A joint newsletter of the Statistical Computing & Statistical Graphics Sections of the American Statistical Association.

Summer 2002
Vol.13 No.1

Statistical COMPUTING & GRAPHICS

A WORD FROM OUR CHAIRS

Statistical Computing

Susan Holmes is the 2002 Chair of the Statistical Computing Section.

I'm writing my first column for the 2002 as chair of computing, although we have already met in these columns as I have been editor for the section for more than a year now. As I scramble to find time to do my own research on the bootstrap and its applications to biology, in particular the problem of phylogenetic trees, I am confronted more and more by the digital divide that separates me from the community for which I have chosen to work: biologists.

I have been collaborating with biologists for more than 20 years now, and even my initial venture involved statistical computing, since at the time I helped develop software for teaching multivariate analysis using the Apple II. In 2002, we have many more tools available to the specialists in statistical computing, as professionals we use `C++`, `Java`, `R/Splus` or `matlab`, but our research only trickles down to scientists very slowly.

`R` in particular has been making leaps and bounds towards the biological and medical community as the project `Bioconductor` has emerged for analyzing microarrays, and two very nice Mac versions (one for Mac OS 9 and one for Mac OS X) of R now available to the Mac-bios.

Statistical Graphics

Steve Eick is the 2002 Chair of the Statistical Graphics Section.

It is my pleasure to be your graphics section president this year and one of the honors that goes with the office is to write a short message to the members.

Mario Peruggia pulled together a great Graphics program for the JSM this summer. For more details, see his write-up later in this newsletter. The Stat Graphics and Computing Meeting is Monday night at the JSM. It's always fun, good food, interesting door prizes, please come.

The world has certainly become much more dangerous than last year. For us in the high-tech industry, the current sharp recession has made life interesting, challenging, and highly distracting for me on a personal level. Being an executive at a high-tech company during a period when capital spending for software essentially freezes is really tough an experience I wish never to have again.

One a positive note, however, the need for statistical graphics and visualization continues to increase. Spending freezes are lifted eventually. Interesting computers are inexpensive and conductivity is becoming ubiquitous. At a fundamental level, every nine months the amount of data stored on disk doubles, a trend that is likely to continue for another decade. The technical, business, and research challenge is how to

SPECIAL ARTICLE

"Its Just Implementation!"

Di Cook (dicook@iastate.edu), **Igor Perisic** (iperisic@entopia.com), **Duncan Temple Lang** (duncan@research.bell-labs.com), **Luke Tierney** (luke@stat.umn.edu)

Col says: "I just got off the phone with my sister. I called to wish her happy birthday but before long the conversation shifted from fun topics and onto work-related topics. She has a PhD in entomology and works on phylogeny of insects based on genetic information. In our conversation she revealed that she was doing 'Bayesian inference using Markov Chain Monte Carlo'. I was stunned! I asked her how she heard of these techniques, was she working with some of the excellent statisticians on campus, ... Her response was that she had read a couple of papers that mentioned the methods, and then did an internet search for software. The search yielded two packages Mr Bayes (http://morphbank.ebc.uu.se/mrbayes/) and 'PAUP' (http://paup.csit.fsu.edu/about.html). She has been testing both programs on her data. (And she has bought herself a dual-process, large memory Mac to do the testing!)

Software is a major communication tool today. It is common that people download publicly available software to experiment with new methodology. They may even be motivated to then read the associated literature and learn about the methods. But software is becoming a major entry point into new research."

Cam says: "The development of prominent statistical software systems such as S, XGobi, (SPSS, etc.) have been done fortuitously and in very special environments that may not be reproducible. We need to find a way to make such breakthroughs more likely to occur."

Col: "Statistical software breakthroughs have historically occurred under the nurturing of liberal research departments in commercial companies and its still occurring in places such as Lucent Bell Labs. But support for liberal research is dwindling in the face of demands for short-term accountability and productivity. Its going to be important for academia to develop research environments that facilitate computational breakthroughs."

Ric says: "If we want to give some academic value to code that is freely distributed within say R or ... Then we still do have a problem of communicating its value to the statistical community."

Kel says: "My view is that at this point many departments would in principle like to consider software development as part of research and scholarly work but do not really know how to make the case among themselves, i.e. for departmental promotion recommendations, or to the college level."

Col: "The provost at Institution X verbally supported the inclusion of software research in faculty promotion decisions, if we can tell them how it should be counted."

Liam says: "The key to accountability is peer evaluation. Technical papers have a peer evaluation process, software does not."

Col: "Although the section on Statistical Software of the Journal of Computational and Graphical Statistics may be changing this."

Cam: "Serious computing is a considerably time-consuming endeavor. Unfortunately, we are at a point where there are different generations of statisticians cohabiting the field with very different experiences. Now, almost all graduate students will be experienced with the computer. However, their advisors will be less familiar. Importantly, many of these more senior members will be less aware of a) the learning curve and general day-to-day complexity of computer programming, and b) are not able to necessarily identify or appreciate well-written programs that support easy modification, etc. The effect is often to 'encourage' people to write code quickly and often rather than to stop and think about how to write good software once (or twice :-)).

We don't want to necessarily move away from ad hoc statistical software development. Rather we want to encourage good statistical computing practice research and innovation. One aspect of good practice involves re-using what is already there (and hence leveraging and integrating with existing tools)..... The requirement that people survey the existing 'literature' in the area and reference and build on it is vital to any academic discipline. It is an area where we in statistical computing have not been very rigorous, to our detriment. We don't want to give people the impression that there is a single way to do things. Just that there is level or standard of software that needs to be considered.

It is clear that there is a growing awareness and appreciation of statistical computing research. However, given that it is a reasonably new endeavor, there is less understanding of how to evaluate this type of research, especially when considering hiring and promotion of faculty. In order to count these new 'beans', we need to help evaluate statistical computing research."

Background: A year ago the American Statistical Association section on Statistical Graphics and Comput-

ing held a workshop on the future of statistical computing in conjunction with the Interface between Computer Science and Statistics. A summary of the workshop was published in the Statistical Computing and Graphics newsletter, volume 11, issue 1 (2001). The workshop grew out of concerns for the status of computing in the Statistics community. One of the topics discussed was how to improve the recognition of software research in academia: when recruiting graduate students, in assessing student's research, when considering new candidates for positions, promoting and rewarding faculty. These comments here are a snapshot of discussions that have been occurring in email since the workshop. Duncan Temple Lang, Luke Tierney, Igor Perisic and Di Cook 'volunteered' to write a document describing approaches to evaluate statistical computing and graphics research. Its turned out there's been some considerable difference of opinion on the direction of the article. We first began to develop a 'check list' of items that could frame a work of software or place it in context. The check list grew from the experiences of the John Chambers award committee last year. Here are some comments that arose after this original draft was distributed...

Kel: "I think a check list is a bad idea for two reasons. One is that software development is too heterogeneous for a single check list to work for all cases, much like research papers are too heterogenous (you can't write a check list that asks 1) how many theorems, 2) how many plots, 3) how many figures or some such and hope to get anything useful out of it. The other reason is that the target audience, faculty who are evaluating someone who does software research but don't do software development themselves, may not be able to appropriately use such a list even if we could present them with one."

Ric: "I'd agree with you here in the sense that you will not judge if a code is 'useful' or .. by a check list, but there are accepted standards within each type of publication. Research papers follow some rules which although often implicit are nevertheless present. Say for example, you should be able to read the paper and understand what it is all about or that it starts with an abstract and intro, and so on ... The idea is similar to the fact that a great idea/result still needs to be clearly transcribed in order to make a good paper.

Now, we all know that departments hold lists of at least top-tier/middle tier/lower tier publications. So the problem becomes how we can match this list to software/code. How should a published code be valued? Does it matter if it is available only from your web page as opposed to the official page of ... which is 'filtered'? The peculiarity of code is that a lot of public code is managed by a reputation management scheme that is automatic. i.e. you can publish your code and new code but it will get less and less used/downloaded if it is shoddy. In this regard the task that we would have here is, how do we tell evaluating faculty members that this software development is worthy? "

Col: "Developing software is often viewed to be primarily implementation, a less than creative, respectable endeavor. This is a misguided opinion. Creating software commonly involves ingenuity, good computing literacy and awareness of the current computational technology. It takes considerable skill to develop a framework for statistical computations, especially in interactive and dynamic environments. I'm concerned that we are not adequately including enough people with sophisticated computing expertise in academia. How do we encourage creative research in statistical computing and graphics in academia?"

Kel: "Most universities will have other departments that face similar issues, such as performing arts where performance evaluations may figure into promotion decisions, and finding out what the locally appropriate mechanisms are is one useful recommendation."

Cam: "There are many different areas of statistical computing. In the past we have focussed on the important field of algorithms, numerical analysis, etc. These are the more traditional computational equivalents of mathematical theory. While development of systems (e.g. S, XGobi, XLisp-Stat, etc.) has been rarer for obvious reasons, these haven't been as widely recognized as 'regular research'. Similarly, practical parallel and distributed computing, integration and computer automation are important but less visible aspects of computing that we leave to other disciplines. We need to find a way to encourage such research within the statistical community.

And more important, a very insightful view of how to use existing code in a different way is great statistical computing research but doesn't necessarily provide any code. As we move more and more towards component based software, the focus will be on the way we glue them together."

Col: "Software is a major communication device in the electronically connected global community. Making software available on the web puts a methodology into the public domain in a usable form. Software is often the first entry point for analysts to new methodology, and ideally encourages the user to learn more about the methods."

Kel: "Making software available on the web should count as a bean."

Kel: "Where we can help is:

- to have a statement by a top professional organization like ASA that explicitly identifies software development as a form of research and scholarly work worthy of reward and consideration in promotion decisions that departments can use to back up using this as part of their evaluation.
- to recommend in broad terms how departments can put together evaluations of software by obtaining peer reviews of the software from qualified experts in the field.
- to come up with a mechanism that can help departments identify appropriate individuals who can be commissioned to write peer reviews of software contributions that are part of promotion cases."

Ric: "This suggests that we do need to make a case for software development. If so, then I fully agree with it. But this would require that we define what software development (for the ASA) is."

Rob says: "Evaluating the Chambers award submissions is really tough. The software is usually part of a student's PhD work, so it's very specialized. The letters that accompany the software almost never mention the software design, or its quality or degree of difficulty; the advisors may talk about usability or impact, but often they're more interested in discussing the mathematics (or in their own research!) than anything else. Finally, the judges don't have explicit guidelines, so the judging process is very intuitive and subjective. A different set of judges could easily rank the work very differently."

Cam: "One thing that comes to mind is that I don't think we would give a seminar that describes the details of the software and its configuration and its documentation. Instead, we would pose the problem the software addresses and show how it solves that problem, how it can be used for other problems, how it relates to other software in that domain and other domains that form other stages of the problem. In other words, we describe the high-level concept, not the details of the implementation. And that is a difference between the 'code' and the idea or the research."

Col: "Why isn't it ok to discuss the software structure in a seminar? Traditionally it been ok to discuss a mathematical proof in a seminar, although it is now much preferred to listen to an outline of a proof rather than to the tedious details. Full proofs can be read at leisure at a later point. It seems appropriate that issues of software design, and particularly difficult parts to code, should be communicated in a seminar. It may even be good to show code fragments sparingly. How else does the community learn about good coding practices?

But back to evaluating software work as an expression of statistical computing research... It is important to have a framework or definitions from which to evaluate the work. A framework gives people a place to start, and a context in which to assess the relevance of the work.

In evaluations for grants, promotion and awards there are usually standards to follow. They are usually not as detailed as a checklist might be. Sometimes evaluation takes the form of an 'expert' panel discussion with very few guidelines, and then there is a danger that a strong opinion can determine the outcome of the committee's decision. A checklist can constrain and divert the discussion. They can be good and bad, but they are most useful when the committee does not have much expertise in the area. That is the case in statistical computing so having some areas to focus the conversation can be helpful.

One would never start a paper without telling the reader what it was about in a general sense first. How can we expect software to be evaluated if we don't define what we mean by software? As an evolving new, yet old, discipline it is fruitful to re-define the parameters from those that defined statistical computing in the past."

Summary: So a year after the workshop we are still debating aspects of statistical computing. The purpose of this document is to raise the awareness, communicate some of the discussion and open the discussion to the statistical computing and graphics community. There are some deliberately provocative statements, and there are numerous open questions:

What is statistical computing research? statistical graphics research?

How do we assess research involving software development?

How do we encourage academic departments to hire and be supportive of computationally literate faculty?

What measures can be used to evaluate statistical computing research?

If you would like to contribute to the discussion, provide answers to any of these questions or ideas related to the issues raised here please email the authors. We may provide a public forum if enough insightful email results.

Editorial

We are doing better this year in getting an issue out before the JSM '02 in New York City. We're hoping for 2 issues this year, a marked improvement on last year's effort. This issue contains information on the JSM '02 events for the Statistical Computing and Graphics sections, a continuation of the report from the Workshop on Statistical Computing in Academia, an article on bootstrapping and more articles on statistical software continuing the section from last issue. If you have articles on software to contribute to the next issue please submit them. We're also interested in articles from any area of statistical computing or graphics. Send them down!

Di has a request from the readership. This coming academic year she will be working with a graduate student to design a series of posters on statistical graphics. The idea is to profile a variety of graphics techniques, for example, scatterplot matrix, tours, parallel coordinate plots, on visually attractive posters. The posters would be made available to the members of the Statistical Graphics section for free and the general community for a small charge. We envision perhaps 5 posters in all. Di would like to hear suggestions for graphical methods that could be featured, explanations of techniques, or design ideas.

The editors would like to thank contributors to this issue and request submissions in the form of papers for the next newsletter, to appear at the end of 2002.

Susan Holmes and Dianne Cook

FROM OUR CHAIRS (Cont.). . .

Statistical Computing

CONTINUED FROM PAGE 1

As we work more often with the large genetic databases, interfacing our usual tools with these databases seems essential. Some statisticians have chosen `Perl/Bioperl` or `Python/Biopython` for their interfaces, both communicate easily with `R` through the `Omegahat` tools. Now all we need now is develop some GUI's so our users don't have to deal with all the intricacies of command line interfaces, and teach them some more statistics so they can understand all our wonderful pictures!

One of the most difficult part of my job as section chair has been interacting with the central administration of the ASA. It was extremely difficult this year to round up up a complete roster of possible volunteers for the various offices up for election and prepare for the JSM meetings.

I must thank all the volunteers who helped me put this list together, in particular Mark Hansen, our past chair, but also the many people I contacted when I was trying to build our list; those on the lists Alfred O. Hero III, Carey E. Priebe, Jun Liu, David Madigan, Tim Hesterberg, R. Webster West, Vincent J. Carey, Roger Koenker, and even some who we hope to invite in the near future- in particular, Chris Genovese.

As a plea to our members, please make yourself known to us if you would like to particpate more in our activities, we would love to have you on board!

We are all looking forward to the JSM and seeing you all again at the mixer. The meeting has been carefully planned so you can enjoy New York and all the wonderful sessions put together for us by Tim Hesterberg.

Now back to my laptop, happy that it can now hold a 1GB database and all the Python/R tools necessary to analyze the data, last night while I was working, the computer on my lap (as is wont of a laptop) I was even more worried than usual that the data had become too much to handle, there was a definite overflowing feeling coming from all those loops?? But no, it was only one of our small California quakes reminding me that there are other things around us that are just as exciting!

Susan Holmes, Chair,
Statistical Computing Section,
`susan@stat.stanford.edu`

Statistical Graphics

CONTINUED FROM PAGE 1

make sense of this information in a way that leads to actions that create value. One obvious opportunity is to build visual analytic applications that package sophisticated analyses for broad usage within the technical community. It seems clear that there will be a need for many such applications over the next few years.

For me, I'm onto my next startup, Visintuit. There is a story behind the name that I'll save for another day. By the way, it seemed like Visintuit was the only domain name that wasn't taken. Our idea is to apply visualization and analytical techniques to understand a particular class of financial data. Why financial data? As Willy

Sutton said, "that's where the money is". Working for a startup is really hard. The best part is that there are no rules. The worst part, so far, is that we have no revenues, although we're moving nicely to change this. Life is fun and it's a great time to be doing statistical graphics.

I'm looking forward to seeing everyone in NYC this August. If you don't see me, it means that I may have failed to solve our revenue problem. Please consider getting involved with section activities. The health of our section, one of the largest in ASA, depends on volunteers.

Stephen G. Eick, Ph.D.
Co-founder and CTO Visintuit
eick@visintuit.com
630-778-0050

TOPICS IN STATISTICAL COMPUTING

An Introduction to the Bootstrap with Applications in R

A. C. Davison and **Diego Kuonen**

kuonen@statoo.com

Introduction

Bootstrap methods are resampling techniques for assessing uncertainty. They are useful when inference is to be based on a complex procedure for which theoretical results are unavailable or not useful for the sample sizes met in practice, where a standard model is suspect but it is unclear with what to replace it, or where a 'quick and dirty' answer is required. They can also be used to verify the usefulness of standard approximations for parametric models, and to improve them if they seem to give inadequate inferences. This article, a brief introduction on their use, is based closely on parts of Davison and Hinkley (1997), where further details and many examples and practicals can be found. A different point of view is given by Efron and Tibshirani (1993) and a more mathematical survey by Shao and Tu (1995), while Hall (1992) describes the underlying theory.

Basic Ideas

The simplest setting is when the observed data $y_1, \ldots, y_n$ are treated as a realisation of a random sample $Y_1, \ldots, Y_n$ from an unknown underlying distribution F. Interest is focused on a parameter θ, the outcome of applying the statistical functional $t(\cdot)$ to F, so $\theta = t(F)$. The simplest example of such a functional is the average, $t(F) = \int y\, dF(y)$; in general we think of $t(\cdot)$ as an algorithm to be applied to F.

The estimate of θ is $t = t(\hat{F})$, where $\hat{F}$ is an estimate of F based on the data $y_1, \ldots, y_n$. This might be a parametric model such as the normal, with parameters estimated by maximum likelihood or a more robust method, or the empirical distribution function (EDF) $\hat{F}$, which puts mass n^{-1} on each of the y_j. If partial information is available about F, it may be injected into $\hat{F}$. However $\hat{F}$ is obtained, our estimate t is simply the result of applying the algorithm $t(\cdot)$ to $\hat{F}$.

Typical issues now to be addressed are: what are bias and variance estimates for t? What is a reliable confidence interval for θ? Is a certain hypothesis consistent with the data? Hypothesis tests raise the issue of how the null hypothesis should be imposed, and are discussed in detail in Chapter 4 of Davison and Hinkley (1997). Here we focus on confidence intervals, which are reviewed in DiCiccio and Efron (1996), Davison and Hinkley (1997, Chapter 5) and Carpenter and Bithell (2000).

Confidence Intervals

The simplest approach to confidence interval construction uses normal approximation to the distribution of T, the random variable of which t is the observed value. If the true bias and variance of T are

$$\begin{aligned} b(F) &= \mathrm{E}(T \mid F) - \theta = \mathrm{E}(T \mid F) - t(F), \quad (1) \\ v(F) &= \mathrm{var}(T \mid F), \end{aligned}$$

then we might hope that in large samples

$$Z = \frac{T - \theta - b(F)}{v(F)^{1/2}} \dot{\sim} N(0,1);$$

the conditioning in (1) indicates that T is based on a random sample $Y_1, \ldots, Y_n$ from F. In this case an approximate $(1 - 2\alpha)$ confidence interval for θ is

$$t - b(F) - z_{1-\alpha} v(F)^{1/2}, \quad t - b(F) - z_\alpha v(F)^{1/2}, \quad (2)$$

where z_α is the α quantile of the standard normal distribution. The adequacy of (2) depends on F, n, and T and cannot be taken for granted.

As it stands (2) is useless, because it depends on the unknown F. A key idea, sometimes called the *bootstrap* or *plug-in principle*, is to replace the unknown F with its known estimate $\hat{F}$, giving bias and variance estimates $b(\hat{F})$ and $v(\hat{F})$. For all but the simplest estimators T these cannot be obtained analytically and so

simulation is used. We generate R independent bootstrap samples $Y_1^*, \ldots, Y_n^*$ by sampling independently from $\hat{F}$, compute the corresponding estimator random variables $T_1^*, \ldots, T_R^*$, and then hope that

$$b(F) \doteq b(\hat{F}) = \mathrm{E}(T \mid \hat{F}) - t(\hat{F}) \quad (3)$$

$$\doteq R^{-1} \sum_{r=1}^{R} T_r^* - t = \bar{T}^* - t, \quad (4)$$

$$v(F) \doteq v(\hat{F}) = \mathrm{var}(T \mid \hat{F}) \quad (5)$$

$$\doteq \frac{1}{R-1} \sum_{r=1}^{R} (T_r^* - \bar{T}^*)^2. \quad (6)$$

There are two errors here: statistical error due to replacement of F by $\hat{F}$, and simulation error from replacement of expectation and variance by averages. Evidently we must choose R large enough to make the second of these errors small relative to the first, and if possible use $b(\hat{F})$ and $v(\hat{F})$ in such a way that the statistical error, unavoidable in most situations, is minimized. This means using approximate pivots where possible.

If the normal approximation leading to (2) fails because the distribution of $T - \theta$ is not close to normal, an alternative approach to setting confidence intervals may be based on $T - \theta$. The idea is that if $T^* - t$ and $T - \theta$ have roughly the same distribution, then quantiles of the second may be estimated by simulating those of the first, giving $(1 - 2\alpha)$ *basic bootstrap* confidence limits

$$t - (T^*_{((R+1)(1-\alpha))} - t), \quad t - (T^*_{((R+1)\alpha)} - t),$$

where$T^*_{(1)} < \cdots < T^*_{(R)}$are the sorted T_r^*'s. When an approximate variance V for T is available and can be calculated from $Y_1, \ldots, Y_n$, *studentized bootstrap* confidence intervals may be based on $Z = (T - \theta)/V^{1/2}$, whose quantiles are estimated from simulated values of the corresponding bootstrap quantity $Z^* = (T^* - t)/V^{*1/2}$. This is justified by Edgeworth expansion arguments valid for many but not all statistics (Hall, 1992).

Unlike the intervals mentioned above, *percentile* and *bias-corrected adjusted* (BCa) intervals have the attractive property of invariance to transformations of the parameters. The percentile intervals with level $(1 - 2\alpha)$ is $(T^*_{((R+1)\alpha)}, T^*_{((R+1)(1-\alpha))})$, while the BCa interval has form $(T^*_{((R+1)\alpha')}, T^*_{((R+1)(1-\alpha''))})$, with α' and α'' cleverly chosen to improve the properties of the interval. DiCiccio and Efron (1996) describe the reasoning underlying these intervals and their developments.

The BCa and studentized intervals are second-order accurate. Numerical comparisons suggest that both tend to undercover, so the true probability that a 0.95 interval contains the true parameter is smaller than 0.95, and that BCa intervals are shorter than studentized ones, so they undercover by slightly more.

Bootstrapping in R

R (Ihaka and Gentleman, 1996) is a language and environment for statistical computing and graphics. Additional details can be found at `www.r-project.org`. The two main packages for bootstrapping in R are `boot` and `bootstrap`. Both are available on the 'Comprehensive R Archive Network' (CRAN, `cran.r-project.org`) and accompany Davison and Hinkley (1997) and Efron and Tibshirani (1993) respectively. The package `boot`, written by Angelo Canty for use within `S-Plus`, was ported to R by Brian Ripley and is much more comprehensive than any of the current alternatives, including methods that the others do not include. After downloading the package from CRAN and installing the package, one simply has to type

```
require(boot)
```

at the R prompt. Note that the installation could also performed within R by means of

```
install.packages(boot)
```

A good starting point is to carefully read the documentations of the R functions `boot` and `boot.ci`

```
?boot
?boot.ci
```

and to try out one of the examples given in the 'Examples' section of the corresponding help file. In what follows we illustrate their use.

Example

Figure 1shows data from an experiment in which two laser treatments were randomized to eyes on patients. The response is visual acuity, measured by the number of letters correctly identified in a standard eye test. Some patients had only one suitable eye, and they received one treatment allocated at random. There are 20 patients with paired data and 20 patients for whom just one observation is available, so we have a mixture of paired comparison and two-sample data.

```
blue <- c(4,69,87,35,39,79,31,79,65,95,68,
          62,70,80,84,79,66,75,59,77,36,86,
          39,85,74,72,69,85,85,72)
red <-c(62,80,82,83,0,81,28,69,48,90,63,
        77,0,55,83,85,54,72,58,68,88,83,78,
        30,58,45,78,64,87,65)
acui<-data.frame(str=c(rep(0,20),
           rep(1,10)),red,blue)
```

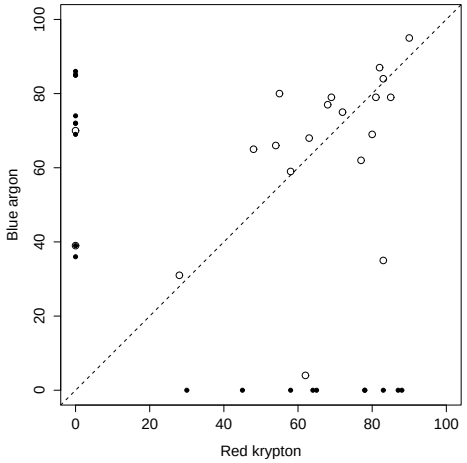


Figure 1:Paired (circles) and unpaired data (small blobs).

We denote the fully observed pairs $y_j = (r_j, b_j)$, the responses for the eyes treated with red and blue treatments, and for these n_d patients we let $d_j = b_j - r_j$. Individuals with just one observation give data $y_j = (?, b_j)$ or $y_j = (r_j, ?)$; there are n_b and n_r of these. The unknown variances of the d's, r's and b's are σ_d^2, σ_r^2 and σ_b^2.

For illustration purposes, we will perform a standard analysis for each. First, we could only consider the paired data and construct the classical Student-t 0.95 confidence interval for the mean of the differences, of form $\bar{d} \pm t_{n-1}(0.025)s_d/n_d^{1/2}$, where $\bar{d} = 3.25$, s_d is the standard deviation of the d's and $t_{n-1}(0.025)$ is the quantile of the appropriate t distribution. This can be done in R by means of

```
> acu.pd <- acui[acui$str==0,]
> dif <- acu.pd$blue-acu.pd$red
> n <- nrow(acu.pd)
>tmp<-qt(0.025,n-1)*sd(dif)/sqrt(n)
> c(mean(dif)+tmp, mean(dif)-tmp)
[1] -9.270335 15.770335
```

But a Q-Q plot of the differences looks more Cauchy than normal, so the usual model might be thought unreliable. The bootstrap can help to check this. To perform a nonparametric bootstrap in this case we first need to define the *bootstrap function*, corresponding to the algorithm $t(\cdot)$:

```
acu.pd.fun <- function(data, i){
 d <- data[i,]
 dif <- d$blue-d$red
 c(mean(dif), var(dif)/nrow(d)) }
```

A set of $R = 999$ bootstrap replicates can then be easily obtained with `acu.pd.b<-boot(acu.pd,acu.pd.fun,R=999)` The resulting nonparametric 0.95 bootstrap confidence intervals can be calculated as shown previously or using directly

```
> boot.ci(acu.pd.b,
 type=c("norm","basic","stud"))
...
Normal         Basic           Studentized
(-8.20,14.95) (-8.10,15.05)  (-8.66,15.77)
```

The normal Q–Q plot of the $R = 999$ replicates in the left panel of Figure 2underlines the fact that the Student-t and the bootstrap intervals are essentially equal.

An alternative is to consider only the two-sample data and compare the means of the two populations issuing from the patients for whom just one observation is available, namely

```
acu.ts<- acui[acui$str==1,]
```

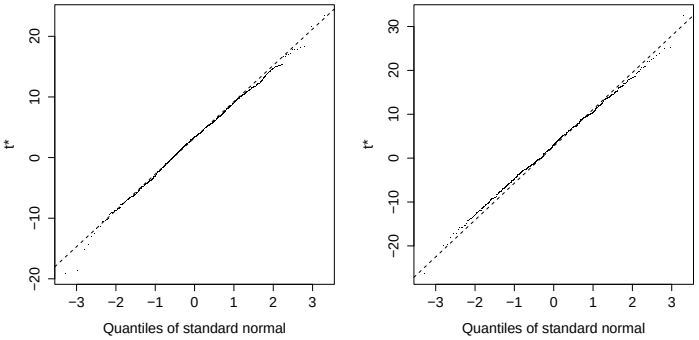


Figure 2:Normal Q–Q plots of bootstrap estimate t^*.
Left: for the paired analysis.
Right: for the two-sample analysis.

The classical normal 0.95 confidence interval for the difference of the means is $(\bar{b} - \bar{r}) \pm z_{0.025}(s_b^2/n_b + s_r^2/n_r)^{1/2}$, where s_b and s_r are the standard deviations of the b's and r's, and $z_{0.025}$ is the 0.025 quantile of the standard normal distribution.

```
>acu.ts <- acui[acui$str==1,]
>dif <- mean(acu.ts$blue)-mean(acu.ts$red)
>tmp <- qnorm(0.025)*
     sqrt(var(acu.ts$blue)/nrow(acu.ts)+
     var(acu.ts$red)/nrow(acu.ts))
> c(dif+tmp, dif-tmp)
[1] -13.76901  19.16901
```

The obvious estimator and its estimated variance are

$$t = \bar{b} - \bar{r}, \quad v = s_b^2/n_b + s_r^2/n_r,$$

whose values for these data are 2.7 and 70.6. To construct bootstrap confidence intervals we generate

$R = 999$ replicates of t and v, with each simulated dataset containing n_b values sampled with replacement from the bs and n_r values sampled with replacement from the rs. In R :

```
y<-c(acui$blue[21:30],acui$red[21:30])
acu<-data.frame(col=rep(c(1,2),c(10,10)),y)
acu.ts.f <- function(data, i){
d <- data[i,]
m <- mean(d$y[1:10])-mean(d$y[11:20])
v <- var(d$y[1:10])/10+var(d$y[11:20])/10
c(m, v) }
acu.ts.boot<-boot(acu,acu.ts.f,R=999,
              strata=acu$col)
```

Here `strata=acu$col` ensures stratified simulation. The Q–Q plot of these 999 values in the right panel of Figure 2 is close to normal, and the bootstrap intervals computed using `boot.ci` differ little from the classical normal interval.

We now combine the analyses, hoping that the resulting confidence interval will be shorter. If the variances σ_d^2, σ_r^2 and σ_b^2 of the ds, rs and bs were known, a minimum variance unbiased estimate of the difference between responses for blue and red treatments would be

$$\frac{n_d\bar{d}/\sigma_d^2 + (\bar{b}-\bar{r})/(\sigma_b^2/n_b + \sigma_r^2/n_r)}{n_d/\sigma_d^2 + 1/(\sigma_b^2/n_b + \sigma_r^2/n_r)}.$$

As σ_d^2, σ_r^2 and σ_b^2 are unknown, we replace them by estimates, giving estimated treatment difference and its variance

$$\begin{aligned} t &= \frac{n_d\bar{d}/\hat{\sigma}_d^2 + (\bar{b}-\bar{r})/(\hat{\sigma}_b^2/n_b + \hat{\sigma}_r^2/n_r)}{n_d/\hat{\sigma}_d^2 + 1/(\hat{\sigma}_b^2/n_b + \hat{\sigma}_r^2/n_r)}, \\ v &= \left\{n_d/\hat{\sigma}_d^2 + 1/(\hat{\sigma}_b^2/n_b + \hat{\sigma}_r^2/n_r)\right\}^{-1}. \end{aligned}$$

Here $t = 3.07$ and $v = 4.873^2$, so a naive 0.95 confidence interval for the treatment difference is $(-6.48, 12.62)$.

One way to apply the bootstrap here is to generate a bootstrap dataset by taking n_d pairs randomly with replacement from $\hat{F}_y$, n_b values with replacement from $\hat{F}_b$ and n_r values with replacement from $\hat{F}_r$, each resample being taken with equal probability:

```
acu.f <- function(data, i){
d <- data[i,]
m <- sum(data$str)
if(length(unique((i)==(1:nrow(data))))!=1){
d$blue[d$str==1]<-sample(d$blue,size=m,T)
d$red[d$str==1]<-sample(d$red,size=m,T)}
dif<- d$blue[d$str==0]-d$red[d$str==0]
d2 <- d$blue[d$str==1]
d3 <- d$red[d$str==1]
v1 <- var(dif)/length(dif)
v2 <-var(d2)/length(d2)+var(d3)/length(d3)
v <- 1/(1/v1+1/v2)
c((mean(dif)/v1+(mean(d2)-mean(d3))/v2)*v,v)}
acu.b<-boot(acui,acu.f,R=999,strata=acui$str)
boot.ci(acu.b,type=c("norm","basic","stud",
    "perc","bca"))
```

giving all five sets of confidence limits. The interested reader can continue the analysis.

Regression

A linear regression model has form $y_j = x_j^{\mathrm{T}}\beta + \varepsilon_j$, where the (y_j, x_j) are the response and the $p \times 1$ vector of covariates for the jth response y_j. We are usually interested in confidence intervals for the parameters, the choice of covariates, or prediction of the future response y_+ at a new covariate x_+. The two basic resampling schemes for regression models are

- *resampling cases* $(y_1, x_1), \ldots, (y_n, x_n)$, under which the bootstrap data are

  $$(y_1, x_1)^*, \ldots, (y_n, x_n)^*,$$

  taken independently with equal probabilities n^{-1} from the (y_j, x_j), and
- *resampling residuals*. Having obtained fitted values $x_j^T\hat{\beta}$, we take ε_j^* randomly from centred standardized residuals $e_1, \ldots, e_n$ and set

  $$y_j^* = x_j^T\hat{\beta} + \varepsilon_j^*, \quad j = 1, \ldots, n.$$

Under case resampling the resampled design matrix does not equal the original one. For moderately large data sets this doesn't matter, but it can be worth bearing in mind if n is small or if a few observations have a strong influence on some aspect of the design. If the wrong model is fitted and this scheme is used we get an appropriate measure of uncertainty, so case resampling is in this sense robust. The second scheme is more efficient than resampling pairs if the model is correct, but is not robust to getting the wrong model, so careful model-checking is needed before it can be used. Either scheme can be stratified if the data are inhomogeneous. In the most extreme form of stratification the strata consist of just one residual; this is the *wild bootstrap*, used in non-parametric regressions.

Variants of residual resampling needed for generalized linear models, survival data and so forth are all constructed essentially by looking for the exchangeable aspects of the model, estimating them, and then resampling them. Similar ideas also apply to time series models such as ARMA processes. Additional examples and further details can be found in Davison and Hinkley

(1997, Chapters 6–8). We now illustrate case and residual resampling.

The `survival` data (Efron, 1988) are survival percentages for rats at a succession of doses of radiation, with two or three replicates at each dose; see Figure 3. The data come with the package `boot` and can be loaded using

```
> data(survival)
```

To have a look at the data, simply type `survival` at the R prompt. The theoretical relationship between survival rate (`surv`) and dose (`dose`) is exponential, so linear regression applies to

$$x = \texttt{dose}, \quad y = \log(\texttt{surv}).$$

There is a clear outlier, case 13, at $x = 1410$. The least squares estimate of slope is -59×10^{-4} using all the data, changing to -78×10^{-4} with standard error 5.4×10^{-4} when case 13 is omitted.

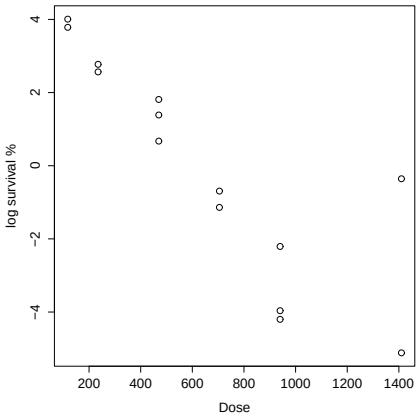


Figure 3:Scatter plot of `survival` data.

To illustrate the potential effect of an outlier in regression we resample cases, using

```
surv.fun <- function(data, i){
  d <- data[i,]
  d.reg <- lm(log(d$surv)~d$dose)
  c(coef(d.reg))   }
surv.boot<-boot(survival,surv.fun,R=999)
```

The effect of the outlier on the resampled estimates is shown in Figure 4, a histogram of the $R = 999$ bootstrap least squares slopes $\hat{\beta}_1^*$. The two groups of bootstrapped slopes correspond to resamples in which case 13 does not occur and to samples where it occurs once or more. The resampling standard error of $\hat{\beta}_1^*$ is 15.6×10^{-4}, but only 7.8×10^{-4} for samples without case 13.

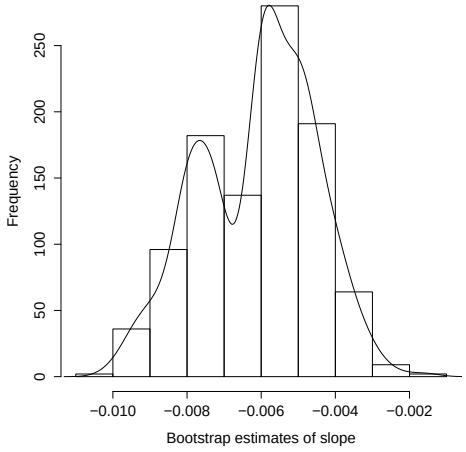


Figure 4: Histogram of bootstrap estimates of slope $\hat{\beta}_1^*$ with superposed kernel density estimate.

A jackknife-after-bootstrap plot (Efron, 1992; Davison and Hinkley, 1997, Section 3.10.1) shows the effect on $T^* - t$ of resampling from datasets from which each of the observations has been removed. Here we expect deletion of case 13 to have a strong effect, and Figure 5obtained through

```
> jack.after.boot(surv.boot, index=2)
```

shows clearly that this case has an appreciable effect on the resampling distribution, and that its omission would give much tighter confidence limits on the slope.

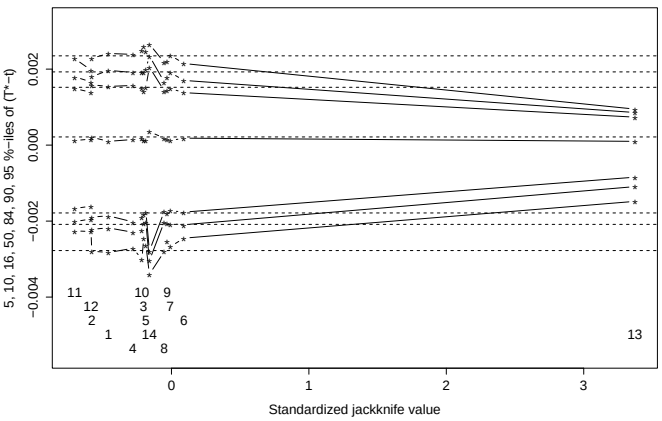


Figure 5: Jackknife-after-bootstrap plot for the slope. The vertical axis shows quantiles of $T^* - t$ for the full sample (horizontal dotted lines) and without each observation in turn, plotted against the influence value for that observation.

The effect of this outlier on the intercept and slope when resampling residuals can be assessed using

`sim=parametric` in the `boot` call. The required R code is:

```
fit <-lm(log(survival$surv)~survival$dose)
res <- resid(fit)
f <- fitted(fit)
surv.r.mle<- data.frame(f,res)
surv.r.fun<-function(data)
       coef(lm(log(data$surv)~data$dose))
surv.r.sim <- function(data, mle){
data$surv<-exp(mle$f+sample(mle$res,T))
            data                  }
surv.r.boot<- boot(survival,surv.r.fun,
              R=999,sim="parametric",
     ran.gen=surv.r.sim,mle=surv.r.mle)
```

Having understood what this code does, the interested reader may use it to continue the analysis.

Discussion

Bootstrap resampling allows empirical assessment of standard approximations, and may indicate ways to fix them when they fail. The computer time involved is typically negligible — the resampling for this article took far less than the time needed to examine the data, devise plots and summary statistics, and to code (and check) the simulations.

Bootstrap methods offer considerable potential for modelling in complex problems, not least because they enable the choice of estimator to be separated from the assumptions under which its properties are to be assessed. In principle the estimator chosen should be appropriate to the model used, or there is a loss of efficiency. In practice, however, there is often some doubt about the exact error structure, and a well-chosen resampling scheme can give inferences robust to precise assumptions about the data.

Although the bootstrap is sometimes touted as a replacement for 'traditional statistics', we believe this to be misguided. It is unwise to use a powerful tool without understanding why it works, and the bootstrap rests on 'traditional' ideas, even if their implementation via simulation is not 'traditional'. Populations, parameters, samples, sampling variation, pivots and confidence limits are fundamental statistical notions, and it does no-one a service to brush them under the carpet. Indeed, it is harmful to pretend that mere computation can replace thought about central issues such as the structure of a problem, the type of answer required, the sampling design and data quality. Moreover, as with any simulation experiment, it is essential to monitor the output to ensure that no unanticipated complications have arisen and to check that the results make sense, and this entails understanding how the output will be used. Never forget: *the aim of computing is insight, not numbers*; *garbage in, garbage out*.

References

Carpenter, J. and Bithell, J. (2000). Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Statistics in Medicine*, **19**, 1141–1164.

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Methods and their Application*. Cambridge University Press.
(`statwww.epfl.ch/davison/BMA/`)

DiCiccio, T. J. and Efron, B. (1996). Bootstrap confidence intervals (with Discussion). *Statistical Science*, **11**, 189–228.

Efron, B. (1988). Computer-intensive methods in statistical regression. *SIAM Review*, **30**, 421–449.

Efron, B. (1992). Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society series B*, **54**, 83–127.

Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. New York: Chapman & Hall.

Hall, P. (1992). *The Bootstrap and Edgeworth Expansion*. New York: Springer-Verlag.

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

Shao, J. and Tu, D. (1995). *The Jackknife and Bootstrap*. New York: Springer-Verlag.

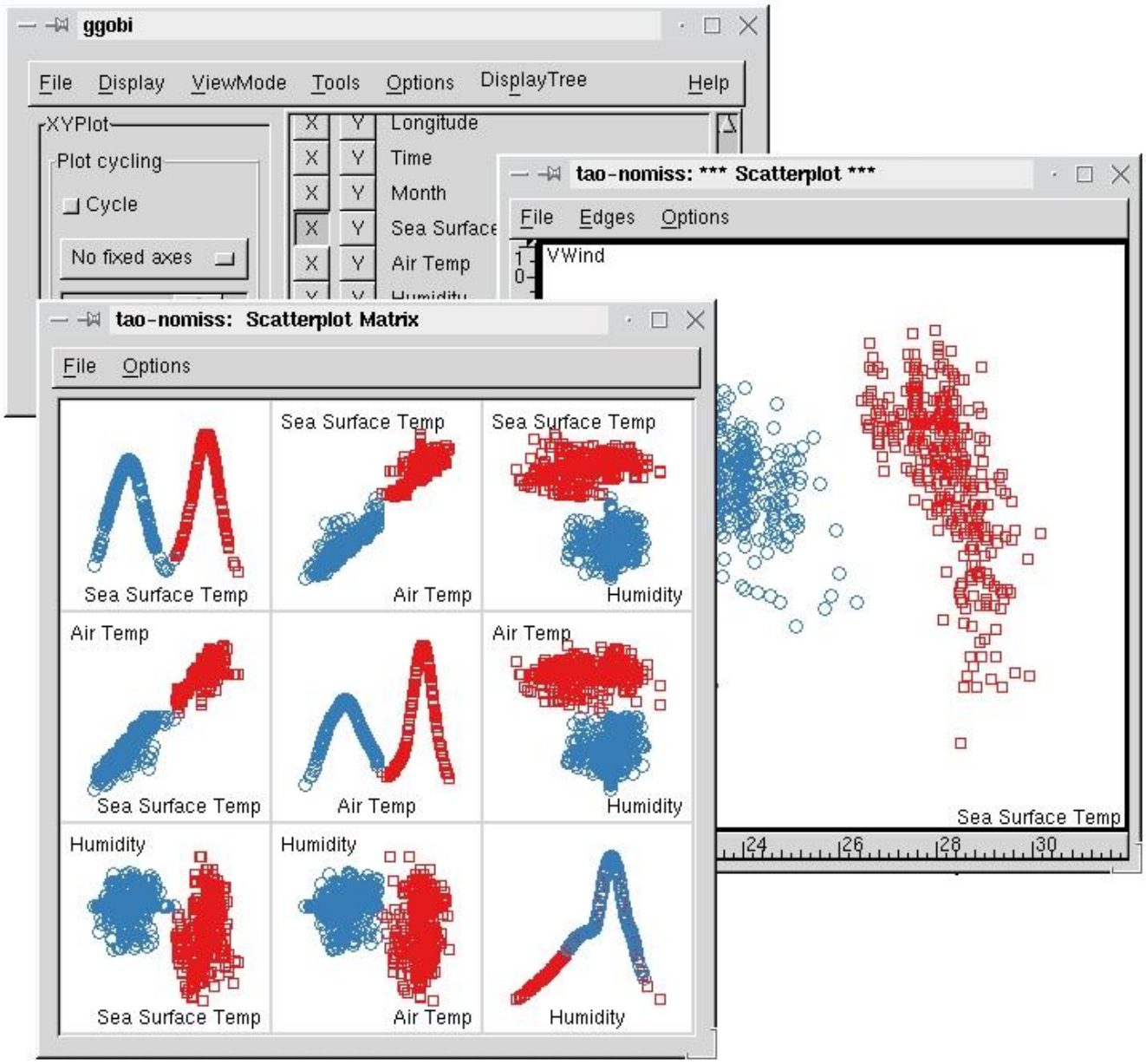
SOFTWARE PACKAGES

GGobi

Deborah F. Swayne, AT&T Labs – Research

`dfs@research.att.com`

GGobi is a new interactive and dynamic software system for data visualization, the result of a significant redesign of the older XGobi system (Swayne, Cook and Buja, 1992; Swayne, Cook and Buja, 1998), whose development spanned roughly the past decade. GGobi differs from XGobi in many ways, and it is those differences that explain best why we have undertaken this redesign.



GGobi's appearance: GGobi's appearance has changed in several ways: 1) It uses a different graphical toolkit with a more contemporary look and feel and a larger set of components. The new toolkit is called GTK+, which explains the initial G in GGobi. 2) With XGobi, there is in general a single plot per process; to look at multiple views of the same data, one launches multiple processes. A single GGobi session can support multiple plots (which may represent multiple datasets), and a single process can support multiple independent GGobi sessions. 3) XGobi's display types are essentially a single scatterplot and a subordinate parallel coordinate plot, but GGobi supports several types of plots as first class citizens: scatterplots, parallel coordinate plots, scatterplot matrices, and time series plots.

Other changes in GGobi's appearance and repertoire of tools include an interactive color lookup table manager, the ability to add variables on the fly, a new interface for view scaling (panning and zooming), and a redesign of the tour code.

Portability: A major advantage of using the new toolkit (GTK+) is portability. It originates in the Linux community, but it has been ported to Microsoft Windows and is being ported to the Macintosh. It is complicated to run XGobi on a machine running Windows, because it requires the installation an X Window System server. GGobi, on the other hand, runs directly under Windows.

GGobi's data format: GGobi's data format has been

extended significantly from that of XGobi. To describe a set of data for the older XGobi, one creates a set of files with a common base name, with the data in one file, and other files for the labels, colors, glyphs, and so on. GGobi continues to support this scheme in a limited way, but its new format uses a single file in XML, the Extensible Markup Language, which is emerging as a standard language for specifying structured documents and data formats.

The use of a single file aids consistency of the different elements of the input, making it easier to validate and maintain. An XML document looks a bit similar to an HTML document, but it allows one to introduce new markup elements. The use of XML in GGobi allows complex characteristics and relationships in data to be specified. For example, multiple datasets can be entered in a single XML file, and specifications can be included for linking them. Using the XML format, GGobi can read compressed files and can read files over the network.

Interoperability: While GGobi is a stand–alone application, it has been designed and constructed as a programming library so that direct manipulation, dynamic visualization functionality can be embedded within other applications. It has an Application Programming Interface (API) which developers can use to integrate the GGobi functionality with other code. At the highest level, there are three different approaches to integrating GGobi with other software, each of which is described in more detail in Symanzik et al (2002). These approaches create an environment that encourages others to experiment with innovations in data visualization without the need to build an entire system from scratch.

Embedding GGobi within other Applications: In this approach, we treat GGobi as a programming library and allow its data structures to be compiled into other software to create customized applications. When GGobi is embedded in this way, it can be controlled using programming interfaces from other languages such as Java, Perl, Python and S.

Extending GGobi: The use of modular plugins allows one to dynamically load code into a running GGobi. Using this approach, programmers can add functionality to GGobi without having to dig deeply into the code, and they can share their extensions easily with others. Various authors have been experimenting with plugins to communicate with a database, perform graph layout, and compute a minimum spanning tree, among others.

Distributed/Remote Access: The client/server architecture allows one to create GGobi as a server process offering a variety of methods to control, query and modify the session. Other applications can invoke these methods, even across different machines.

Getting GGobi: The GGobi code, sample data sets, and documentation can be found on www.ggobi.org.

References

Swayne, D. F., Cook, D. and Buja, A. (1992). XGobi: Interactive Dynamic Graphics in the X Window System with a Link to S, In *American Statistical Association 1991 Proceedings of the Section on Statistical Graphics*, 1–8.

Swayne, D. F., Cook, D. and Buja, A. (1998). XGobi: Interactive Dynamic Data Visualization in the X Window System, *Journal of Computational and Graphical Statistics*, **7**(1):113–130.

Symanzik, J., Swayne, D. F. and Temple Lang, D. and Cook, D. (2002) Software Integration for Multivariate Exploratory Spatial Data Analysis, In *New Tools for Spatial Data Analysis: Proceedings of the Specialist Meeting*, Center for Spatially Integrated Social Science.

Making Trees Interactive with Klimt - A COSADA software project

Simon Urbanek, Antony R. Unwin, Department of Computeroriented Statistics and Data Analysis, Augsburg University

simon.urbanek@math.uni-augsburg.de, antony.unwin@math.uni-augsburg.de

Introduction

What do gardeners and statisticians have in common? They both prune trees. Trees in statistics often lack the visual beauty of their counterparts in nature but they offer valuable ways of displaying structure in datasets. Today many software packages allow us to grow trees using different algorithms. In order to be able to choose a possibly best model we need to work with the tree, analyze it and explore it. Static tree diagrams provide little information about the dataset, because it's very hard to display more information for every node without cluttering the tree and making it unreadable. Static trees display only one view of a tree, but dependencies or special cases can better be detected with multiple

views. Our prototype software KLIMT (Klassification - Interactive Methods for Trees) was developed to overcome these shortcomings and allow dynamic, interactive work with trees.

An example of a classification tree drawn by a standard software is given in Figure 1. The tree has been grown with the *tree* library of Rfrom a dataset containing the results of a clinical study about meningitis disease treatment. (Total number of cases is 138. The dataset consists of four different variables (ALTER, FIE, ZZLQ, GRA) plus an outcome (classification) variable. There are 28 cases of negative (no) and 110 cases of positive (yes) outcome.) The patients were classified by the success of treatment (class *yes* means successful, class *no* means unsuccessful). The goal is to find a good explanatory tree-model so it's possible to predict a target group that will positively respond to the treatment, based on the available variables.

Features

The first basic feature of KLIMT is the visualization of the tree. KLIMT offers multiple different views of the tree. The nodes are represented by rectangles either of equal size or proportional to the number of cases. The connecting lines can be either direct from a node to its children, or rectangular giving the tree a "rods-and-wires" look as in R. There are also two different ways of node arrangement. The terminal nodes (leaves) can be displayed at the same level (usually at the bottom of the tree) to allow direct visual comparison among them, or they can be arranged below their parents just like any other node. Labeling of the nodes can be switched on and off. KLIMT initially arranges the tree so it fits the working window and uses recursive partitioning of display space to avoid overlapping of nodes as far as possible and to give the tree a "natural" look. The user can still move each node or entire branches as he pleases, constraint mode is available as well. The entire tree can be rotated by 90 degrees to make analysis of tall trees easier, because computer screens usually have more space horizontally then vertically. Visualization of the tree includes the criterion used for growing the tree. For example if the tree was built using deviance, circles proportional to the deviance gain for a split and rectangles proportional to the remaining deviation for a leaf are displayed. This method identifies splits with too small a gain or leaves with too big a residual deviance. KLIMT makes pruning intuitive. Selecting a node and issuing the "prune" command causes all children of the node to be removed from the tree. A small plus-sign will indicate that pruning has been done here and a simple click on that symbol will cause the pruned branches to be unfolded again.

With tree analyses it's also crucial to understand the underlying dataset. A first step to find out more about the cases is to use interactive querying of nodes. KLIMT offers two levels of queries. Simple queries provide information about the number of cases and percentage for each class in the node, as well as the total number of cases. Extended queries add the latest split, the node classification, the deviance and deviance gain. Besides queries, KLIMT provides standard plots of the data (histograms, bar charts, scatterplots, box plots) and some more sophisticated plots, such as treemaps, fluctuation diagrams, enhanced scatterplots and spineplots of leaves. The plots are linked and highlighting works across all displays of the same dataset. Combination of different selection modifiers, such as union or negation, makes selection very versatile. KLIMT offers a second form of selection based on nodes rather than cases. Selecting a node will cause all plots containing the splitting variable of the current node to visualize the position of the split. All nodes of the same class as the selected node are highlighted so dependencies can be more easily identified.

Finally KLIMT has an interface to the R/S/S-plus software. Whenever R is stated the entire family of software products R, S or S-plus is meant. The interface has been tested thoroughly mainly with R, but is intended to work with all other members of the family. This means that whenever the R library is enhanced, so is KLIMT, because we don't have to re-implement the tree algorithms. Moreover users can work in their familiar environment, study the dataset, prepare models and use all tools they are used to. As soon as they have a ready-to-explore tree they can start KLIMT with just one simple command from within R. KLIMT can use datasets directly from R and modify or return objects back to R if necessary. For example the user can pass a tree to KLIMT, modify the dataset or splits and tell KLIMT to grow a new tree in R, which will be displayed back in KLIMT. Having both trees in KLIMT now allows the user to use interactive methods to compare the trees and analyze them further.

Implementation

KLIMTis written in Java to ensure compatibility and availability. Only JDK 1.1 API was chosen as a requirement, because MacOS prior to 10 does not support JDK 1.2 or higher. The only exceptions are the 3rd-party portions as described below. This makes KLIMT available on all major platforms including Unix, Windows and MacOS. KLIMT uses two separate interfaces

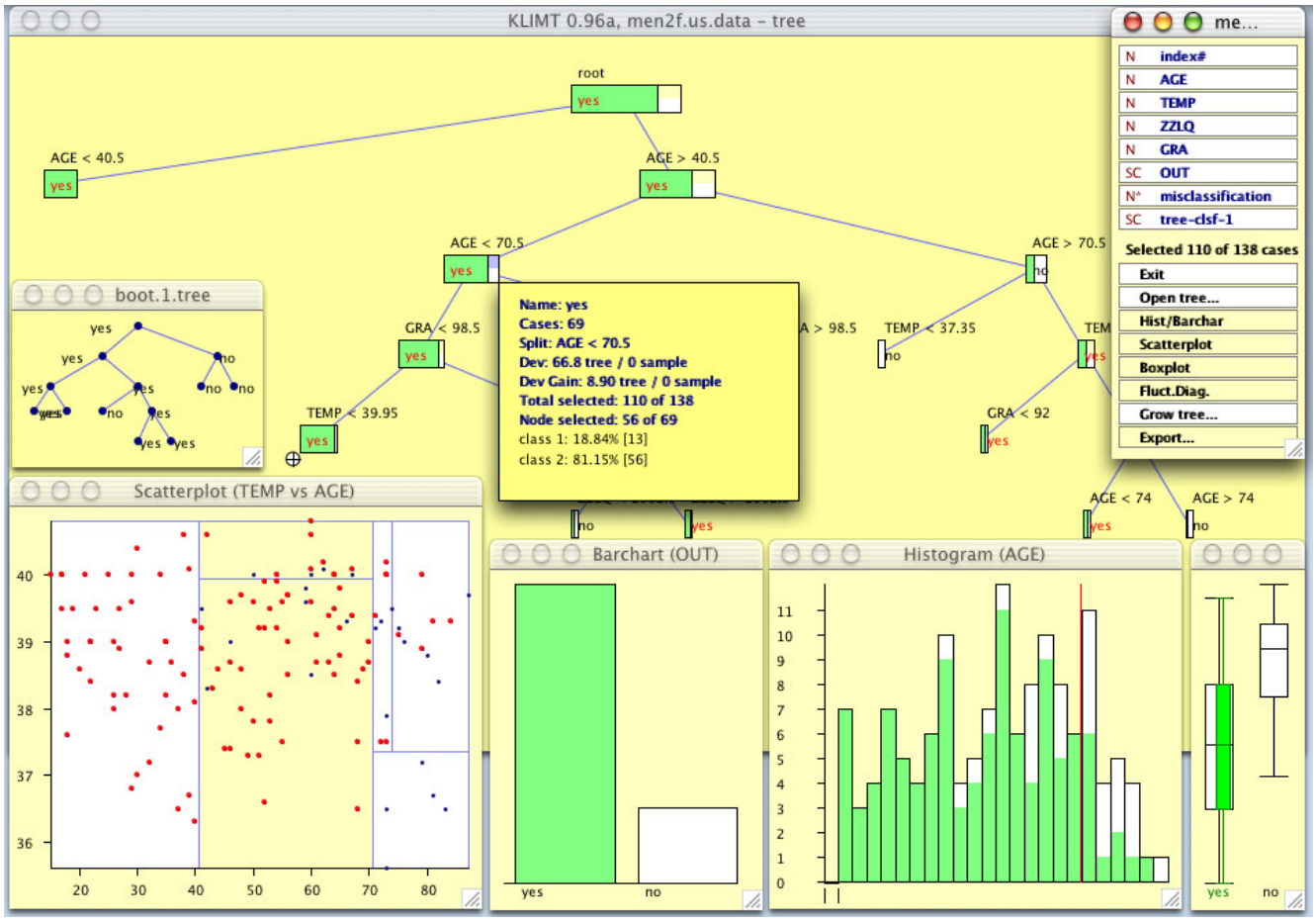


Figure 1: Classification tree in Klimt.

to R: files or *Omegahat* ($\hat{\Omega}$) SJava interface. The interfaces can be compared to methods of communication to R used by xgobi (file) and ggobi ($\hat{\Omega}$) as described in Temple Lang and Swayne (2001). The interface through files is less flexible but it still allows KLIMT to be launched from within R - the dataset and tree definition are passed through a text file between the systems. The advantage of this simple interface is that no special libraries are necessary for R and JDK 1.1 is sufficient. The $\hat{\Omega}$ interface allows KLIMT and R to share the same objects so any changes in either system are immediately active in the other one. $\hat{\Omega}$ makes Java classes of KLIMT available to R. This enables R to take control of KLIMT, e.g. the user can write functions in R that respond to actions in KLIMT, such as case query, node selection etc. R can also directly control display of KLIMT such as animating the tree and plots by selecting various cases over time. The disadvantage of the $\hat{\Omega}$ interface is that it requires JDK 1.2 or higher and is also quite new and under development so it's not fully tested yet.

Another criterion in designing KLIMT was to maintain the highest possible flexibility. All graphical parts such as tree display or plots were realized as `Canvas` over a specially developed `PoGraSS` (Portable Graphics SubSystem). The usage of `Canvas` instead of `Frame`, `Window` or even *Swing* classes allows very versatile application. More plots or trees can be displayed in one window, parts can overlap and even making KLIMT an applet is fairly easy. Simply by changing class types KLIMT can get a *Swing*-look instantly. `PoGraSS` is an abstract class with an API similar to `Graphics` but optimized to allow more independent output. Currently one implementation `PoGraSS-graphics` can act as a bridge between `PoGraSS` and the usual AWT `Graphics` class (used for displaying on screen) and another implementation `PoGraSSPS` produces *Encapsulated PostScript* (EPS) files which can be used for printing or inclusion in documents.

We tried to be consistent in achieving flexibility so KLIMT's dataset structure (`SVarSet`) supports variables (`SVar`) of any type, including any complex types such as objects or lists and `null` denoting a missing value. Automatic support for numerical variables

is included, some basic statistics such as minimum or maximum are created on-the-fly. A variable can be marked as categorical, in which case a list of categories is maintained by `SVar`. The current file parser supports strings and numerical types only (other types are too much format-dependent), but it makes intelligent guesses about the type and kind of the variable for regular ASCII files. Every `SVarSet` contains a marker (`SMarker`) which is the heart of the linked highlighting. A marker contains marks for all cases in the dataset both as a list of marks and a mask for fast access. Markings are of type `int` and support bitwise setting/clearing so multiple independent marks can be set (e.g. allowing multiple colors for brushing). `SMarker` also maintains a list of dependent classes that need to be notified on changes (such as plots).

Finally KLIMT uses a command broadcasting system to make the user interface easily configurable. Every user command can be issued through a central method, so for example a call `Tree1.run(this,"rotate");` rotates the workspace of `Tree1` by 90 degrees. Using only one method makes pass-through or broadcast calls possible (e.g. passing `ActionCommand`'s directly to the owner class's `run` method without processing).

Future plans

We want to improve the R-interface via $\hat{\Omega}$ and define a fixed API for communication with R. Better support for large trees has to be introduced, such as zoom and pan, navigation (separate window with an overview of the entire tree in miniature format). Flexible plots with linkable axes, i.e. multiple plots should be able to share common axis scales to support direct visual comparison, are also planned. We have some ideas about interactive pruning, where parameters for terminal nodes can be changed on-the-fly interactively with instant refresh. Fully interactive construction of trees is also under development. Finally visualization of small subgroups, i.e. when the size of a group on the screen is smaller that one pixel, has to be solved.

Recently our research focuses on the analysis of multiple trees and statistical forests. Several trees can be loaded in KLIMT with full support of visualization and linked highlighting among them. It is also possible to generate forest data, which allow interactive comparison and analysis of multiple tree models in KLIMT.

Conclusion

Static plots of trees don't generally contain enough information for a thorough model analysis. We developed KLIMT - an interactive software for exploratory data analysis of trees. Features include flexible visualization of trees, interactive queries, reordering of nodes, statistical plots, linked highlighting for all components, pruning and criterion visualization. KLIMT is a Java-application designed to be very flexible including universal internal data structures and two different interfaces to Rsoftware allowing users to use KLIMT directly from within Ror to supply datasets and tree information from other statistical applications using text files. Check KLIMT's homepage for further developments - `http://www.klimt-project.com/`

`http://www.omegahat.org/`

Temple Lang D., Swayne D. F.: *ggobi meets R: an extensible environment for interactive dynamic data visualization*, Proceedings of the 2nd International Workshop on Distributed Statistical Computing, TU Vienna (2001)

NEWS CLIPPINGS AND SECTION NOTICES

Programs for the Joint Statistical Meeting in New York City

Graphics Program at JSM 2002

A varied and exciting program awaits us at JSM 2002 in New York City. This year the Section on Statistical Graphics is the primary sponsor of three invited sessions, four topic contributed sessions, a regular contributed session, four roundtable luncheons, and a continuing education activity.

The invited program features a session organized by Debby Swayne on "Data Visualization in the Media: Infographics at the New York Times." The NYT Graphics Director and a NYT Graphics Editor will share with the audience the secrets of their trade, and Lee Wilkinson will discuss their presentations. Bill Eddy has crafted a session on "Graphics and Image Processing for Medical Images," in which three speakers will present several approaches to the analysis and display of spatio-temporal medical image data that are produced in massive quantities by current medical imaging technology. The third invited session, organized by Rob McCulloch, features four talks on "Graphical Methods in Bayesian Statistics," dealing with the use of graphics in Bayesian model building, validation, and fitting.

The first three topic contributed sessions will each feature five common-theme presentations, touching on both methodology and implementation issues. The

three sessions are "Graphical Methods for Categorical Data" (organized by Heike Hofmann), "Using Color in Statistical Graphs and Maps" (organized by Naomi Robbins), and "R Graphics" (organized by Paul Murrell). In addition, make sure you make time in your schedule to attend the topic contributed session organized by Lionel Galway that will feature the presentations by the four winners of the very successful Student Paper Competition run jointly by the Computing and Graphics Sections.

A regular contributed session on "Multidimensional Data and Model Visualization" will complete our contributed program. You will also find interesting graphics content in several other session that we co-sponsor, and in many Invited Technical Exhibits (on Sunday evening) and Contributed Posters (on Monday and Tuesday at noon).

The Graphics Section will also sponsor several fee events. On the continuing education front, we will sponsor the course "Introduction to Latent Class Mixture Models," taught by Jeroen Vermunt and Jay Magidson. On the roundtable luncheon front, thanks to the organizing efforts of our Program Chair Elect, Paul Murrell, we will sponsor discussions on "Visualization Methods For Categorical Data" (led by Michael Friendly), "Statistical Graphics for Biomedical, Pharmaceutical, and Epidemiologic Research," (led by Frank Harrell), "Technology for Statistical Reporting," (led by Duncan Temple Lang), and "Color as an Analytical Symbol for Data Representation." (led by Cynthia Brewer).

Last, but not least, the Joint Statistical Graphics and Computing Business Meeting (where business and fun are traditionally mixed together) is scheduled for Monday evening. Be sure to not miss out on this one! I look forward to seeing all of you in the Big Apple.

Mario Peruggia
Graphics Program Chair

Statistical Computing:
Huge Program for a Huge City

JSM 2002 in New York promises to be a huge meeting, and the Statistical Computing Section has a program to match, with 12 invited sessions, 19 topic contributed sessions, 6 regular contributed sessions, invited technical exhibits, an introductory overview lecture, and poster sessions. Not to mention our Stat Computing/Stat Graphics mixer, with some great door prizes, including tickets to a Broadway hit! Stat Computing is the primary sponsor for four invited sessions:
Model-based analyses of large-scale genetic and genomic data organized by Francesca Chiaromonte,
Data Mining in Practice organized by Matthias Schonlau
(get there early for this one; a record number of other sections signed on as co-sponsors),
Efficient MCMC organized by Gregory Warnes, and
Adaptive Statistical Methods in Imaging
organized by Jrg Polzehl.

We are also primary sponsor for 13 Topic Contributed sessions (3 on gene expression analysis alone!):
Data visualization, data mining and computational statistics organized by Xiangrong Yin,
Significance of observed changes in gene expression data organized by Dan Nettleton,
Assessing Differential Gene Expression from Microarray Studies organized by Mark Segal,
Modern Statistical Methods for Complex Problems organized by Claudia Becker,
Representation and Modeling of Visual Images organized by Yingnian Wu,
High-dimensional Integration organized by Frank Bretz,
Data Augmentation Methods organized by David van Dyk,
Recent Advances in Monte Carlo Methods organized by Jim Hobert,
Gene Linkage, Protein Structure, and Gene Expression Analysis organized by Ingo Ruczinski,
Statistical Computing: Tools, Interfaces, Technologies organized by B. Narasimhan,
Simulation and Monte Carlo organized by John Maryak, and
Statistical Methods in Computer Security organized by David Marchette.

Thanks to the organizers for putting together terrific sessions. For times and other information visit www.amstat.org/meetings, and make your own program!

Tim Hesterberg, Insightful Corp.
Statistical Computing, Program Chair.

SECTION OFFICERS

Statistical Graphics Section - 2002

Stephen G. Eick, Chair
(630) 778-0050
CTO Provalic Technologies,
eick@provalic.com

David A. James, Chair–Elect
(908) 582-3082
Lucent Bell Labs
dj@lucent.com

Deborah F. Swayne, Past–Chair
(973) 360-8423
AT&T Labs - Research
dfs@research.att.com

Mario Peruggia, Program Chair
(614) 292-0963
Ohio State University
peruggia@stat.ohio-state.edu

Paul Murrell, Program Chair–Elect
+64 9 373 7599 x5392
Auckland University, New Zealand
p.murrell@auckland.ac.nz

Dianne Cook, Newsletter Editor
(515) 294-8865
Iowa State University
dicook@iastate.edu

Thomas Lumley, Secretary/Treasurer
(206) 543-1044
University of Washington
thomas@biostat.washington.edu

Graham J. Wills, Publications Liaison Officer and Electronic Communication Liaison
(312) 651-3671
SPSS
gwills@spss.com

Jürgen Symanzik, Rep.(02-04) to Council of Sections
(435) 797-0696
Utah State University
symanzik@sunfs.math.usu.edu

Charles B. Roosen, Rep.(01-03) to Council of Sections
MathSoft, Inc.
roosen@statsci.com

Statistical Computing Section - 2002

Susan Holmes, Chair
(650) 725-1925
Stanford University
susan@stat.stanford.edu

Leland Wilkinson, Chair-Elect
(312) 651-3270
SPSS
leland@spss.com

Mark Hansen, Past–Chair
908-582-3869
Bell Laboratories
cocteau@bell-labs.com

Tim Hesterberg, Program Chair
(206) 283-8802
Insighful
timh@insightful.com

Mary Lindstrom, Program Chair–Elect
(608) 262-4812
Wisconsin University
lindstro@biostat.wisc.edu

Susan Holmes, Newsletter Editor (00-02)
650-725-1925
Stanford University
susan@stat.stanford.edu

Charles Kooperberg Secretary/Treasurer (02-03)
(206) 667-7808
clk@fhcrc.org

John F. Monahan, Publications Liaison Office
919-737-2541
North Carolina State University
monahan@stat.ncsu.edu

Thomas F. Devlin, Electronic Communication Liaison
(973) 655-7244
Montclair State University
devlin@mozart.montclair.edu

Lionel Galway, Awards Officer
(310) 393-0411, ext. 7957
RAND
galway@rand.org

Ranjan Maitra, Education Liaison
(410) 445-2436
University of Maryland
maitra@math.umbc.edu

John J. Miller, Education Liaison
(703) 993-1690
George Mason University
jmiller@gmu.edu

David M. Allen, Rep.(00-02) Council of Sections
(606) 257-6901
University of Kentucky
allen@ms.uky.edu

Elizabeth H. Slate, Rep.(00-02) to Council of Sections
607-255-9148
Cornell University
ehs1@cornell.edu

INSIDE

Statistical COMPUTING & GRAPHICS

The *Statistical Computing & Statistical Graphics Newsletter* is a publication of the Statistical Computing and Statistical Graphics Sections of the ASA. All communications regarding this publication should be addressed to:

Susan Holmes
Editor, Statistical Computing Section
Stanford University
Stanford, CA 94305
(650) 725-1925• FAX:(650) 725-8977
susan@stat.stanford.edu
http://www-stat.stanford.edu/∼susan

Dianne Cook
Editor, Statistical Graphics Section
Department of Statistics
Iowa State University
Ames, IA 50011-1210
(515) 294 8865 • FAX: (515) 294 4040
dicook@iastate.edu
www.public.iastate.edu/∼dicook

All communications regarding ASA membership and the Statistical Computing or Statistical Graphics Sections, including change of address, should be sent to:

American Statistical Association
1429 Duke Street
Alexandria, VA 22314-3402 USA
(703) 684-1221 • FAX (703) 684-2036
asainfo@amstat.org