Technology Innovations in Statistics Education

Volume 1, *Issue* 1 2007 *Article* 5

Computing and Introductory Statistics

Daniel Kaplan*

*Macalaster College, dtkaplan@gmail.com

Abstract

Much of the computing that students do in introductory statistics courses is based on techniques that were developed before computing became inexpensive and ubiquitous. Now that computing is readily available to all students, instructors can change the way we teach statistical concepts. This article describes computational ideas that can support teaching George Cobb's Three Rs of statistical inference: Randomize, Repeat, Reject.

Acknowledgements: The author's work in this area has been supported by grants from the Howard Hughes Medical Institute and the Keck Foundation. The author thanks two anonymous referees for helpful suggestions.

Suggested Citation:
Daniel Kaplan (2007) "Computing and Introductory Statistics", *Technology Innovations in Statistics Education*: Vol. 1: No. 1, Article 5.
http://repositories.cdlib.org/uclastat/cts/tise/vol1/iss1/art5

Computing and Introductory Statistics

Daniel T. Kaplan
Macalester College

1. INTRODUCTION

Here's a chicken-and-egg question: To learn introductory statistics effectively, what should our students know about computing?

We might claim, with reason, that we can already teach statistics effectively. We know that we can do this with the students and equipment that we already have and with the computer skills that our students already possess or can pick up in our statistics class.

On the other hand, if our students knew more about computing how might that change our conception of what it means to learn statistics effectively? Rather than asking what computation is needed to support the statistics that we teach, perhaps we should ask what computation is needed to support the statistics that *we want to teach*.

The last two decades are filled with examples where institutions adapted computers to what they were already doing and missed an opportunity to do much more. In the 1980s and 1990s librarians adapted computers to automate their title/author/subject card catalogs. Now, libraries have largely been supplanted by comprehensive, full-text searches and the electronic delivery and national archiving of content.

The 1980s music industry embraced a CD format which was designed to provide a physically compact version of the standard 30-minute "long-playing" (LP) album. They ignored available compression algorithms that would have expanded the capacity much beyond their LP conception. Now CD sales are collapsing while children carry around entire collections of music in their pockets, using devices such as the iPod based on the ideas of "search" and "playlist" rather than "album" and "shelf."

Similarly, statistics educators have embraced the calculator and computer to teach content that in the main would have been familiar in the 1970s.

A possible retort to that last statement points out that the statistics that was familiar in the 1970s is exactly what students, especially introductory college-level students, need to know. That might be correct, but there are hints even within our present curriculum that we are trying to push beyond what we have traditionally done and that a major obstacle to doing this is what students don't know about computation.

2. WHAT WE DO NOW.

A well documented and readily available indicator of what we currently do in introductory college-level statistics is the Advanced Placement statistics course. The AP test enables students to get college credit for course taken in high school. The number of students taking the AP statistics test has grown exponentially to about 100,000 in 2007; many more students take the course without taking the test. In my opinion, this success is well deserved since the AP curriculum is well designed for covering traditional topics. Here is what the AP *Course Description* (College Board 2006) says about "the use of technology,"

> Because the computer is central to what statisticians do, it is considered essential for teaching the AP Statistics course. However, it is not yet possible for students to have access to a computer during the AP Statistics Exam....
>
> If a graphing calculator is used in the course, its computational capabilities should include standard statistical univariate and bivariate summaries through linear regression. Its graphical capabilities should include common univariate and bivariate displays such as histograms, boxplots, and scatterplots. Students find calculators where data are entered into a spreadsheet format particularly easy to use. Ideally, students should have access to both computers and calculators for work in and outside the classroom.

Immediately following the one-page discussion of computing technology in the *AP Course Description* is the seven pages of formulas and probability tables that are made available to test takers: regression slopes and intercepts, standard errors; normal, t, and χ^2 distributions.

The College Board (2006) provides four sample syllabi for various ways of organizing the course. All four of the example statistics syllabi offered by the College Board emphasize the use of the graphing calculator. Computers, with software such as Minitab, JMP-Intro, Fathom, or special-purpose applets, are used "secondarily" or for "occasional assignments."

Perhaps unbeknownst to the College Board, most students are indeed bringing computers to the AP test session. The graphing calculators that are allowed in the session have the computational power of a full-fledged computer.

Computation has become ubiquitous. An illustration of this was given in a recent newspaper article about toys (Marriot 2007).The article quoted Caleb Chung, the inventor of the Furby, a popular talking toy from the late 1990s. Chung said, "The price of processing power has dropped to the floor. I can buy the equivalent of an Apple II processor for a dime."

What distinguishes the graphing calculator from the "computer" is not computational power but the man-machine interface. Calculators have a low-resolution display and a slow-to-use keypad. They use a rudamentary operator-stack command system: enter a number, press a command, enter a number, press a command, and so on.

With this interface, students can use calculators work that involves the input of a dozen or so numbers in each of two columns. This data format lends itself pretty well to the basic

operations required in AP statistics: display a histogram, boxplot or scatter plot, or compute a mean or standard deviation or straight-line Y versus X regression coefficients. So these operations are what gets taught if you use a calculator. Or, more precisely, this is what gets taught if you use a computer with an interface like a calculator.

An indication about what we would like to teach if we didn't have the interface problem is given by the AP sample syllabi. All four syllabi emphasize the use of the graphing calculator up to the time of the actual exam. This makes sense, because the calculator is what is available during the exam and a central goal of the AP course is to prepare students for the exam. But after the exam is given there are a few weeks left in the academic year.

After the exam, three of the AP syllabi move on to topics not covered on the exam: multiple regression and ANOVA. In every case, the calculator is no longer used for computational support for these topics. Instead, a computer package is used: both JMP-intro and Minitab are referred to in the sample AP statistics.

The computer, with its high resolution graphical display and random access spreadsheet grids, is much better suited to lengthy, multivariate data sets and to operations that cannot easily be specified with a button press, for example selection of a set of explanatory variables and a response variable or plotting of residuals.

The AP *Course Description* gives additional rationale for using computers rather than calculators:

> In addition to its use in data analysis, the computer facilitates the simulation approach to probability that is emphasized in the AP Statistics course. Probabilities of random events, probability distributions of random variables, and sampling distributions of statistics can be studied conceptually, using simulation. This frees the student and teacher from a narrow approach that depends on a few simple probabilistic models (College Board 2006, p. 9).

As an example of what it might mean to study the sampling distributions of statistics at an introductory level, I pose a problem drawn from the my experience in statistical consulting. The US federal government audits its contractors against the possibility that they discriminate illegally in hiring. One bureau, the US Department of Labor Office of Federal Contract Compliance Programs (OFCCP), describes how it calculates the statistical significance of an observed difference in hiring rates, for example, "Female vs. Male Hires for Management Trainee." The OFCCP worksheet (US Department of Labor 1993) is reproduced in the Appendix, but it boils down to a formula.

Suppose there are n_1 applicants from Group 1 of whom h_1 are hired, and similarly n_2 applicants with h_2 hires from group 2. Within each group, the probability of a random applicant being hired can be estimated as $p_1 = h_1/n_1$ and $p_2 = h_2/n_2$. Overall, the probability of a random applicant from either group being hired is $p = (h_1 + h_2)/(n_1 + n_2)$. The OFCCP

looks for a statistically significance difference between p_1 and p_2 using

$$\sigma = \frac{|p_1 - p_2|}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}\sqrt{p(1-p)}}. \quad (1)$$

According to the OFCCP regulations, finding $\sigma > 2$ for any comparison of two groups will subject the contractor to penalties and enforced corrective actions. (I have seen OFCCP impose stiff penalties when $2.0 < \sigma < 2.1$. The US Supreme Court (1977) has indicated that "a fluctuation of more than two or three standard deviations would undercut the hypothesis that decisions were being made randomly with respect to [a protected trait]" (*Hazelwood School District v United States* 1977).

There are several aspects of the OFCCP formula and its application that might be interesting to cover in an introductory statistics course: the denominator is a standard deviation of the difference between two proportions under the hypothesis that hiring rate is the same for the two groups; the observed difference in hiring rates is scaled by this standard deviation. The $\sigma > 2$ threshold corresponds to a significance level of 5%.

Another interesting aspect of the OFCCP formula is that it is wrong. It does not describe the sampling distribution of the difference in hiring rates in the situation to which it is applied. There are two reasons for this: (1) hires made in Group 1 and Group 2 are not independent, since the total number of hires is fixed; (2) OFCCP applies the formula to many sets of groups, and finds a violation if any one of the comparisons produces $\sigma > 2$.

Clearly the formula for σ is not the end of the story. We need to have a way to teach effectively why the formula is wrong and how a student might discover for him or herself that the formula is wrong. Even more important, we need to teach students how they might communicate with others about such issues. We prepare students for exams presumably because that helps them to learn, but the true test is whether our students can apply their knowledge in an authentic setting such as that provided by the OFCCP procedures.

3. THE "TYRANNY OF THE COMPUTABLE"

George Cobb (2005) provides a trenchant critique of the standard introductory course that illustrates the self-reinforcing origins of the introductory curriculum:

> **Question:** Why, then, is the t-test the centerpiece of the introductory statistics curriculum?
>
> **Answer:** The t-test is what scientists and social scientists use most often.
>
> **Question:** Why does everyone use the t-test?
>
> **Answer:** Because it's the centerpiece of the introductory statistics curriculum.

Cobb argues that the "logic of inference" should be at the core of the introductory curriculum and recommends that the permutation test, not the t-test, should be the "central paradigm for

inference." He points out that the t-test is so widely used because the computations involved were easily performed in the days before computers. Cobb quotes Fisher: "[T]he statistician does not carry out this very simple and very tedious process [of the permutation test], but his conclusions have no justification beyond the fact that they agree with those which could have been arrived at by this elementary method."

Computer hardware makes performing the permutation test a simple matter. If, as Fisher characterizes it, it is an "elementary method," why is it not taught in introductory statistics? The barriers (some of which are described by Duckworth and Stephenson (2002)) are not fundamentally computational; providing the necessary capabilities to either calculators or computers is straightforward. Instead, an important barrier relates to language or notation: the way we describe things.

When we describe the t-test or a confidence interval, we are often describing an algorithm whose individual steps can be presented in the algebraic or calculator notation familiar to high-school students or as look-up instructions. For example, here are the steps in finding the margin of error on a sample mean:

1. Find the sample mean: $m = \frac{1}{N}\sum_{i=1}^{N} x_i$.
2. Find the sample standard deviation: $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - m)^2}$.
3. Look up $t^\star$ for 95% confidence in a table of the t-distribution with $N-1$ degrees of freedom.
4. The margin of error is $\pm t^* s/\sqrt{N}$.

Steps 1, 2, and 4 involve arithmetic. Step 3 is a function evaluation. These are skills taught extensively in high school.

The corresponding bootstrapped confidence interval is straightforward, but involves different operations for which we have no standard algebraic or calculator notation: resample, collection.

By analogy to the traditional Three Rs of elementary school (reading, writing, arithmetic), Cobb (2007) lays out the "three Rs of inference":

> Randomize, Repeat, Reject.

We know how to support computationally a course that puts the t-test at the center, since it involves computing means, standard deviations, and square roots: familiar stuff for a calculator user. How do we support a Three-R course? None of these Rs are conveyed by algebraic or calculator notation.

4. CODE IS BAD NOTATION

When we add a new procedure to a calculator or a statistics package, we write in a special

language supported by that package. Such instructions are often called "code," and for good reason. Here is part of a macro written for Minitab to produce a bootstrapping confidence interval on the sample mean (Stephenson 1993)):

```
OneMeanCI n Data CC nreps
Mconstant n CC cl cu Lower Upper nreps I
Mcolumn Data Bootmean Resample

DO I= 1:nreps
     Sample n Data Resample;
     replace.
     Let Bootmean(I) = Mean(Resample)
ENDDO
```

An approach to the Three Rs that is based on notation like this is not likely to be comprehended by introductory statistics students. The macro is lengthy, complicated, and cryptic: it's computer code that mixes together irrelevant technical details of the Minitab macro language — `Mconstant`, `Mcolumn`, `Let` — with the essential structure of bootstrapping.

Good notation reveals structure; bad notation obscures it. But no notation can be clear if you haven't learned to read it and haven't learned the concepts that underlie it. Consider again the Formula 1 used by the the US Federal government in detecting hiring discrimination:

$$\sigma = \frac{|p_1 - p_2|}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}\sqrt{p(1-p)}}.$$

We wouldn't expect people who had never seen algebraic notation to understand this; why would we expect people who have never studied computation to understand something like the Minitab macro (which is not so nice notationally in any event)?

Although the algebraic notation of Formula 1 seems clear to those with good algebraic skills, it is mysterious and imposing to most people. Given that students are not comfortable with such notation, it forms a barrier to teaching statistics effectively. As an example of how far people can go to avoid using algebraic notation, see Appendix A for the way the algorithm behind Formula 1 is presented officially by the US Department of Labor (1993).

5. NOTATION AS A STARTING POINT

Let's start by example with one possible notation to support the Three Rs.

Suppose we have a data set such as might be used in studying hiring patterns, a table that gives various information about each person who applied for a job. The data have been stored in an object called `apps`. Typing the name of an object after the prompt (`>`) causes the object's value to be printed.

```
> apps
```

```
  ID        job sex race hired age         date
3233   clerical   F    H     Y  32   2007-03-05
3234    laborer   M    H     N  51   2007-03-05
3235   clerical   F    B     N  35   2007-03-06
3236 supervisor   F    H     N  32   2007-03-06
3237    service   F    W     N  19   2007-03-07
3238    service   F    H     N  27   2007-03-08
  ... and so on.
```

Even this trivial snipet of dialog exemplifies an important aspect of the notation: names are associated with values and the values can have a potentially complicated structure, in this case a table of data consisting of "variables" (the columns) and "cases" (the rows).

Suppose we want to know the mean of one of the variables, say the age of the applicants.

```
> mean( apps$age )
  35.4
```

In order to understand such notation, a student would need to know several things: `subset` is an *operator* that takes *arguments*. We can access one variable of the `apps` table using the `$` notation.

None of these things is obvious to an untrained reader, but none of them is difficult.

Once you know the syntax of the operator/argument notation and the names of a few operations, some things become self explanatory:

```
> sd( apps$age )
  10.4
> proportion( apps$hired )
      Y
   0.26
```

We need operators for Cobb's first R, Randomize. Here is one, that randomly samples (with replacement) from a collection:

```
> resample( apps$age )
  21 47 26 47 19 49 ... and so on
> resample( apps$age )
  44 51 44 29 37 47 ... and so on
```

We can build a bit on top of this:

```
> mean( resample( apps$age ) )
```

```
    37
> mean( resample( apps$age ) )
    35.4
```

Since `resample` is random (and with replacement), we got different results when we repeated the command.

Cobb's second R is Repeat. In plain English, we might naturally say, "Repeat *this* 10 times." Our notation need only convey the meaning of *this* and 10 as arguments:

```
> do(10) (mean(resample(apps$age)))
    34.7 37.0 37.0 34.7 31.7 36.3 33.2 34.2 35.8 38.3
```

The notation for "store this result" is straightforward:

```
> samps = do(500) (mean(resample(apps$age)))
```

Finally, we can find a standard error:

```
> sd(samps)
    2.59
```

I do not claim that this notation is the easiest way to provide a student with ability to do statistical calculations. For pure ease of use, it's hard to beat a spreadsheet GUI with a drop-down menu on each variable labeled "Bootstrap the Standard Error." But the point isn't to implement a calculation, it's to implement a calculation in a way that makes the structure and logic clear to a student and lets the student generalize a bit. Could a student look at the above few lines and figure out how one would find the standard error of a sample median or of a sample standard deviation? I think so. Could a student look at the lines and explain why the standard error is not the same as the sample standard deviation of the variable? Yes, so long as the student had been taught to see that `samps` is generated from `apps$age` but isn't exactly the same as it.

Let's return to the employment discrimination example. The basic logic of inference is that we need to compare the observed difference in hiring rates to that we would expect if there were no discrimination. What makes this a statistical problem is that we acknowledge that random fluctuations can cause a non-zero difference in hiring rates even when there is no discrimination.

To study this, we can create a world in which there is no hiring discrimination and see what sorts of fluctuations there are. In such a world the hiring decisions would be independent of the applicants' traits, for instance sex. It's a profound statistical concept that we can create such independence by randomizing the value of sex for each applicant. Teaching that concept isn't trivial, but the notation for doing this on the computer is a straightforward extension of what we have already seen.

Here are the observed hiring rates for men and women:

```
> proportion( hired, by=sex, data=apps )
      Y
F 0.286
M 0.227
```

The difference in rates is

```
> diff(proportion( hired, by=sex, data=apps ) )
        Y
M -0.0584
```

Let's take one trial from a world without discrimination:

```
> diff(proportion( hired, by=resample(sex), data=apps ) )
        Y
M -0.0994
```

The subtlety here is that we are resampling the sex variable on its own.

To compare the observed differences to see if they are plausibly from a world in which there is no discrimination, repeat the no-discrimination trial many times and see how often it happens that the non-discrimination world produces differences as big as those that we observed in the actual world.

```
> samps = do(1000) ( diff( proportion( hired, by=resample(sex), data=apps) ) )
> proportion( abs(samps) > abs(-0.0584) )
   TRUE
0.704
```

That last command shows an instance of Cobb's third R: Reject.

I don't want to imply that this notation must change what we teach, just that it supports teaching in a different way. If you want to teach χ^2 tests on two-way tables, do that. But perhaps your students would benefit from seeing what the χ^2 values look like when the connection between the two variables is severed by randomization.

Some will object that this notation is too difficult or too abstract, or that students can't be expected to enter commands without errors and that, when errors inevitably occur, students will not be able to diagnose and fix the mistake. This is a legitimate objection. Students can be confused by the details of the syntax: curly braces versus parentheses, the need to balance left and right brackets, and so on.

It's essentially an empirical issue whether such notation can be used effectively by mainstream students. My own experience at a selective liberal arts college, which has been successful, may not be representative of the difficulties faced by instructors at other types of schools. Still, I see no reason why this notation is harder than that successfully used by millions of high-school students on graphing calculators.

It's not just students. Many if not most instructors of introductory statistics are not familiar with computer language syntax. The past education didn't provide any such training and there is little or no mechanism for ongoing professional development. Unlike other professions which require strong technical skills in an evolving environment — medicine, engineering, aviation, accounting — statistics education has no licensing system that mandates ongoing training.

The history of personal computers suggests that point-and-click interfaces are more effective than command-line interfaces, not for technical reasons but because they can be operated effectively with little training. There are now several statistical software packages with easy-to-use point-and-click interfaces, e.g., FATHOM (Finzer 2007) or JMP (SAS Institute 2007) SPSS offers a "student version" (Prentice Hall 2007) which disables the command-line interface. Systems such as FATHOM offer the ability to write dynamic simulations which can be effective for teaching ideas such as sampling distributions, bootstrapping, and so on. But none of these systems provide a notation or user interface that supports Cobb's Three Rs: a way to assemble statistical operations in terms of these Rs.

An intermediate possibility — between the flexibility of command line statements and the user-robustness of point-and-click — is graphical programming languages such as Labview (National Instruments 2007) in which algorithms are built by connecting modules graphically: a kind of software breadboard. Perhaps we need to develop such modules to support statistical operations, e.g., a "select variable" module, a "resampling" module, and so on that could be connected to a "repeat this" module.

For the present, however, I think it important to focus on the computational concepts that we want students to learn, supporting this learning with already available software. Once we have consensus about the concepts, we can optimize the ease of use of software to help convey them.

6. THE BASICS OF COMPUTATION

In evaluating any notation for use in teaching statistics I look at two main criteria: First the notation should allow students to perform the necessary calculations; Second, the notation should highlight the logic of what is going on. Black-box macros or calculator button satisfy the first criterion, but they fail to satisfy the second.

In my experience, students can successfully learn the notation and concepts presented above by example through their work in a statistics class. The notation is implemented in R (2006), a free, high-level statistics package.Much of the notation is native R, but a few extensions have been written some of which are described in Appendix B. (The extension software file

is available as an auxiliary file to this article, `math155.r`, or by contacting the author.)

Students respond well to examples, but it's also important to identify and name the concepts underlying them. This helps us communicate and it helps students to generalize their learning to new settings. This certainly applies to new languages (even the Minitab macro!), but I find that it even applies to the effective use of graphical user interfaces.

To be effective users of computers in statistics, I believe that all students should be taught and learn these principles of computation:

1. **Computation is a transformation.** A computation takes one or more inputs and produces an output. `sqrt(10)` $\rightarrow$ 3.162 is an example, as is `resample(apps$age)` $\rightarrow$ $21, 47, 26, 47, 19, 49, ...$

2. **Algorithm.** A potentially complicated procedure can be specified as a collection of simpler procedures. The output of one computation can be used as the input to other computations. For instance, `mean(resample(apps$ages))`

3. **Syntax.** Syntax matters. For example, braces and parentheses must be used properly (e.g., they must be balanced, brackets for indexing may differ from brackets for indexing), `=` means something very different from `==`, ...

   Humans communicate largely using context — we can do a pretty good job satisfying basic needs even when we don't know the local language. The cost of this is ambiguity. The syntax of computer languages is a compromise between ease of use and the need to avoid ambiguity.

4. **Collection.** Sets of values can be collected together and the set can be dealt with as a whole or by accessing individual elements. Examples: `mean(apps$age)` or `sd(samps)`.

5. **Repetition.** An operation can be repeated many times and the individual results made part of a collection. `do(5)(mean(resample(apps$age)))`

6. **Assignment.** Numbers and other values can be stored for later access.

   ```
   samps = do(500)(mean(resample(apps$age)))
   hist(samps)
   sd(samps)
   ```

7. **Selection.** Values that meet a given criterion can be pulled out. For example, here's some notation for selecting applicants for laborer jobs:

   ```
   Lcases = subset( apps, Job=='laborer' )
   ```

   The second argument contains the answer to a question: which cases have a value of Job that matches the string "laborer".

8. **Abstraction.** An new operation can itself be created out of old ones and stored and even passed as an argument. For example, here is an operator that generates the sum of n dice:

```
roll.dice = function(n=2){ sum( resample(1:6,n) ) }
```

I don't have my students do much of this, but building operations exercises students' understanding of arguments. It also empowers students: they see there is nothing mysterious about creating a new operator.

7. WHEN, WHERE, AND WHO?

The point of the above is not to claim that it's easier for students to do resampling or permutation tests than to use traditional methods to construct confidence intervals or to perform t-tests. The reason to teach resampling or permuation is because they reflect more easily generalizable ideas and, for many students, they are more accessible conceptually.

If we give students the appropriate tools and teach them the appropriate concepts about computation, it is very feasible to teach resampling and permutation at the introductory level. We can move into the statistics that we want to teach rather than the statistics that we can teach with only a calculator.

There is an "if" in the above paragraph. "If we give students the appropriate tools ...." When and where should this happen? Who should be responsible for it?

Currently no one at a typical college or university considers themselves responsible for teaching computation. To test this statement in operational terms, ask yourself this: Which course could a student at your university plausibly take to prepare himself or herself well for the Three Rs? Math departments, where students already are taking courses, don't talk about randomization and repetition. Computer science departments do, but few non-majors have contact with them.

Statisticians are uniquely positioned to fill the gap. First, statistics itself is required for a wide range fields, so statisticians have contact with large number of students. Second, statistical computation is needed in follow-up courses; providing transferable skills is valued. Third, much of the computation that people encounter involves collecting, storing, and accessing data, functions that are closely related to the statistical analysis of data.

By embracing computation, statisticians can do a better job of teaching statistics and can provide a valuable service to students and the college or university as a whole. Let's do it.

8. APPENDIX A

Formula 1 is my translation of an official US Department of Labor (1993) publication. Here is the original.

WORKSHEET 17-12a: STATISTICAL ANALYSIS
(Based on Selection Rates)

This Worksheet may be used to find the statistical significance of a difference in selection rates.

At the top of the sheet, enter the SCRR page 17 problem number and, beside "Issue," enter the names of the particular groups whose selection rates you are comparing, the type of selection decision and for what job(s); i.e., Female vs. Male Hires for Management Trainee. Then complete this page as follows:

(a) Step #1: Find the difference in selection rates by subtracting the lower rate from the higher.

(b) Step #2: For the value of one Standard Deviation (S.D.):

1. Multiply the total selection rate by 1 minus the total selection rate.
   NOTE: "1 minus the total selection rate" is just a shortcut for the total non-selection rate; i.e., if the total selection rate is .25, then 1 - .25 or .75 is the total non-selection rate.
2. Then find the value of 1 divided by the number of one group in the pool (of applicants, of candidates) and add it to the value of 1 divided by the number of the other group in the pool.

Multiply the result of (a) by the result of (b) and find the square root. This is one Standard Deviation.

(c) Step #3: For the number of Standard Deviations of the difference in selection rates shown in Step #1, divide the result of Step #1 by the result of Step #2.

Generally speaking, a result of 2 or more Standard Deviations is considered statistically significant.

9. APPENDIX B

The notation presented in Section 5 is built on the R (2006) statistics system and is very similar to the standard R notation.

R provides many types of objects: simple types (numbers, character strings, booleans), compound types (arrays, lists, data frames, factors), and more advanced types (formulas, etc.) This diversity is important for general programming, but to introductory-level students it can be daunting. It's helpful for such students to keep the set of operations small and to provide packaged operations which, while conveying a unified concept, avoid syntactical complexities.

I'll illustrate this by contrasting two operations native to R with the corresponding operations in the notation I propose.

R provides a flexible `sample` operator which can be used to select random elements of a set. It operates more-or-less intuitively on a simple array, but when given a data frame (a table with cases as rows and variables as columns) it does a very counter-intuitive thing: it samples the variables at random, not the cases! The `resample` operator in Section 5 samples the cases.

`Resample` is a trivial program based on `sample`, but it effectively hides the need to think about the internal structure of data frames.

The `do` operator in Section 5 parallels a built-in R operator called `replicate`. However, `replicate` returns a matrix rather than a data frame — that is, each replication generates one column of the result rather than one row. To a programmer, this is a trivial difference, easily fixed by applying the transpose operator to the result of `replicate`. But for introductory students, it's better to follow established patterns: each replication should be one more case, and cases are rows.

The `do` operator also differs from `replicate` in how it handles complicated types. For example, in the introductory course that I teach, students frequently bootstrap model coefficients. Linear models are fitted in R using the `lm` command, which, importantly, uses an excellent, clear notation for specifying the model. Here's an example using the job applicant data:

```
> lm( age ~ sex + job, data=apps)
Coefficients:
(Intercept)    sexM   joblaborer  jobservice  jobsupervisor
     30.937  -2.608        8.036       7.564          2.396
```

The `lm` program illustrates an important aspect of algorithms, that the output of one program can become the input to another. For example:

```
> summary(lm( age ~ sex + job, data=resample(apps)))
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)     31.331      3.264   9.600 1.84e-12 ***
sexM            -3.663      2.222  -1.648  0.10628
joblaborer       5.631      3.849   1.463  0.15047
jobservice       7.366      3.208   2.296  0.02637 *
jobsupervisor   18.669      6.254   2.985  0.00457 **

Residual standard error: 7.545 on 45 degrees of freedom
Multiple R-Squared: 0.2726,     Adjusted R-squared: 0.2079
F-statistic: 4.215 on 4 and 45 DF,  p-value: 0.005542
```

or

```
> anova(lm( age ~ sex + job, data=resample(apps)))
Analysis of Variance Table

Response: age
            Df Sum Sq Mean Sq  F value Pr(>F)
(Intercept)  1  59444   59444 672.3052 <2e-16 ***
sex          1     29      29   0.3301 0.5684
```

```
job          3    329     110   1.2385 0.3069
Residuals   45   3979      88
```

It's straightforward to fit a model to the case-wise resampled data, giving one realization of a bootstrap distribution of the coefficients:

```
> lm( age ~ sex + job, data=resample(apps))
Coefficients:
(Intercept)    sexM  joblaborer  jobservice  jobsupervisor
     35.674  -4.082       2.784       1.128         -9.007
```

The `do` operator has been arranged to recognize when the operation it is iterating produces a linear model object. It then does the right thing, pulling out the coefficients (as opposed to the object as a whole or the residuals, fitted values, etc.)

```
> do(3)(lm( age ~ sex + job, data=resample(apps)))
  X.Intercept.      sexM joblaborer jobservice jobsupervisor
1     29.10099 -4.269316  10.970199  10.533664     -1.600993
2     33.78310 -6.090139   8.274848   7.116478      9.550235
3     26.19910  1.845650  10.361776  10.323694      8.800904
```

10. REFERENCES

College Board (2006), "Statistics Course Description", http://apcentral.collegeboard.com/apc/public/repository/52269_apstatistics_lo_4328.pdf Technical report.

College Board (2007), "Sample Syllabi",http://apcentral.colegeboard.com/apc/public/repository/ap06_stat_syllabus1.pdf Technical Report.

Cobb, George W., (2007) "The Introductory Statistics Course: A Ptolemaic Curriculum?", *Technology Innovations in Statistics Education*,1, Article 1.

Duckworth, W. and Stephenson ,W., (2002) "Beyond Traditional Statistical Methods," *The American Statistician*, 56, 230-233.

Finzer, W. (2007), *Fathom Dynamic Data Software*, Key Curriculum Press, Emeryville, CA.

Hazelwood School District v United States, 433 U.S. 299 (1977).

JMP, Version 7. SAS Institute INc., Cary, NC, 1989-2007.

LabView Student Edition, National Instruments, 2007. http://www.ni.com/labviewse

Marriot, Michel (2007) "If Leonardo Had Made Toys", *New York Times*, February 8, 2007.

R Development Core Team (2006). "R: A Language and Environment for Statistical Computing", *R Foundation for Statistical Computing", Vienna, Austria.*

SPSS 15.0 Student Version for Windows, Prentice Hall, 2007.

Stephenson, W. Robert (1993) Minitab macro for bootstrapping, http://www.public.iastate.edu/~wrstephe/JSM2003/OneMeanCI.mac

United States Department of Labor (1993), "Employment Standards Administration", Onsite Review Procedures.