

(We do not show the output here because it is quite long.) To examine the log relative risk for PM_{10} we can just run the last two expressions.

```
> runcode(11:12)
```

```
    Estimate   Std. Error      t value     Pr(>|t|)
0.0007084746 0.0002930637 2.4174763362 0.0158692343
```

Here we see that the log relative risk for PM_{10} is 0.000708. This translates to a 0.71% increase in mortality associated with a 10 μg/m^3 increase in PM_{10} (this is a common unit for reporting air pollution risks). While this might seem like a miniscule risk to worry about, it is important to remember that very large populations are typically exposed to ambient air pollution.

In addition to exploring the objects in the cache database, you may wish to run the analysis on your own computer for the purposes of reproducing the original results. You can run individual expressions or a sequence of expressions with the `runcode` function. The `runcode` function accepts a number or a sequence of numbers indicating expressions in an analysis. For example, the first seven expressions in the "newyork.R" analysis create exploratory plots of some of the time series data in the New York City dataset.

```
> code(1:7)

source file: newyork.R
1  classes <- readLines("colClasses.txt")
2  ny <- read.csv("data/ny.csv", colClasses = classes)
3  par(mfrow = c(2, 2), mar = c(2,
4  with(ny, plot(date, pm10tmean +
5  with(ny, plot(date, cotmean + comtrend,
6  with(ny, plot(date, o3tmean + o3mtrend,
7  with(ny, plot(date, death, ylab = "Mortality"))
```

In order to run the first seven expressions in the "newyork.R" analysis, we could call

```
> runcode(1:7)
```

In this case, expressions 3 through 7 are evaluated but expressions 1 and 2 are loaded from the cache. By default, for efficiency reasons `runcode` does not evaluate expressions for which it can load the results from the cache. In order to force evaluation of all expressions, you need to set the option `forceAll = TRUE`.

The resulting plot is shown in Figure 2. This figure shows daily PM_{10}, carbon monoxide, ozone, and mortality counts for New York City for the 14-year period 1987–2000. We can see that levels of carbon monoxide have decreased substantially over the past 14 years while levels of PM_{10} and

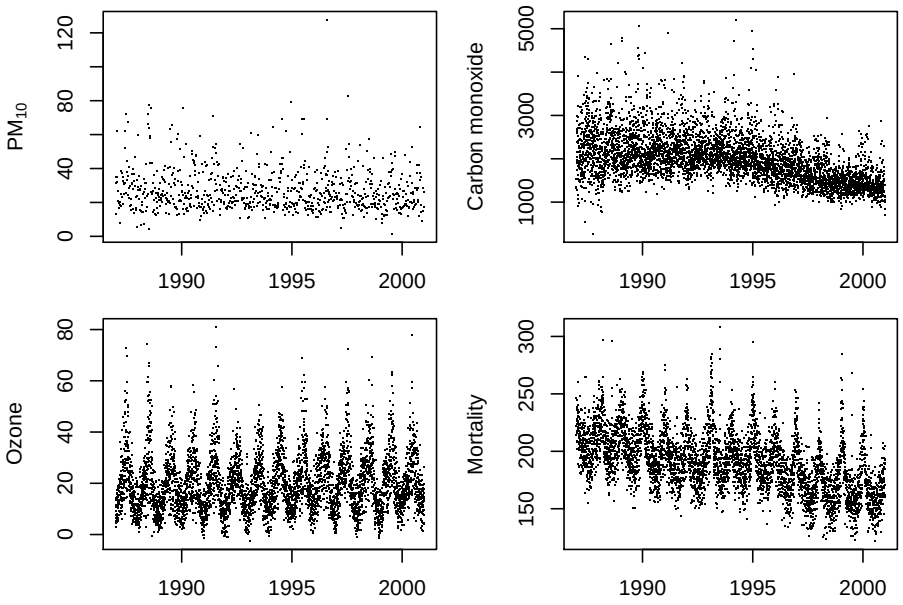

Figure 2: Daily PM_{10}, carbon monoxide, ozone, and mortality counts for New York City, 1987–2000

ozone have remained relatively constant. Daily mortality counts have also decreased steadily over this time period.

3.2 Alternate analyses

One important reason for making statistical analyses reproducible is to allow others to build on published findings and conduct alternate analyses. In this section we provide an example of conducting such an alternate analysis and compare our results to the original.

The original analysis that we will examine builds on the analysis of the previous section. This analysis estimates the risk of mortality due to particulate matter in the largest 20 cities in the United States. Combining data from multiple locations in air pollution studies is a powerful approach to risk estimation and it allows us to examine the variation in risks across locations. Using data from multiple locations, we can calculate a "national average" risk estimate that is much more precise than any single estimate from a particular city. Such national risk estimates are often useful for policy makers who must promulgate uniform air quality standards across the country.

As before the code and data for this analysis can be downloaded using the `clonecache`

function and the cache package's abbreviated identification string from the Reproducible Research Archive.

```
> clonecache(id = "092dc")
```

We can see what source files are available and set the appropriate one.

```
> showfiles()

[1] "top20.R"

> sourcefile("top20.R")
```

The code for this analysis can be viewed with the `showcode` function.

```
> showcode()

## Analysis of the largest 20 cities in the NMMAPS database

## Read in the data
cities <- readLines("citylist.txt")
classes <- readLines("colClasses.txt")

## Only keep variables we need
vars <- c("date", "dow", "death", "tmpd", "rmtmpd", "dptp", "rmdptp",
          "l1pm10tmean")

data <- lapply(cities, function(city) {
        filename <- file.path("data", paste(city, "csv", sep = "."))
        d0 <- read.csv(filename, colClasses = classes, nrow = 5200)

        d0[, vars]
})
names(data) <- cities

## Analysis of PM10
estimates <- sapply(data, function(city) {
        fit <- glm(death ~ dow + ns(date, 7*14) + ns(tmpd, 6) + ns(rmtmpd, 6)
```

```
                    + ns(dptp, 3) + ns(rmdptp, 3) + l1pm10tmean,
                    data = city, family = quasipoisson)
        summ <- summary(fit)
        summ$coefficients["l1pm10tmean", 1:2]
})

effect <- weighted.mean(estimates[1, ], 1 / estimates[2, ]^2)
stderr <- sqrt(1 / sum(1 / estimates[2, ]^2))
```

This analysis reads in the data from the 20 cities and fits a log-linear Poisson model with overdispersion to the data from each city to estimate the association between PM_{10} exposure (at a one day lag) and mortality. A national estimate is produced by taking a weighted average of the city-specific estimates where the weights are inversely proportional to the variances of the risk estimates. This national estimate and its standard error are stored in the `effect` and `stderr` objects.

While we could run the entire analysis again with the `runcode` function, we can save some time by loading the objects directly and examining the final product

```
> loadcache()
> ls()

 [1] "cities"    "classes"   "data"      "effect"    "estimates" "fit"
 [7] "ny"        "stderr"    "summ"      "vars"
```

We can first see the national average estimate and its standard error.

```
> print(effect)

[1] 0.0002313219

> print(stderr)

[1] 0.000052457
```

This log relative risk estimate translates to a 0.23% increase in mortality associated with a 10 μg/m^3 increase in PM_{10}. We can also generate a t-statistic to gauge the statistical significance of the estimate.

```
> print(effect/stderr)

[1] 4.409743
```

With a t-statistic of 4.4 we could interpret the national average estimate as strong evidence of an association between PM_{10} and daily mortality.

However, one might prefer an alternate modeling approach to the one taken here. In particular, taking a simple weighted average of the city-specific estimates ignores any heterogeneity due to unmeasured factors that might exist between the risk estimates from the different cities. We can fit a random effects model to the data to examine whether taking into account such heterogeneity would substantially alter our conclusions

One package that can be used to fit random effects models to data like these is the Two-level Normal Independent Sampling Estimation ('tlnise') software of Phil Everson (Everson and Morris, 2000). The `tlnise` function fits a random effects model and produces a pooled estimate (similar to taking the weighted average) as well as estimates of the heterogeneity.

```
> library(tlnise)

Two-level normal independent sampling estimation (version 0.2-7)

> fit <- tlnise(estimates[1, ], estimates[2, ]^2, prnt = FALSE)
```

We can print the final national average estimate produced by the `tlnise` function.

```
> print(fit$gamma)

              est            se   est/se
[1,] 0.0002323276 0.00006915325 3.359604
```

Here we see that the estimate is roughly the same as when we took the weighted average but the standard error is somewhat larger (0.00007 versus 0.00005), resulting in a somewhat reduced t-statistic. Nevertheless, the evidence of an association between PM_{10} and mortality still appears to be strong, even with the random effects model.

4 Discussion

In this article we have described a method by which reproducible research can be distributed using cached computations. Cached computations are created during the execution of a statistical analysis and are stored in a collection of databases. These databases can subsequently be published in public repositories for others to download and explore. These databases of cached computations provide a middle ground where the authors and readers can virtually interact with each other for the purposes of reproducing scientific results. The method we describe enables authors to easily give access to their data and code to a large number of potential readers and allows readers to obtain immediate access to the materials needed for reproducing a specific result. As an implementation of our cached computation framework, we have developed the 'cacher' package for R.

Leisch (2002) and others have built upon Donald Knuth's literate programming concept (Knuth, 1984) and have extended it to the creation of reproducible statistical documents. The specific implementation of Leisch (2002) is called Sweave and combines the LaTeX document formatting language with the R programming language. In this scheme, documents are divided into "chunks"—either text chunks or code chunks—each of which is processed in a different way. Text chunks are written in a document formatting language and code chunks are written in a computer programming language. Analogous to Knuth's system, the document can be "weaved" to form a human readable document (e.g. an article or report in PDF) and "tangled" to produce machine executable code.

In general, there is still a need for developing tools that authors can use to create reproducible documents. For authors using R, a variety of tools are currently being developed by others towards this aim. The Emacs Speaks Statistics package is a powerful tool for writing documents using the Emacs text editor and a number of different statistical programming languages (Rossini et al., 2004). Also, the 'DynDoc' package from Bioconductor defines data structures for working with vignettes created by Sweave. The StatDocs project (http://www.stat.berkeley.edu/users/statdocs/) is an interesting effort to provide a number of tools via Omegahat (http://www.omegahat.org/) for creating dynamic statistical documents with a focus on reproducibility.

The idea of using a single file that contains both statistical programming code and human readable text is not limited to Sweave and R. Another common statistical software package, Stata, is used

mainly for interactive data analysis by way of hand-entered commands or commands run through a .do file. Stata can save command outputs except for graphs in a .log file, but this .log file is not readily readable by a larger audience. Recently, Gini and Pasquini (2006) have demonstrated that it is possible to generate reports, presentations or webpages in an automated fashion using Stata .do files. Their work builds on that of Newson (2003) and others in translating Stata output into markup languages for inclusion in LaTeX or HTML documents. Instead of interweaving code and text chunks, the Gini and Pasquini (2006) approach consists entirely of code. Human readable text is included by writing each line of the text to an output file using the Stata command 'file write' in the master .do file. Hence human readable text is not as readily incorporated in the final document as it is in the Sweave approach. MATLAB offers a structured, Windows interface Report Generator that allows for push-button generation of analysis documentation (MathWorks, 2007). The MATLAB Report Generator can be used to automatically generate reports in a wide variety of formats including a navigatable set of HTML webpages, PDF, and Microsoft Word. The MATLAB Report Generator can produce reports whose content is conditional on analysis results.

The software described in this article consists primarily of tools for assisting authors in conducting reproducible research and producing reproducible documents. For readers, the 'cacher' package provides tools for viewing code, exploring data, and reproducing results in a data analysis via cache packages. However, substantial improvements could be made to allow the reader to interact more closely with the analysis and to perhaps lessen the dependence on knowledge of R programming. One possible step in this direction would be to allow the reader to interact with an analysis using a graphical user interface or a Web browser. One example along these lines is the 'vExplorer' package of Zhang and Gentleman (2004) which is written with the tcl/tk language and toolkit and provides a graphical user interface for exploring R package vignettes.

Finally, our implementation of a cached computation framework in R is not meant to imply that it is not possible to develop a similar implementation using another software system. The reproducibility of research necessarily depends on the specific software system used in an analysis (and its availability to others) and at this point it does not seem possible to develop a more general system to encompass all possible analyses. We propose the cached computation approach as a

model around which similar systems could be developed.

The potential benefits of reproducible research to scientists and statisticians are substantial. By expediting the dissemination of ideas and publishing the "research behind the research", investigators can more easily build on existing knowledge. However, reproducible research cannot be conducted without the proper tools available to authors and readers and the development of a reproducibility infrastructure should be a major focus of future work.

5 Data

One can download all of the data used in the examples in this paper from

http://penguin.biostat.jhsph.edu/CiSE/data.zip

The archive file also contains the R code for analyzing the data.

6 Acknowledgments

This research was supported in part by a Faculty Innovation Fund award from the Johns Hopkins Bloomberg School of Public Health, grant ES012054-03 from the National Institute of Environmental Health Sciences, and the Johns Hopkins Training Program in the Epidemiology and Biostatistics of Aging (NIA T32 AG00247).

References

Buckheit, J. and Donoho, D. L. (1995), "Wavelab and reproducible research," in *Wavelets and Statistics*, ed. Antoniadis, A., Springer-Verlag, New York.

Everson, P. J. and Morris, C. N. (2000), "Inference for Multivariate Normal Hierarchical Models," *Journal of the Royal Statistical Society, Series B*, 62, 399–412.

Gentleman, R. (2005), "Reproducible Research: A Bioinformatics Case Study," *Statistical Applications in Genetics and Molecular Biology*, 4, Article 2.

Gentleman, R. and Temple Lang, D. (2007), "Statistical Analyses and Reproducible Research," *Journal of Computational and Graphical Statistics*, 16, 1–23.

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A. J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J. Y. H., and Zhang, J. (2004), "Bioconductor: open software development for computational biology and bioinformatics," *Genome Biology*, 5, R80.

Gini, R. and Pasquini, J. (2006), "Automatic generation of documents," *The Stata Journal*, 6 (1), 22–39.

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

Laine, C., Goodman, S. N., Griswold, M. E., and Sox, H. C. (2007), "Reproducible Research: Moving toward Research the Public Can Really Trust," *Annals of Internal Medicine*, 146, 450–453.

Leisch, F. (2002), "Sweave: Dynamic generation of statistical reports using literate data analysis," in *Compstat 2002 — Proceedings in Computational Statistics*, eds. Härdle, W. and Rönz, B., Physika Verlag, Heidelberg, Germany, pp. 575–580, ISBN 3-7908-1517-9.

MathWorks (2007), "MATLAB Report Generator," [Online; accessed 30-April-2007].

Newson, R. (2003), "Confidence intervals and *p*-values for delivery to the end user," *The Stata Journal*, 3 (3), 245–269.

Peng, R. D. (2008), "Caching and distributing statistical analyses in R," Tech. Rep. 169, Johns Hopkins University Department of Biostatistics, http://www.bepress.com/jhubiostat/paper169.

Peng, R. D., Dominici, F., and Louis, T. A. (2006a), "Model choice in time series studies of air pollution and mortality (with discussion)," *Journal of the Royal Statistical Society, Series A*, 169, 179–203.

Peng, R. D., Dominici, F., and Zeger, S. L. (2006b), "Reproducible Epidemiologic Research," *American Journal of Epidemiology*, 163, 783–789, doi:10.1093/aje/kwj093.

Peng, R. D. and Welty, L. J. (2004), "The NMMAPSdata Package," *R News*, 4, 10–14.

R Development Core Team (2008), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.

Rossini, A. and Leisch, F. (2003), "Literate Statistical Practice," in *Proceedings of the 2nd International Workshop on Distributed Statistical Computing*, eds. Hornik, K. and Leisch, F., pp. 1–10.

Rossini, A. J., Heiberger, R. M., Sparapani, R. A., Mächler, M., and Hornik, K. (2004), "Emacs Speaks Statistics: A Multiplatform, Multipackage Development Environment for Statistical Analysis," *Journal of Computational and Graphical Statistics*, 13, 247–261.

Ruschhaupt, M., Huber, W., Poustka, A., and Mansmann, U. (2004), "A Compendium to Ensure Computational Reproducibility in High-Dimensional Classification Tasks," *Statistical Applications in Genetics and Molecular Biology*, 3, Article 37.

Samet, J. M., Dominici, F., Curriero, F. C., Coursac, I., and Zeger, S. L. (2000a), "Fine Particulate Air Pollution and Mortality in 20 US Cities," *New England Journal of Medicine*, 343, 1742–1749.

Samet, J. M., Dominici, F., Zeger, S. L., Schwartz, J., and Dockery, D. W. (2000b), *The National Morbidity, Mortality, and Air Pollution Study, Part I: Methods and Methodological Issues*, Health Effects Institute, Cambridge MA.

Samet, J. M., Zeger, S. L., Dominici, F., Curriero, F., Coursac, I., Dockery, D. W., Schwartz, J., and Zanobetti, A. (2000c), *The National Morbidity, Mortality, and Air Pollution Study, Part II: Morbidity and Mortality from Air Pollution in the United States*, Health Effects Institute, Cambridge, MA.

Sawitzki, G. (2002), "Keeping Statistics Alive in Documents," *Journal of Computational and Graphical Statistics*, 17, 65–88.

Zeger, S. L., McDermott, A., Dominici, F., Peng, R. D., and Samet, J. M. (2006), *Internet-Based Health and Air Pollution Surveillance System*, Communication 12, Health Effects Institute, Boston MA.

Zhang, J. and Gentleman, R. (2004), "Tools for Interactively Exploring R Packages," *R News*, 4, 20–25.