CS50 1997 Quantitative Study

The Harvard community has made this article openly available. Please share how this access benefits you. Your story matters

Citation

Citable link
http://nrs.harvard.edu/urn-3:HUL.InstRepos:25104524

Terms of Use
This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA
CS50 1997 Quantitative Study

Daniel J. Ellard

TR-12-98
April 27, 1998

Computer Science Group
Harvard University
Cambridge, Massachusetts
CS50 1997 Quantitative Study

Dan Ellard

April 27, 1998
Contents

1 Introduction .................................................. 1
  1.1 Organization ........................................... 1
  1.2 Acknowledgments ....................................... 1

2 Overview and Research Goals .................................. 2
  2.1 Related Work .......................................... 2
     2.1.1 Work Habits ..................................... 2
     2.1.2 Gender Differences ................................ 3
  2.2 Research Goals ......................................... 3
     2.2.1 Inherent Limits of Observational Studies ............ 4

3 Modeling Student Habits and Performance .................... 5
  3.1 The Relationship of Test Grades to Student Performance ....... 7
     3.1.1 Characteristics of Test Grades .................... 9
     3.1.2 Correlation Analysis ............................... 9
     3.1.3 The Effect of Unusual Values ....................... 9
     3.1.4 The Hazard of Interactions ........................ 10
     3.1.5 Substantive Discussion ................................ 10
  3.2 Time versus Test Grades ................................ 11
  3.3 Assignment Grades versus Test Grades ................... 11
  3.4 The Relationship Between Debugging Sessions and Time Spent on Assignments .... 12
     3.4.1 Discussion ........................................ 13
  3.5 The Relationship Between Rate, Time, and Assignment Grades .......... 13
  3.6 Does Starting Early Help? ................................ 16

4 The Demographics of CS50 Students ......................... 18
  4.1 Data Background ....................................... 18
     4.1.1 CS50 - 1996 ..................................... 18
  4.2 Relationships Between Gender, Year, and Prior Programming Experience .... 19
     4.2.1 Gender and Year .................................... 19
     4.2.2 Year and Prior Experience .......................... 19
     4.2.3 Gender and Prior Experience ...................... 20
     4.2.4 Combining Gender, Year, and Prior Experience ......... 22
  4.3 The Effects of Gender .................................. 23
     4.3.1 Intuitive Explanations .............................. 23
     4.3.2 Prior Experience and Enrollment Differs by Gender ........ 24
     4.3.3 Test Grades Differ by Gender ....................... 24
4.4 Quantitative Measurements of Differences in Work Habits ................................................................. 24
4.4.1 Do Females Spend More Time Programming in CS50? ................................................................. 26
4.4.2 Do Females Start Their Assignments Earlier? .................................................................................... 26
4.4.3 Do Females Spend More Time Attending Office Hours? ................................................................. 27
4.5 Gender Differences, Role Models, and Society ..................................................................................... 27
4.5.1 Discussion ......................................................................................................................................... 27

5 Conclusion ............................................................................................................................................. 30
  5.1 Summary ............................................................................................................................................... 30
  5.1.1 Accomplishments .......................................................................................................................... 30
  5.2 Directions for Future Research ........................................................................................................... 30

A Data Collection Methodology ............................................................................................................... 32
  A.1 Recruitment of Subjects ...................................................................................................................... 32
    A.1.1 Volunteer Bias .............................................................................................................................. 32
  A.2 The Data Collection System ............................................................................................................... 33
  A.3 The Anonymizer .................................................................................................................................. 34
    A.3.1 The Need for Anonymization ...................................................................................................... 34
    A.3.2 The CS50 Study Protocol ............................................................................................................ 37
    A.3.3 The Psychology of Anonymization ............................................................................................. 38
    A.3.4 Weaknesses in the Implementation ............................................................................................ 38

B Analysis of the Study Data ..................................................................................................................... 40
  B.1 Description of the Variables ............................................................................................................... 40
    B.1.1 General Criticisms of these Variables ......................................................................................... 40
  B.2 Summary Statistics and Discussion of Grades .................................................................................... 42
    B.2.1 Midterm, Exam, and Assignment Grades .................................................................................. 42
    B.2.2 Time Spent on Assignments ....................................................................................................... 42
    B.2.3 Discussion of Distributions ....................................................................................................... 43

C Assignment Descriptions ......................................................................................................................... 48
  C.1 Assignment 1 ....................................................................................................................................... 48
  C.2 Assignment 2 ....................................................................................................................................... 48
  C.3 Assignment 3 ....................................................................................................................................... 48
  C.4 Assignment 4 ....................................................................................................................................... 49
  C.5 Assignment 5 ....................................................................................................................................... 49
  C.6 Assignment 6 ....................................................................................................................................... 49
  C.7 Assignment 7 ....................................................................................................................................... 49
  C.8 Assignment 8 ....................................................................................................................................... 50
  C.9 Assignment 9 ....................................................................................................................................... 50
  C.10 The Final Project ............................................................................................................................... 51

D Lessons Learned ....................................................................................................................................... 52
Chapter 1

Introduction

1.1 Organization

This paper documents the quantitative study of the work habits, grades, and class composition of CS50 during the Fall semester of 1997. At the suggestion of Professor Margo Seltzer, this paper includes a much more detailed discussion of my research design and methodology than would ordinarily appear in a journal article. Most of this extra discussion is confined to the appendices, but parts of it appear in the main discussion.

Readers interested in my main results should read chapters, 2, 3, and 5.

Chapter 4 is a discussion of the role or effect of gender on enrollment and success in CS50 and CS51. Although not directly related to my original research questions, the data was too interesting and the issue too important to ignore.

Appendix A describes how the data was collected, including the anonymizer, and appendix C describes the assignments.

1.2 Acknowledgments

I would like to acknowledge the support of Professor Margo Seltzer and Professor Brad Chen, who supplied me with gender, year, and grade information for the 1997 CS50 class, as well as letting me sit in on TF meetings and lectures so I could keep track of the class for any unusual changes.

I would like to thank Dr. Brian Kernighan for providing 1996 CS50 data, as well as many very useful suggestions about how to conduct my research.

I would like to thank Professor Harry Lewis for providing guidance and helping me to find a way to study CS50 students that is both ethical and scientific.

Mark David Kernighan’s comments were very useful, and raised some interesting questions.

Finally, I would like to thank all the CS50 students who participated in the study. Without their voluntary (and unrewarded) participation, none of this research could have been possible.
Chapter 2

Overview and Research Goals

2.1 Related Work

2.1.1 Work Habits

This study was inspired by a recent paper written by Parrish, Lester, Cordes, and Moore based on a study they performed at the University of Alabama [PLCM97]. Their study found no convincing correlations between different “work styles” and in particular how early each student started on an assignment and how well they did on each of the assignments monitored by this study. I found their results very surprising, given my own experiences and intuitions, and decided that it would be interesting to perform a similar study here, using CS50 students.

There were several differences between the Alabama study and a study of CS50 that seemed worth investigating:

- There are important differences between CS50 students and the students who participated in the Alabama study. The Alabama students were intermediate computer science concentrators, who may already have established work patterns that are effective for them. In contrast, CS50 students come from a wide variety of backgrounds. Many CS50 students do have prior programming experience, but a large number have none.

  Perhaps different work patterns are employed by the CS50 non-CS concentrators, and perhaps these will have different effects on grades and time spent per assignment.

- The Alabama assignments appear to be easier and less time-consuming than the typical CS50 assignments, and perhaps the effects of work habits are reduced or eliminated when the assignments are less challenging.

- No information is given about the gender, year, or level of prior programming experience of the students in the Alabama study. Perhaps these factors have an effect on work habits.

  In addition, the number of students in the Alabama study was also relatively small– 43 students were enrolled in the Alabama study, and at least 34 of them participated in each assignment. Perhaps the sample is dominated by unusual students, and the results do not generalize.

- The only summary statistics given for the distributions of time spent per student per assignment are the mean and standard deviation. No information about the distribution of grades is given at all. Are
these distributions similar for CS50, or are there fundamental differences? Is my intuition about the relationship between grades and work habits an artifact of the grading methodology we use in CS50?

Another important question is whether the distributions of grades and time spent per assignment in the Alabama study meets the general criteria for correlation analysis. No mention is made of this in the Alabama study (and in fact, the significance level of the correlations is not even mentioned) and when I began to analyze the CS50 data it became apparent that it would be an error to consider only the raw distributions.¹

Parrish has continued his research in this area, studying how students fix the problems in their code [Par].

Most of the other research I am aware of in the area of programming habits and how students learn to program has focused on professional programmers [Wie71] or juvenile students [SS89]. My CS50 study is the largest study of its type which I am aware of.

### 2.1.2 Gender Differences

The topic of gender in computer science education has received considerable attention. Of particular concern is the difference between the percentage of women and men who concentrate in computer science, and their eventual graduation rates. Scrugg & Smith present current theories about the causes of this gap in [SS98].

Much of this research has been qualitative. My study provides an opportunity to measure some quantitative aspects of the possible differences between the habits and experiences of male and female students in CS50— or to show that there are no significant differences.

### 2.2 Research Goals

This research attempts to answer the following questions:

1. Do students who use the compiler frequently to check small changes for syntax errors get different scores on their assignments than students who make large numbers of changes in one edit?

   This question attempts to relate “level of intensity” (with respect to compilations) and student performance. This is addressed in section 3.5 (page 13).

2. What is the pattern of debugger use— at what point in the semester, if ever, do students really begin using the debugger? Do students who use the debugger spend less time on their coding? Is there a relationship between debugger use and how well the resulting programs work?

   These questions are addressed in section 3.4 (page 12).

3. Do factors such as year in school and gender predict performance in CS50? Are these predictors independent, or do they interact with each other?

   These questions are addressed in chapter 4, particularly sections 4.3 (page 23) and 4.4 (page 24).

¹It is possible for a correlation between two variables to be completely masked by an interaction effect with a third variable. It is also possible, though unusual, for a significant correlation to be masked by the presence of outliers. It is not stated whether the Alabama researchers checked for either of these cases.
2.2.1 Inherent Limits of Observational Studies

It must be noted that because of the inherent limits in purely observational studies, my study cannot prove that there exist causal relationships between variables, and it would be inappropriate to treat the resulting models as causal. For example, although the analysis in chapter 3 shows that there is a negative correlation between time spent on assignments and test scores, this should not be interpreted to mean that if a particular student scores higher on their tests, then they will spend less time on their assignments, nor does it predict that a student who spends less time on their assignments will get better grades on the tests! This model makes no predictions whatsoever about how a change in time would be reflected in a change in test grade.

Therefore, this study is incapable in providing hints as to how students might improve their grades, or decrease the amount of time they spend on the assignments. The best I can hope to do is to identify common traits of successful (or unsuccessful) students in an effort to generate causal hypotheses that can be tested via true experiments. This is disappointing, because what I’d really like to be able to do is to be able to give students concrete advice about how to improve their learning experience in CS50.
Chapter 3

Modeling Student Habits and Performance

The main goal of my study is to determine if there are relationships between different kinds of work habits and student performance, and whether these relationships varied among different subpopulations (i.e. gender and year). If such relationships exist, then my next goal is to construct models to attempt to quantify these relationships.

I investigated a large number of possible relationships between pairs of variables— and, in almost every case, I was unable to reject the null hypothesis that the variables were unrelated. My results support the findings of Parrish et al [PLCM97, Par] and extend these results to show that there are even fewer relationships than I expected.

Although correlations between the variables are generally low and insignificant\(^1\), there are three exceptions:

1. Grades on programming assignments are positively correlated with test grades— people who do better on the assignments tend to do better on the tests. This correlation is shown in figure 3.1, and discussed in section 3.1.

2. Time spent on programming assignments is negatively correlated with test grades— people who spend less time working on the assignments tend to do better on the tests. This correlation is displayed in figure 3.2, and discussed in 3.2.

3. Gender is significantly related to exam grades; males tend to do better on the exams than females. This difference is illustrated in figure 3.3 and discussed in more depth in section 4.3.

\(^1\text{Note: in this document, significance is always meant to represent that the null hypothesis can be rejected with a confidence level of 0.95 or higher. In addition, the following notation is used as a shorthand for other significance levels:}

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{Symbol} & \textbf{Confidence Level} \\
\hline
\texttt{~} & A confidence level between 0.90 and 0.95. \\
\texttt{*} & A confidence level between 0.95 and 0.99. \\
\texttt{**} & A confidence level between 0.99 or 0.999. \\
\texttt{***} & A confidence level of 0.999 or higher. \\
\hline
\end{tabular}
\end{center}
Table of correlations for the entire data set (omitting people who were not enrolled in the study for a particular assignment or who did not attempt a particular assignment). The “by gender” correlations have had the effect of the gender variable removed by partial correlation.

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Asst2 Score</th>
<th>Asst3 Score</th>
<th>Asst4 Score</th>
<th>Asst5 Score</th>
<th>Asst6 Score</th>
<th>Asst7 Score</th>
<th>Asst8 Score</th>
<th>Asst9 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exams</td>
<td>0.39</td>
<td>0.50</td>
<td>0.49</td>
<td>0.49</td>
<td>0.50</td>
<td>0.38</td>
<td>0.20</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>**</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>Exams (by gender)</td>
<td>0.38</td>
<td>0.51</td>
<td>0.49</td>
<td>0.50</td>
<td>0.51</td>
<td>0.40</td>
<td>0.17</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>~</td>
<td>*</td>
</tr>
<tr>
<td>N</td>
<td>69</td>
<td>117</td>
<td>124</td>
<td>120</td>
<td>121</td>
<td>120</td>
<td>101</td>
<td>115</td>
</tr>
</tbody>
</table>

Table of correlations after high leverage and outlier points have been removed.

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Asst2 Score</th>
<th>Asst3 Score</th>
<th>Asst4 Score</th>
<th>Asst5 Score</th>
<th>Asst6 Score</th>
<th>Asst7 Score</th>
<th>Asst8 Score</th>
<th>Asst9 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exams</td>
<td>0.26</td>
<td>0.38</td>
<td>0.46</td>
<td>0.44</td>
<td>0.46</td>
<td>0.31</td>
<td>0.43</td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>**</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>Exams (by gender)</td>
<td>0.25</td>
<td>0.42</td>
<td>0.49</td>
<td>0.47</td>
<td>0.45</td>
<td>0.36</td>
<td>0.41</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>**</td>
<td>**</td>
<td>*</td>
</tr>
<tr>
<td>N</td>
<td>64</td>
<td>83</td>
<td>118</td>
<td>102</td>
<td>114</td>
<td>100</td>
<td>58</td>
<td>102</td>
</tr>
</tbody>
</table>
Figure 3.2: Correlations between time spent on assignments and total examination grades.
Table of correlations for the entire data set (omitting people who were not enrolled in the study for a particular assignment or who did not attempt a particular assignment).

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Asst2 Time</th>
<th>Asst3 Time</th>
<th>Asst4 Time</th>
<th>Asst5 Time</th>
<th>Asst6 Time</th>
<th>Asst7 Time</th>
<th>Asst8 Time</th>
<th>Asst9 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exams</td>
<td>-0.45</td>
<td>-0.28</td>
<td>-0.13</td>
<td>0.05</td>
<td>-0.33</td>
<td>-0.27</td>
<td>0.09</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>***</td>
<td>**</td>
<td>*</td>
<td></td>
<td>***</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exams (by gender)</td>
<td>-0.42</td>
<td>-0.26</td>
<td>-0.12</td>
<td>-0.05</td>
<td>-0.32</td>
<td>-0.26</td>
<td>0.06</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>***</td>
<td>**</td>
<td>*</td>
<td></td>
<td>***</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>69</td>
<td>117</td>
<td>124</td>
<td>120</td>
<td>121</td>
<td>120</td>
<td>101</td>
<td>115</td>
</tr>
</tbody>
</table>

Table of correlations after high leverage and outlier points have been removed.

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Asst2 Time</th>
<th>Asst3 Time</th>
<th>Asst4 Time</th>
<th>Asst5 Time</th>
<th>Asst6 Time</th>
<th>Asst7 Time</th>
<th>Asst8 Time</th>
<th>Asst9 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exams</td>
<td>-0.24</td>
<td>-0.34</td>
<td>-0.23</td>
<td>-0.17</td>
<td>-0.35</td>
<td>-0.18</td>
<td>-0.21</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>~</td>
<td>**</td>
<td>*</td>
<td>~</td>
<td>***</td>
<td>~</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Exams (by gender)</td>
<td>-0.24</td>
<td>-0.33</td>
<td>-0.22</td>
<td>-0.17</td>
<td>-0.35</td>
<td>-0.19</td>
<td>-0.23</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>~</td>
<td>**</td>
<td>*</td>
<td>~</td>
<td>***</td>
<td>~</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>64</td>
<td>83</td>
<td>118</td>
<td>102</td>
<td>114</td>
<td>100</td>
<td>58</td>
<td>102</td>
</tr>
</tbody>
</table>

3.1 The Relationship of Test Grades to Student Performance

I consider student performance on an assignment to be a combination of how high a score a student got on the assignment, and how quickly a student was able to complete the assignment. The goal of a student is to achieve the highest possible score, while spending the smallest possible amount of time working on the assignment. I believe that most students can achieve very good grades on the assignments, given enough time— but with so many other demands on their time, this would be only a Pyrrhic victory for the student.

My initial suspicion would be that assignment grades would be positively correlated with longer time spent working on the assignment. This, however, is not the case.

My current theory is that CS50 students find ways to succeed— there is no fixed recipe for how they accomplish their success, and they do not necessarily find the most efficient or effective way to do their homework. Instead, they work on their homework in whatever ways seem best to them, and when they feel that they have done an adequate job, they stop working and hand their homework in. In most cases, the students have a relatively good idea of how well they’ve done, and therefore they do not stop working until their homework is good enough to earn a reasonably good grade. In my experiences as teaching fellow, few students handed in homework that was badly broken without realizing it— and many students chose to not hand in their homework at all if they believed that it was largely incorrect. Among students who did well in the course, there were many who expressed genuine surprise (or indignation) when their assignments that

---

2Nor do I suspect that they spend very much time thinking about what the most efficient or effective way might be, or investing time to learn tools that might help them complete the assignments more quickly and easily. This hypothesis is discussed in more detail in section 5.2 (page 30). Mark Kernighan has suggested that students who spend more time on earlier assignments may be investing time to learn tools or techniques that will help them later; it would be interesting to see whether there are negative correlations between time spent on early assignments and later assignments.
Figure 3.3: Summary statistics for test grades of 1996 and 1997 CS50 students, divided by gender.

<table>
<thead>
<tr>
<th>Test</th>
<th>Gender</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midterm 1</td>
<td>Female</td>
<td>105</td>
<td>35.54</td>
<td>9.10</td>
<td>14</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>263</td>
<td>37.83</td>
<td>8.60</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>Midterm 2</td>
<td>Female</td>
<td>105</td>
<td>26.67</td>
<td>9.02</td>
<td>10</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>262</td>
<td>29.12</td>
<td>9.57</td>
<td>11</td>
<td>50</td>
</tr>
<tr>
<td>Final Exam</td>
<td>Female</td>
<td>106</td>
<td>121.69</td>
<td>19.69</td>
<td>64</td>
<td>155</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>263</td>
<td>126.40</td>
<td>20.76</td>
<td>55</td>
<td>172</td>
</tr>
</tbody>
</table>

1997 CS50 Test Grades by Gender

<table>
<thead>
<tr>
<th>Test</th>
<th>Gender</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midterm 1</td>
<td>Female</td>
<td>65</td>
<td>32.58</td>
<td>9.34</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>166</td>
<td>35.97</td>
<td>8.87</td>
<td>13</td>
<td>50</td>
</tr>
<tr>
<td>Midterm 2</td>
<td>Female</td>
<td>65</td>
<td>32.20</td>
<td>4.86</td>
<td>21</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>167</td>
<td>34.22</td>
<td>6.27</td>
<td>17</td>
<td>46</td>
</tr>
<tr>
<td>Final Exam</td>
<td>Female</td>
<td>65</td>
<td>128.85</td>
<td>19.59</td>
<td>43</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>166</td>
<td>137.44</td>
<td>19.37</td>
<td>74</td>
<td>176</td>
</tr>
</tbody>
</table>

Midterm grades lower than 10 and final exam grades of zero are omitted from these statistics.

All of the differences between male and female mean test scores are significant at the 0.05 level. All of the 1997 differences are significant at the 0.02 level, and the difference for the 1996 midterm 1 and 1997 final exam grades are significant at the 0.01 level. (Computed via a t-test with unequal variances.)
they believed to have been correct had lost points, which suggests that they believed that they had done enough work to get full credit and might have done more work had they known how we were going to grade their work.

If this theory is correct, and most CS50 students managed to achieve approximately the same grades on their homeworks, my methodology is extremely hard to apply. Instead of relating particular work habits with particular grades, I must instead consider other factors, such as the time required to complete the assignments, as measurements of how effective a particular set of work habits is. I have attempted to do that, to a small extent, as part of my study, and it has shown no significant correlation between any predictors and the amount of time that is required to complete the assignments, with the exception of mid-term and final examination grades. There is a strong negative correlation between test grades and the amount of time required to complete most of the assignments. I believe that this corresponds to the hypothesis that students are doing as much work as they feel is necessary – and students who do well on the tests are either faster at completing assignments, because they are more familiar with or have a better grasp of the material, or perhaps because they have done well on the exams they feel less pressure to do well on the homeworks and therefore spend less time working on them.3

The relationship between test grades (midterm and final exam scores) and student performance is significant in two ways—there is a positive correlation between test grades and assignment grades, and a negative correlation between test grades and time spent working on the assignments. This section explores these relationships.

3.1.1 Characteristics of Test Grades

The grades for each of the three tests (hourly 1, hourly 2, and the final exam) show a normal tendency.

To assess student performance on the tests, I created a composite variable called EXAMS which is simply the sum of the scores from each student’s tests. Since the tests are scored in such a way that the total number of points available is proportional to the amount of time spent on the test, the three tests share a common scale and their sum is easy to interpret. It would also be conceivable to create a composite based on the weight each of these grades has in computing the final grade of each student; the result is very similar.

3.1.2 Correlation Analysis

There are interactions between these correlations which, when considered together, complicate them considerably. For example, males tend to do better on the tests, and students who do well on the tests tend to do better on the assignments. Based on these correlations, one might assume that males would do better on the assignments—but this is actually not the case.4 The actual changes to the correlations between test scores and assignment scores, with the effects of gender partialled out, are shown in figure 3.1.

3.1.3 The Effect of Unusual Values

One hazard of correlation analysis is the potential impact of unusual values on the correlation coefficient. A small number of high-leverage points can increase the correlation coefficient until it becomes significant—when this happens, a strong relationship between a small number of points can have the same effect on the

---

3 As Brian Kernighan has pointed out, this effect cannot apply to the final exam, since it is the last opportunity to do well in the course, and so students who have not done well on the assignments may tend to cram for it. In this case, students who did less work on the assignments may appear to do better on the exams because of a last-ditch effort.

4 A T-test of the assignment scores shows no significant difference by gender, either for each of the assignments or the sum of the assignments over the semester. Females do score somewhat higher, on average, on assignment 2, 3, 4, 5, and 7, while males do better on 6, 8, and 9.
correlation coefficient as a more general relationship. In the other extreme, a set of outliers may reduce, eliminate, or even reverse a correlation.

One method to detect potentially influential points is to examine plots of the bivariate distribution of two variables before testing whether the two variables are correlated. I performed this analysis, and discovered a number of potentially problematic points. In particular, for each assignment there are always at least a handful of students who do very poorly. Since the majority of students do very well on most of the assignments, the grades of students who do poorly can potentially exert a great deal of leverage on the correlation (or regression) of grades to other variables. Similarly, there are a small number of students who appear to spend extraordinary amounts of time working on the assignments—and these students can potentially have a profound effect on the apparent relationship between time spent working on the assignments and another variable.

After examining the bivariate distributions, I decided that the appropriate course of action would be to omit some students from the analysis entirely (because they missed a midterm or the final) and to omit statistics for specific assignments for students who did particularly badly. Figure 3.4 lists the criteria for omitting students from the analysis, and figures 3.2 and 3.1 include the resulting correlations.

Note that many of the correlations in 3.2 and 3.1 are weakened by the removal of these data points (although others, such as the correlations for homework 8, become stronger).

### 3.1.4 The Hazard of Interactions

Another hazard is the existence of interaction effects. A simple two-way interaction effect occurs when the relationship between two variables depends on the value of a third variable. More complex interactions are also possible.

In a study of this size, there are innumerable potential interaction effects. I have not tested for them all, but instead focused primarily on possible interactions between gender and the relationships between other variables. I discovered no conclusively significant interactions.\(^5\)

### 3.1.5 Substantive Discussion

Although the three kinds of correlations identified in the previous section are significant (i.e., their presence is unlikely to be due to a random fluke in the sample), the question remains whether they have any substantive importance.

\(^5\)I did discover some significant interactions—but any time you conduct hundreds of significance tests at the 0.05 level, it is expected that 5% of these significance tests will pass. Since there was no discernable pattern to the interactions, I do not believe that any belong in my models.
Figure 3.5: Models for the relationship between time spent on assignments and test grades.

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Asst 2 Time</th>
<th>Asst 3 Time</th>
<th>Asst 4 Time</th>
<th>Asst 5 Time</th>
<th>Asst 6 Time</th>
<th>Asst 7 Time</th>
<th>Asst 8 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>4.69***</td>
<td>12.32***</td>
<td>11.83***</td>
<td>18.70***</td>
<td>15.11***</td>
<td>6.50***</td>
<td>45.3***</td>
</tr>
<tr>
<td>Exams</td>
<td>-0.013</td>
<td>-0.029</td>
<td>-0.025</td>
<td>-0.032</td>
<td>-0.040</td>
<td>-0.012</td>
<td>-0.10</td>
</tr>
</tbody>
</table>

Note that assignment 9 does not fit this model – exam grades are not a significant predictor of assignment 9 time. Gender is, however: male students, on average, spent 1.08 hours longer on assignment 9.

For example, the strongest of these correlations (for the reduced sample) is between a student’s test scores and their score on assignment 4. This relationship has a correlation coefficient of 0.49. This means that the fraction of the variance of test scores “accounted for” by this correlation is $(0.49)^2 = 0.24$. This means that 76% of the variance must be accounted for by other factors.

To investigate the nature of this relationship further, it is necessary to build models.

### 3.2 Time versus Test Grades

Models for the relationship between time spent on assignments and test grades are shown in figure 3.5.

For each of the assignments except assignment 9, test grades are a significant predictor of how much time a student spends on each assignment. The parameters of the model can be interpreted as how difficult, in some sense, it is to get points on each assignment relative to the other assignments and the tests. For example, in assignment 6, the test grade parameter is -0.10, which means that a one point change in points scored on the combination of the midterms and the final is associated with an expected difference of the number of hours spent on assignment 6 of -0.04: if one student scores 10 more points than another on his or her tests, then we would expect them to also spend 0.4 fewer hours working on assignment 8.

Similarly, since the test score parameter for assignment 8 is -0.10, then we would expect that a student who scores 10 more points than another on his or her tests would spend one less hour working on assignment 8. The expected “benefit” of doing well on the tests, therefore, is 2.5 times larger for assignment 8 than assignment 6.

The largest ratio between the test score parameters is between assignment 7 and 8—the incremental change in the expected number of hours is eight times larger for assignment 8 than for assignment 7. The factors that determine how well a student does on the exams have a much greater influence on the number of hours a student spends on assignment 8 than 7. Perhaps this indicates that assignment 8 is a better measure of a student’s ability, or perhaps that assignment 7 is simply very easy and doesn’t take anyone much time to program.

### 3.3 Assignment Grades versus Test Grades

Models for the relationship between assignment grades and test grades are shown in figure 3.6.
Figure 3.6: Models for the relationship between assignment scores and test grades.

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Asst2 Score</th>
<th>Asst3 Score</th>
<th>Asst4 Score</th>
<th>Asst5 Score</th>
<th>Asst6 Score</th>
<th>Asst7 Score</th>
<th>Asst8 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>37.4</td>
<td>32.6</td>
<td>33.0</td>
<td>28.6</td>
<td>31.7</td>
<td>42.8</td>
<td>17.8</td>
</tr>
<tr>
<td>Exams</td>
<td>0.06</td>
<td>0.09</td>
<td>0.09</td>
<td>0.10</td>
<td>0.09</td>
<td>0.06</td>
<td>0.16</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.07</td>
<td>0.15</td>
<td>0.21</td>
<td>0.19</td>
<td>0.21</td>
<td>0.10</td>
<td>0.19</td>
</tr>
<tr>
<td>p-value</td>
<td>0.04</td>
<td>0.0004</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.08</td>
<td>0.0007</td>
</tr>
</tbody>
</table>

Note that assignment 9 does not fit this model—exam grades are not a significant predictor of assignment 9 score.

Figure 3.7: Correlations between time spent on each assignment and number of debugging sessions during that assignment.

Table of correlations for the entire data set (omitting people who were not enrolled in the study for a particular assignment or who did not attempt a particular assignment). High leverage and outlier points have been removed. People who never use the debugger at all for a particular assignment are omitted from the correlation calculation.

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Asst3 Time</th>
<th>Asst4 Time</th>
<th>Asst5 Time</th>
<th>Asst6 Time</th>
<th>Asst7 Time</th>
<th>Asst8 Time</th>
<th>Asst9 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugger Sessions</td>
<td>0.37 ***</td>
<td>0.35 ***</td>
<td>0.29 **</td>
<td>0.56 ***</td>
<td>0.56 ***</td>
<td>0.75 ***</td>
<td>0.59 ***</td>
</tr>
<tr>
<td>N</td>
<td>84</td>
<td>105</td>
<td>84</td>
<td>111</td>
<td>62</td>
<td>58</td>
<td>91</td>
</tr>
</tbody>
</table>

It is clear that there is a significant and positive relationship between test grades and assignment grades, but this relationship has little substantive meaning. The strongest relationships (for assignments four and six) have an $R^2$ of only 0.21, which means that this model only explains 21% of the variability in assignment grades—the rest of the variability must be explained by other factors.

3.4 The Relationship Between Debugging Sessions and Time Spent on Assignments

One of the original research questions was to explore the relationship between use of the debugger and time spent on the assignments. The hypothesis was that people who use the debugger more often spend less time on the assignments. In fact, the opposite appears to be the case—there is a strong positive correlation between the number of debugging sessions and the number of hours spent on the programming assignments. Figure 3.7 illustrates the correlation.

The relationship between number of gdb sessions and the time spent on each assignment appears to be
nonlinear. Tukey's ladder suggests that the logarithm of the number of debugging sessions per assignment is a better match, and some experimentation showed that this works well. Scatterplots of time versus number of debugging sessions and the log of the number of debugging sessions are shown in Figure 3.8.

The resulting non-linear models are shown in Figure 3.9.

Note that test grades are still a significant predictor of time spent on each assignment, even after controlling for the number of debugging sessions, for assignments 3 through 6, but are not significant for assignments 7, 8, or 9. Also note that the parameter estimates for the test grades are similar for this model and the model based only on test grades in the previous section.

3.4.1 Discussion

The interpretation of these models is interesting, and not without controversy. The non-linear models suggest that as the number of debug sessions increases, the corresponding length of time required to complete the assignments increases exponentially. The linear models, on the other hand, suggest that the change in the length of time required to complete the assignments is proportional to the change in the number of debug sessions. The reason why these seemingly contradictory models can both fit the data is that the values of the data, and the range of the values in the data, are relatively small. Their interpretation, therefore, is not very different over the set of ranges of CS50 assignments.

For assignments 3, 4, 5, and 7, the logarithm of the number of debugs per hour falls within a reasonably tight range. For each increase of between 1.2 and 1.5 in the natural log of the number of debugs, the expected number of hours increases by one. For assignments 5 and 9 the number is somewhat higher, however (and for assignment 8, the factor is enormously higher and perhaps pointless to interpret, due to the strange nature of assignment 8).

3.5 The Relationship Between Rate, Time, and Assignment Grades

One of the original research questions was whether there is a relationship between the frequency at which a student uses the tools (i.e. runs code through the compiler to check for syntax errors, instead of trying to compile only perfect code) is related to the score a student receives on an assignment, or how much time is spent. The data from this study suggests that there is no simple relationship between rate of use and either grade or total time spent per assignment.

The data suggests a weak positive correlation between rate and time and between rate and grade for assignments 8 and 9, but this correlation is undermined by the existence of several extreme values. In addition, any values derived from the time variables need to be treated with some skepticism, due to the crude way in which time spent per assignment is estimated.

There is also no generally significant relationship between gender and rate. The only assignment that the difference is significant for is assignment 8. Similarly, there is no clear relationship between student rate of work and whether the work is done in the terminal room or from their dorm rooms.

---

6A linear regression gives a good fit across the observed values but the residuals are quite heteroskedastic. Depending on your analysis philosophy, it can be acceptable to fit a linear model for the relationship between time and debug sessions (and a linear model is certainly easier to interpret), but I believe that the non-linear model I've chosen is a better fit. The linear models are illustrated, for the sake of comparison, in figure 3.9. Again, note that the test grades remain significant for assignments 3 through 6, but are not significant for assignments 7, 8, or 9, and note that the parameter estimates for the test grades are very similar to those for the previous models, as well as the model based only on test grades (in the previous section).

7In addition, there are artifacts of the session clustering algorithms— for example, a scatterplot of the number of jobs versus the number of hours spent per assignment will have a frontier along the 45° line. This is because of the way that time is measured, which artificially prevents any rates of less than one task per hour from being reported, at least using the current parameters (described in section B.1). The plot, which looks very intriguing to the naked eye, can lead to completely erroneous conclusions.
Figure 3.8: Plots of number of time versus debugging sessions and the log of the number of debugging sessions, for assignments 5 and 6.
Figure 3.9: Models for time spent on assignments as a function of debugging sessions.

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Asst3 Time</th>
<th>Asst4 Time</th>
<th>Asst5 Time</th>
<th>Asst6 Time</th>
<th>Asst7 Time</th>
<th>Asst8 Time</th>
<th>Asst9 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>8.14</td>
<td>11.85</td>
<td>19.00</td>
<td>8.77</td>
<td>4.27</td>
<td>-10.8</td>
<td>1.27</td>
</tr>
<tr>
<td>Log(DBG)</td>
<td>1.23</td>
<td>1.23</td>
<td>1.30</td>
<td>1.58</td>
<td>1.16</td>
<td>11.73</td>
<td>1.71</td>
</tr>
<tr>
<td>Exams</td>
<td>-0.026</td>
<td>-0.034</td>
<td>-0.043</td>
<td>-0.026</td>
<td>***</td>
<td>**</td>
<td>***</td>
</tr>
<tr>
<td>(R^2)</td>
<td>0.25</td>
<td>0.22</td>
<td>0.14</td>
<td>0.36</td>
<td>0.32</td>
<td>0.67</td>
<td>0.35</td>
</tr>
<tr>
<td>p-value</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.003</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Note that in the non-linear models, the \(R^2\) statistic measures the amount of variance in the time explained by the variance in Log(DBG), not the variance in DBG.

Figure 3.10: Linear models for time spent on assignments as a function of debugging sessions.

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Asst3 Time</th>
<th>Asst4 Time</th>
<th>Asst5 Time</th>
<th>Asst6 Time</th>
<th>Asst7 Time</th>
<th>Asst8 Time</th>
<th>Asst9 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>9.91</td>
<td>12.57</td>
<td>18.78</td>
<td>10.0</td>
<td>5.40</td>
<td>13.29</td>
<td>3.86</td>
</tr>
<tr>
<td>Log(DBG)</td>
<td>0.08</td>
<td>0.14</td>
<td>0.14</td>
<td>0.16</td>
<td>0.19</td>
<td>0.19</td>
<td>0.15</td>
</tr>
<tr>
<td>Exams</td>
<td>-0.026</td>
<td>-0.033</td>
<td>-0.038</td>
<td>-0.025</td>
<td>***</td>
<td>~</td>
<td>*</td>
</tr>
<tr>
<td>(R^2)</td>
<td>0.31</td>
<td>0.23</td>
<td>0.17</td>
<td>0.36</td>
<td>0.44</td>
<td>0.66</td>
<td>0.32</td>
</tr>
<tr>
<td>p-value</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0006</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

For this model, one high-influence point was omitted—the student who spent a lot of time on assignment 8, but rarely used the debugger.
3.6 Does Starting Early Help?

Conventional wisdom suggests that students who start the assignments early and spend an hour or two per day on the assignments (interspersed with attending office hours, section, and lecture) do better on the assignments than students who start only a day or two before the due date, and do a marathon session.

Although I have not completed an analysis of this theory, the data suggests that many (if not the overwhelming majority) of the students do not do not begin their assignments early.

Figure 3.11 (page 17) illustrates the cumulative number of tasks performed over time for each assignment. Each assignment’s trace begins with a handful of jobs by a few students, and does not become steep until a few days before the assignment is due. Since different students have their assignments due on different days, I suspect that the curve is even steeper than it appears— if all of the times were adjusted on a per-student basis to make it appear that their assignments were all handed in simultaneously. I would not be surprised to find that the most common strategy is to not begin programming in earnest until the night before the assignment is due. Even for assignments such as assignment 8, which students were repeatedly urged to begin early, I believe that most students did the majority of their programming during the last few days.

Of course, there is more to the assignments than just the programming, and it may well be that students are starting the other parts of the assignment and sketching out the design of their assignments for days before starting programming.\footnote{Although there are skeptics who believe that this is very unlikely, I do know of students who wrote out their code on paper before even attempting to use the computer. I was very surprised to discover that people still do this.} Unfortunately, I do not have any way to measure this activity. Additional discussion of this issue is given in section 4.4.2 (page 26).
Figure 3.11: Cumulative Task Count For All Assignments.

Each curve corresponds to the cumulative number of jobs (make, gdb, aa, ad, or the ANT test scripts) run by students working on each assignment over time. The curves represent, from left to right, assignments 3 through 9. (Assignment 2 was omitted because the number of students enrolled in the study during assignment 2 was small.)

Note that the curves represent simple counts, and are not normalized by the number of students enrolled in the study at each time, or the number of students who turned in each assignment. Given the large number of people who did nothing for assignment 8, this makes the assignment 8 curve even more extraordinary. The x axis is measured in days.
Chapter 4

The Demographics of CS50 Students

In the previous chapter, I discussed relationships between the work habits and elements of a student’s performance. In this chapter, I focus on demographic predictors of student performance and work habits.

To develop basic background on the population of CS50 students, I have started to study the relationships between variables that I believe may be related to student work habits.

Currently I am considering the following variables: year (freshman, sophomore, junior, and senior), gender, and self-reported prior programming experience (none, little, some, or lots).

The fact that these variables are important is demonstrated in Section 4.3.3, which explores the alarming tendency of female students to score significantly lower on CS50 exams than their male counterparts.

4.1 Data Background

4.1.1 CS50 - 1996

For 1996, I have a database of the year, gender, whether the student took the class pass/fail, and some information about their performance in the course.\textsuperscript{1} I do not have any information about the identities of any of these students, however, so I cannot cross-correlate this information with the survey forms for that year in order to enter their self-reported level of prior programming experience.

The small number of graduate students and special students enrolled in CS50 in 1996 were omitted from this database, since these students are not “typical” CS50 students and there are not enough of them to draw any significant statistical inferences. A small number of students who did not complete their course work were also omitted from this database.

For 1997, I have a database of the year, gender, and self-reported amount of prior programming experience for each student who registered for the course by the official registration date at the beginning of the semester.\textsuperscript{2}

\textsuperscript{1}This information was provided by Dr. Brian Kernighan, who taught CS50 in 1996.

\textsuperscript{2}This database was constructed from information supplied by Professor Margo Seltzer, and from an on-line survey form that most students voluntarily filled out at the start of the semester.
Figure 4.1: Distribution of 1996 CS50 students by year and gender.

<table>
<thead>
<tr>
<th></th>
<th>1-fresh</th>
<th>2-soph</th>
<th>3-junior</th>
<th>4-senior</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>22</td>
<td>23</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11.11</td>
<td>5.96</td>
<td>6.23</td>
<td>5.42</td>
<td>28.73</td>
</tr>
<tr>
<td></td>
<td>38.68</td>
<td>20.75</td>
<td>21.70</td>
<td>18.87</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30.60</td>
<td>23.40</td>
<td>28.75</td>
<td>32.79</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>263</td>
</tr>
<tr>
<td></td>
<td>25.20</td>
<td>19.51</td>
<td>15.45</td>
<td>11.11</td>
<td>71.27</td>
</tr>
<tr>
<td></td>
<td>35.36</td>
<td>27.38</td>
<td>21.67</td>
<td>15.59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>69.40</td>
<td>76.60</td>
<td>71.25</td>
<td>67.21</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>134</td>
<td>94</td>
<td>80</td>
<td>61</td>
<td>369</td>
</tr>
<tr>
<td></td>
<td>36.31</td>
<td>25.47</td>
<td>21.68</td>
<td>16.53</td>
<td>100.00</td>
</tr>
</tbody>
</table>

### 4.2 Relationships Between Gender, Year, and Prior Programming Experience

#### 4.2.1 Gender and Year

Figures 4.1 and 4.2 illustrate the distribution of CS50 students by class and gender for the 1996 and 1997 semesters. The 1996 data is based on the final grades, and omits a small number of students who did not complete their coursework. The 1997 data is based on the preliminary registration at the beginning of the semester, and declined somewhat during the semester as students dropped the course.

Both tables show that there is a general downward trend in enrollment for older classes—freshmen constitute the largest percentage of the class, and seniors the smallest, while sophomores and juniors are approximately equally numerous.

This trend appears to be stronger among males than females (in fact, the number of junior females is actually more than twice that of sophomore females in 1997), but this apparent relationship is not statistically significant.

#### 4.2.2 Year and Prior Experience

Figure 4.3 illustrates the distribution of 1997 CS50 students by class and prior experience.

The p-value of the chi-square statistic for this table is highly significant, so based on this data we can infer that there is a very significant relationship between the class of a CS50 student and their prior programming experience— in general, the younger students are more likely to self-report that they have “some” or “lots” of prior programming experience. More than one third of the freshmen reported having “some” or “lots” of
 prior experience, while less than 7% of seniors report “some” experience and none whatsoever report that they have “lots” of prior experience.

This analysis is open to criticism, however. It may be that younger students (particularly freshmen) are more likely to attribute to themselves higher levels of prior programming experience, because in their experiences they have not yet encountered other people who have done substantial amounts of programming. Upperclassmen who have watched their comrades struggle through CS50 and CS51 may have reassessed their programming experience in this light. To strengthen this analysis, it would be useful to quantify the amount of prior programming experience in a more objective manner—for example, asking people if they have taken the AP exam, any programming course at all, or written programs that employ certain constructs (i.e. “have you ever written a program that used loops? Nested loops?”).

To address this issue, I plan to revise the survey and certainly rethink the survey for any repetitions of this study.

### 4.2.3 Gender and Prior Experience

Figure 4.4 illustrates the distribution of students by self-reported prior programming experience and gender.

Note that the level of prior experience was self-reported by the students, and I did not get 100% response; I do not have this data for approximately 16% of the students. Therefore, any inferences drawn based on the prior experience data must be interpreted with caution.

There appears to be a relationship between gender and prior programming experience, but this is a statistically significant interaction between gender and self-reported prior programming experience. Male students are more likely to claim to have “some” or “lots” of programming experience than female students. This gap, and its implications, are discussed in more detail in section 4.5 (page 27).
Figure 4.3: Distribution of 1997 CS50 students by year and prior experience.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percent</th>
<th>Row Pct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Col Pct</td>
<td>0–None</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>1-Fresh</td>
<td>15</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>6.28</td>
<td>11.72</td>
</tr>
<tr>
<td></td>
<td>16.48</td>
<td>30.77</td>
</tr>
<tr>
<td></td>
<td>29.41</td>
<td>33.33</td>
</tr>
<tr>
<td>2-Soph</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>4.18</td>
<td>6.28</td>
</tr>
<tr>
<td></td>
<td>19.23</td>
<td>28.85</td>
</tr>
<tr>
<td></td>
<td>19.61</td>
<td>17.86</td>
</tr>
<tr>
<td>3-Junior</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>7.95</td>
<td>8.79</td>
</tr>
<tr>
<td></td>
<td>32.76</td>
<td>36.21</td>
</tr>
<tr>
<td></td>
<td>37.25</td>
<td>26.00</td>
</tr>
<tr>
<td>4-Senior</td>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>2.93</td>
<td>8.37</td>
</tr>
<tr>
<td></td>
<td>18.42</td>
<td>52.63</td>
</tr>
<tr>
<td></td>
<td>13.73</td>
<td>23.81</td>
</tr>
<tr>
<td>Total</td>
<td>51</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>21.34</td>
<td>35.15</td>
</tr>
</tbody>
</table>

Note: the Unknown students did not report their level of prior programming experience.
Figure 4.4: Distribution of 1997 CS50 students by gender and prior experience.

<table>
<thead>
<tr>
<th>Gender</th>
<th>0-No Exp</th>
<th>1-Little</th>
<th>2-Some</th>
<th>3-Lots</th>
<th>5-Unkwn</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>20</td>
<td>28</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>8.37</td>
<td>11.72</td>
<td>3.77</td>
<td>1.26</td>
<td>1.67</td>
<td>26.78</td>
</tr>
<tr>
<td></td>
<td>31.25</td>
<td>43.75</td>
<td>14.06</td>
<td>4.69</td>
<td>6.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>39.22</td>
<td>33.33</td>
<td>18.00</td>
<td>18.75</td>
<td>10.53</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>31</td>
<td>56</td>
<td>41</td>
<td>13</td>
<td>34</td>
<td>175</td>
</tr>
<tr>
<td></td>
<td>12.97</td>
<td>23.43</td>
<td>17.15</td>
<td>5.44</td>
<td>14.23</td>
<td>73.22</td>
</tr>
<tr>
<td></td>
<td>17.71</td>
<td>32.00</td>
<td>23.43</td>
<td>7.43</td>
<td>19.43</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60.78</td>
<td>66.67</td>
<td>82.00</td>
<td>81.25</td>
<td>89.47</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>51</td>
<td>84</td>
<td>50</td>
<td>16</td>
<td>38</td>
<td>239</td>
</tr>
<tr>
<td></td>
<td>21.34</td>
<td>35.15</td>
<td>20.92</td>
<td>6.69</td>
<td>15.90</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Note that 15.9% of the class (12 freshmen, 11 sophomores, 7 juniors, and 8 seniors) did not report their level of prior programming experience. (Also note that female students appear to be more likely to fill out the survey.)

4.2.4 Combining Gender, Year, and Prior Experience

Looking at all three of these variables together for the 1997 data reveals an interesting fact: there is a strong relationship between year and prior programming experience for male students. The younger male students are much more likely to have a higher self-reported level of prior programming experience.

This trend appears to be repeated for females (for example, all of the females who reported that they have “lots” of prior experience are freshmen, as are more than half of the females who reported that they had “some”. However, the simple lack of females with “some” or “lots” of prior programming experience (12 out of 76) makes it very hard to show that there is a relationship at all.

In summary, the relationship between scores and year appears to be interesting and worth further analysis.

The relationship between self-reported prior experience, gender, and score on the first midterm is also interesting, and is illustrated in figure 4.5. The frequency counts of many of the cells in this table are small, so the results should be treated with some skepticism, but there does appear to be a positive correlation between midterm grades and prior experience—students with higher self-reported levels of prior programming experience tend to do better on the first midterm. Among students with no reported prior experience, gender does not play a significant role, but in every other level of prior experience, male students score higher than females (although these differences are not statistically significant, because of the small number of female students with higher levels of prior programming experience). This could imply that the effect of prior programming experience is different for men and women, or it could simply mean that men and women self-report their level of prior programming experience on different scales.
Figure 4.5: 1997 hourly 1 grades, separated by gender and self-reported level of prior programming experience.

<table>
<thead>
<tr>
<th>Experience</th>
<th>Gender</th>
<th>N</th>
<th>Mean</th>
<th>StdDev</th>
<th>Max</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Female</td>
<td>20</td>
<td>29.6</td>
<td>9.4</td>
<td>17</td>
<td>44</td>
</tr>
<tr>
<td>Male</td>
<td></td>
<td>31</td>
<td>29.9</td>
<td>8.7</td>
<td>16</td>
<td>46</td>
</tr>
<tr>
<td>Little</td>
<td>Female</td>
<td>28</td>
<td>33.8</td>
<td>9.1</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Male</td>
<td></td>
<td>56</td>
<td>37.3</td>
<td>7.8</td>
<td>13</td>
<td>48</td>
</tr>
<tr>
<td>Some</td>
<td>Female</td>
<td>9</td>
<td>37.2</td>
<td>8.1</td>
<td>26</td>
<td>49</td>
</tr>
<tr>
<td>Male</td>
<td></td>
<td>41</td>
<td>40.3</td>
<td>7.4</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Lots</td>
<td>Female</td>
<td>3</td>
<td>34.3</td>
<td>11.4</td>
<td>25</td>
<td>47</td>
</tr>
<tr>
<td>Male</td>
<td></td>
<td>13</td>
<td>39.8</td>
<td>7.1</td>
<td>26</td>
<td>48</td>
</tr>
<tr>
<td>Unreported</td>
<td>Female</td>
<td>4</td>
<td>30.0</td>
<td>12.7</td>
<td>18</td>
<td>48</td>
</tr>
<tr>
<td>Male</td>
<td></td>
<td>34</td>
<td>33.0</td>
<td>9.9</td>
<td>8</td>
<td>50</td>
</tr>
</tbody>
</table>

4.3 The Effects of Gender

4.3.1 Intuitive Explanations

There are several “intuitive” reasons for the difference in test scores by gender, but none of them are strongly supported by the data:

- **Do pass/fail students skew the distribution for females?**
  The pass/fail percentage of females is roughly 1.5 times greater than males, and pass/fail students tend to do worse, in general, than average. Removing all of the pass/fail students from the data set does not eliminate the relationships between gender, year, prior programming experience and enrollment in CS50, however, although it does slightly weaken their statistical significance.

  Instead of treating the pass/fail percentage as a predictor of how students perform, I suspect that it should actually be treated as a covariate to some hidden variable that is related to gender.

- **Does year play a role?**
  Controlling for year does not change the nature of the relationship either. Although there is some variation in the proportion of students of each gender by year, and a variation of scores by year, these variations do not explain the relationship between test scores and gender.

  In fact, it seems to deny this relationship: the sophomore students, who are the class with lowest percentage of females, also have the lowest average test scores, while the freshman class, which has a higher percentage of females than average, also has the highest test scores. Year is clearly a useful predictor of test scores, but does not explain the gender differences.

- **Does the level of prior programming experience play a role?**
  The level of self-reported prior programming experience appears to have a relationship to scores on the first midterm, as shown in figure 4.5. However, there does not appear to be a convincing relationship between level of prior programming experience and scores on the first four assignments.³

³I have a table of year, gender, level of prior programming experience, and scores on the first four assignments and the first midterm, provided by Professor Mango Seltzer. I do not have data for the rest of the semester.
It would be interesting to see whether there is a relationship between prior programming experience and time spent per assignment, but unfortunately I do not have the data necessary to do this analysis.

4.3.2 Prior Experience and Enrollment Differ by Gender

Male students are more likely to enroll in CS50, and are much more likely than females to claim to have prior programming experience and to continue to CS51. This effect was illustrated in figure 4.4. This effect also contributes to the effect illustrated in figure 4.6, which shows that in addition to the relatively large number of males who enter CS50 with substantial programming experience, there are a similar number who feel confident enough in their prior experience to completely skip CS50 and proceed directly to CS51.

In the 1997-8 school year, 27% of the students in CS50 were female, but only 18% of the students in CS51 were female. The odds that a male CS50 student continued on to CS51 were 2.7 times higher than that of a female student.

There is some reason to believe that this effect is not due to anything that happens in CS50, but instead is a more general sociological effect. Indeed, as reported in [SS98]:

\[ \ldots \text{, the largest barriers to retaining women in computer science may be circumstances that occur long before they enter our program. \ldots they may be systemic societal problems or may be caused by the early education process. But the simple observation remains: at the time women enter our program, they have had less experience with computing and do not intend to continue in the major.} \]

It may be the case that many females who enroll in CS50 have no intent to concentrate in computer science, but instead are taking CS50 simply to fulfill a requirement or because they believe that it will teach useful skills. It would be interesting to ask students entering CS50 whether they are considering concentrating in computer science, to see what fraction of the female students consider computer science a viable concentration before and after taking CS50.

4.3.3 Test Grades Differ by Gender

Male students do significantly better on the CS50 tests. This was true in both 1996 and 1997. The differences are summarized in figure 3.3 (page 8).

The difference for the first midterm is particularly interesting (or dismaying) because it occurs very early in the semester (before study habits have really had a chance to form) and is also the first time that students have a chance to get a feel for how they are doing in relationship to the rest of the class, because the grade distribution of the first hourly is the first grade distribution made public.\(^4\) It must be a somewhat dismaying experience for the females in the course (and the freshman females in particular) to discover that they are already apparently behind the curve. It would be interesting to investigate what effect this has on the attitudes of the females in the course toward CS50 and computer science in general.

4.4 Quantitative Measurements of Differences in Work Habits

Do male and female students have different work habits? With respect to the possible factors that I have investigated so far, I find no convincing reason to reject the null hypothesis that male and female students have similar work habits.

\(^4\)The grade distributions for the assignments are never made public, so females do not get a chance to see that they are doing as well as the males on the assignments— the only distributions they get to see are exam distributions, on which they tend to do worse.
Figure 4.6: Enrollment patterns of CS50 students, separated by gender.

- **CS50 Only**
  Students who enroll in CS50, but do not enroll CS51 the next semester.

- **CS50 and CS51**
  Students who enroll in CS50, and then do CS51 the next semester.

- **Skip CS50**
  Students who enroll in CS51 without having taken CS50 at all.

![Enrollment Patterns Diagram]

Data about CS51 1998 enrollment was provided by Chris Thorpe and Professor Henry Leitner.
4.4.1 Do Females Spend More Time Programming in CS50?

An extreme hypothesis is that the females who take CS50 require, on average, more time to solve certain kinds of problems than males, although given enough time their solutions are no worse than the solutions discovered by males. Since there is no limit on how much time can be spent on the assignments, the female students can work harder and spend more time and achieve the same scores. On the tests, time plays a critical role\(^5\), and so female scores suffer.

If this hypothesis was true, then since females achieve roughly the same grades on their assignments as the male students, they must be working longer on these assignments. This theory is not supported by the data; in none of the assignments do the female participants spend significantly more time on their assignments than the males. In fact, assignment nine is the only assignment where females and males spend significantly different lengths of time, and on this assignment females spend less time than males (4.9 hours on average for the females, compared to 5.9 for the males).

Examination grades are a powerful predictor of how long a student spends working on their assignments, however, so it makes sense to repeat this test, controlling for test grades. Controlling for examination grades (and testing for an interaction between gender and the relationship between test grades and time spent on the assignments) shows that for all but one of the assignments, gender has no effect on time. The solitary exception is assignment 4, where gender, test grades, and the interaction between gender and test grades are all statistically significant predictors of how long it will take to complete the assignment.

Although this data does not support the hypothesis that females spend more time working on their programming assignments than males, it cannot refute it either. It may be that females are spending more time preparing their assignments (making notes to take with them to the terminal room, attending review sessions, etc) than males. I cannot measure this time, however, and so the question is not settled. However, it is clear that females are not spending more time participating in activities that are monitored by this study.

4.4.2 Do Females Start Their Assignments Earlier?

One piece of data I can derive from my study is how many days elapse between the time students start working on their assignments on the computer and when they stop. Using this information, I can begin to approach the question of whether females start their assignments earlier.

This data has two flaws, however:

1. As in the previous section, this data does not reveal when the student actually began whatever activities they do prior to their first programming session for an assignment. Some students spend a considerable amount of time before starting to use the computer, but I have no way to estimate this time. It would be extremely interesting to see if females start preparing before or after the males, but I have no way to know.

2. Because I do not know when the student’s assignments are actually due, I cannot actually determine how far in advance of the due date they began working. Based on my own experiences, I assume that most students turn in most things on time, or at least soon thereafter, and so this should not be a concern. To learn this data, I would either have to know what section each student is in (since due dates are determined by section times) and possibly which students have extensions and how long these extensions are. The possibility of gathering this information as part of the database was purposefully omitted from the design, however, because it would represent a potential breach in the anonymity of the participants.

\(^5\)The CS50 tests are notorious among the students for being too long.
Another possibility would be for individual TFs to record, along with their grades, how many days late each assignment was actually turned in. This was purposefully omitted from the design of the database, however, because it makes too much work for the individual TFs. If this study is repeated, however, it would be very useful to find a way to gather this information.

Based on the information that I have, there is no significant difference between the number of days that male and female students spend working on the assignment.

However, there is a positive and significant correlation between the number of days that a student spends working on an assignment and the total time spent on the assignment. Controlling for this effect, however, reveals no relationship between gender and days spent.

Similarly, there is a negative and significant correlation between a student’s test scores and how much time they spend on each assignment. Controlling for this, however, again reveals no relationship between gender and days spent.

4.4.3 Do Females Spend More Time Attending Office Hours?

Are females more likely to go to office hours than males, and among students who attend office hours at all, do females typically spend more time than males?

Again, my data is imperfect— I can determine whether or not a student is running on one of the workstations, but I cannot tell whether they are actually at “office hours” and asking questions or whether they are simply doing their work in the science center (or whether they are logged in to one of the workstations remotely6).

My data does not support the theory that females spend more time in office hours than males, nor is the gender distribution of office hour attendees different than that of the general population of CS50 students during the latter part of the semester. However, it does appear to be true that females are more likely to come to office hours earlier in the semester than males: almost 80% of the females who attend office hours do so by assignment 2, compared to approximately 60% of the males.

4.5 Gender Differences, Role Models, and Society

This data does not appear to show any clear reason why females consistently do worse than males on the tests, nor are the work habits of females generally different than males (with the possible exception of office hours attendance).

It may be that females are more prone to falling victim to stress during tests, or may simply expect to do worse due to societal or cultural biases.7 A student who does not expect to do well or who expects that their efforts to do well will be in vain may well decide not to waste time studying more.

4.5.1 Discussion

The notion that women and men have grossly different innate cognitive abilities is no longer given any serious consideration by psychologists, although there is experimental evidence that cognitive differences

---

6Originally I believed that the system was configured so that students would only be able to log in to the workstations via the console, so if a student was logged in to a workstation, they must be sitting in front of one. This turned out to be the case, and occasionally students would log in remotely. I did not have any way to monitor and compensate for this in an anonymous way; however.

7One female student told me that she expected to do worse on the tests than males, and that it is “common knowledge” among her female peers that men do better on tests. Another female student told me that she had asked all her friends in CS50 what they had gotten on the first midterm, and had noticed a strong bias for the males to get higher grades.
between the genders do exist and can have measurable, though usually unsubstantial, effects on performance in common activities. Fernald and Fernald ([FF85], page 390) sum this up succinctly:

"The question of sex differences also can be dispatched summarily. There is no significant difference between the sexes in global intellectual capacity. This finding, suggested in small samples of women in World War II, has been corroborated in hundreds of studies, ranging from grammar school children to psychiatric patients to college students (Matarazzo, 1972). There are, of course, specific test items on which males and females perform differently. . . . The slight differences in specific abilities fit the popular notions. Females tend to perform slightly higher on vocabulary items; males show somewhat greater success in mathematics. It is not yet known whether these minimal findings reflect cultural influences or innate differences in cognitive functioning (Hyde, 1981; Majaeres, 1983)"

Much of the research on this last area—whether these differences are innate or learned—is qualitative and introspective. There have been studies of young children that show that gender differences appear at a young age, perhaps before societal influences are a factor [RDHDGQCI90]:

... females get a higher score in primary abilities of verbal understanding, fluency, and memory, whereas males outstand in special primary ability and inductive reasoning. In numerical ability and perceptual quickness there are no significant differences except those mediated by age and culture.

No studies conclusively show that these differences are not caused (or at least increased) by cultural biases, however, and this issue is the root of much research and controversy. Much of the objective and quantitative research, however, is targeted toward very basic cognitive processes and not particularly helpful in our efforts to understand differences in more complex tasks that require a spectrum of cognitive abilities, such as CS50 tests. However, one experimental result in particular might be applicable:

Men, who are supposed to be the "stronger" sex, may be less inclined than women to admit mental distress in interviews or questionnaires. Supporting this view, experiments have shown that when men and women are subjected to the same stressful situation, such as a school examination, men report less anxiety than do women even though they show physiological signs of distress that are as great as, or greater than, those shown by women. (Polefrone & Manuck, 1987, from [Gra94])

This result supports the hypothesis that if male and female CS50 students suffer the same amount of "mental distress" during the course, female students are more likely to report that they are more stressed. (Whether this is an innate difference between men and women or a learned behavior inflicted on men and women by societal biases is an open question, but not important to this discussion.) An outside observer might easily conclude that females in CS50 are less happy, or doing less well, than their male counterparts even if the females they observe are doing exactly the same amount of work and getting exactly the same grades as the males. Polefrone and Manuck might conjecture that all other things being equal, after a difficult semester of CS50, a female is less likely to be advised to take CS50 in the next year by her roommate than a male is by his.

This leads to a difficult dilemma. Haller & Fossum [HF98] at the University of Wisconsin believe that more peer role models for females are necessary to attract and retain females in computer science. As they state, “female role models presented to young women are often unusual individuals who have overcome tremendous social and political obstacles to pursue outstanding careers in mathematics and science.” Female (or male) students may find it difficult to “readily identify” with such role models. Haller & Fossum suggest

---

8A lot of this research also seems to be intended to serve a specific agenda as well.

28
that less lofty role models, taken from the peer group of the female students, may be more effective— but if Polefrone & Manuck are correct, then these peer role models may actually be counterproductive, because in direct comparison with their male counterparts these female role models may appear more stressed and anxious and will be more likely to give voice to their unhappiness.

Haller & Fossum are building an experiment to test their theories; it will be interesting to see their results over the next several years. In the meanwhile, CS50 has had its share of both lofty (Margo Seltzer) and peer role models (female TFs and UAs).

Another barrier to women entering the field of Computer Science may be due to the popular perception of Computer Science as an all-consuming activity that requires a huge commitment, in terms of time, energy, and other opportunities lost. This is true both at the undergraduate level and in the post-graduation professional world— a large percentage of the graduating Harvard CS concentrators are offered large salaries to work for companies that expect their employees to work long hours (to the point of personal sacrifice) and be extremely productive. Since these companies are selective in their recruitment process, and large salaries are generally equated with personal success, it is a positive measure of success to be recruited by them. Unfortunately, this type of success is often not as attractive to women:

(FF85], page 316) “High achievement, synonymous with masculinity, has been seen as incompatible with the traditional concept of the ideal woman. As a result some women learn to underplay their abilities, especially when in the company of men. In these instances, it is hypothesized, there has existed among women a motivation to avoid success, thereby providing greater opportunity for fulfilling the traditional female affiliative motive (Stein & Bailey, 1973).”

These tendencies are fading quickly as gender stereotypes are beginning to lose their pervasive power, but perhaps CS is such a competitive and achievement-oriented field that the effect will linger longer in CS than in other careers. If these stereotypes still have an effect on the choices that female students make, this may worsen the problem even more— females are not drawn to computer science because the few role models they have are seen as unusual, extraordinary people, while peer role models are few because undergraduate females, cautious of appearing to be “high achievers”, are unwilling to act as role models for their juniors, and provide less positive encouragement to them.
Chapter 5

Conclusion

5.1 Summary

The goal of this study was to explore the relationships between student work habits and how well they do in the course. I was not very successful in modelling these relationships, although my results are interesting and statistically robust. Students appear to succeed, and do so in a variety of different ways. The small number of variables that I measured, together with the limits of the accuracy of my measurements and the lack of control for other contributing factors combine to weaken the statistical precision of my results.

5.1.1 Accomplishments

- The data collection system itself was a success. It logged student activity correctly and efficiently, gathering data throughout the semester and storing it safely without placing a perceptible load on the system.

- Approximately half of the students did enroll in the study, even though participation in it was completely voluntary.

5.2 Directions for Future Research

- When do students really work?

  I have not even scratched the surface of what might be done with the time information present in the current logs. Possible research questions are:

  1. Are time patterns different for males and females?
     The analysis I have performed suggests that there is no significant difference in the amount of time that men and women spend— but do they spend it at different times?

  2. Are time patterns different for members of different classes?
     Do freshmen attend office hours more frequently, or stay later? (Perhaps they do, because they do not have to walk as far to get home afterward.)

  3. Do usage patterns change through the semester? Can we determine when students are “stressed” by watching for changes in their work patterns? Are changes in work patterns reflected in changes in grades?
• **Does lecture attendance matter?**

Lecture attendance was low this year, and it varied from time to time. Does lecture attendance have an effect on grades or time required to do the assignments? If there is an effect, is it localized to assignments related to the missed lectures, or is it chronic? What kind of student skips lectures¹, and is there any way to predict this? Do students get as much out of watching the videotapes as going to lecture (or more, since they can pause, rewind, and repeat topics they find difficult)?

• **Do “distractions” matter?**

For example, do students who leave an email window open do worse than students who do not? Do students who turn off messaging and automatic email notification do better than those who do not?

The amount of email that many students exchange is astonishing, and seems to increase every year. It has reached the point where some students are engaged in a virtual conversation with several of their friends whenever they login– whenever their mail program delivers another message, they drop whatever else they might be doing and compose an answer. I suspect that this is very distracting– but have no way to measure this, or even to identify which types of students are more likely to do this.

As alluded to in section B.1.1 (page 40), the question of whether what other things students are doing while they are working on their CS50 assignments is a tantalizing question that I currently have no means to answer.

Perhaps the level and amount of distracting influences is important, but right now there is no way to gather this information. With so many possible combinations, I suspect that no observational methodology will suffice– this might require an explicit experiment to investigate properly.

• **Does familiarity with the tools help?**

1. **Do students who learn more about how to interface with the computers do better than those who do not?**

   The percentage of students who use the 21” monitors on the workstations solely to display one 24x80 10-point window is dismaying. In addition, the default window manager on the DEC-Alpha workstations is (at least in my opinion) unfriendly and unlike anything that the students are likely to have used before. Similarly, there are many students who do not take advantage of their telnet program’s ability to manage multiple sessions. Do students who learn how to use these utilities do better work, or work faster, than students who do not?

   In my own experience, these things are very important. When I figured out how to change the default font used by my window manager so that I could have two 80-column windows side-by-side on the screen, it changed the way I programmed for years to follow.

2. **Do students who learn more about the editor do better than those who do not?**

   Most students learn how to use only a tiny fraction of the power of vi or emacs. It would be interesting to see if students with more knowledge of their editor complete their assignments more quickly.

   It would also be interesting and important to see whether the potential reduction in typing reduces the number of students who suffer from RSI.

3. **Do students who learn more about the debugger do better than those who do not?**

   Most students learn how to use only a tiny fraction of power of gdb. Is more truly useful for the sort of debugging problems faced by CS50 students? It currently appears that many students do not use the debugger at all (see section 3.4).

¹ As Mark Kernighan pointed out, this is probably the same sort of student who doesn’t participate in studies, doesn’t fill out a CUE form, and is generally a hard person to measure anything about.


Appendix A

Data Collection Methodology

A.1 Recruitment of Subjects

Subjects were recruited from CS50 on a purely voluntary basis. Early in the semester, as soon as the data collection system was completely functional\(^1\) and ready to run, I gave a short announcement about the study—how to enroll, what data would be collected, and what would be done with the data.

A copy of the information form is online at:

http://www.fas/"eillard2/research/cs50-ann.html

Subjects were also recruited in the same manner from E50a, an extension school course similar to CS50. Very few people from this course volunteered, however, and I did not gather enough data to make its analysis worthwhile. Therefore, I have not performed any in-depth analysis of the E50a data.

A.1.1 Volunteer Bias

One of the first questions raised about any study where the participants are volunteers are whether there is a volunteer bias— is there a difference between people who volunteer for such studies and the population as a whole, and is this difference related to any of the effects being studied? Unfortunately, without any control over who volunteers (or in our case, with very limited knowledge about who has volunteered), it is impossible to know [LSW90]. There is some reason to anticipate that there will be a difference, due to effects such as the Hawthorne effect [FF85, Wie71]— people who are being observed (and know that they are being observed) tend to do better work than people who are not. In the case of CS50, however, all of the students are being “observed” in the sense that they are all being graded on their work, and all desire to get a good grade. In addition, the data for the study was gathered in such a way that the students were not reminded of the fact that they were being studied— there was no perceptible difference between being studied or not. Therefore, I believe that the Hawthorne effect is not likely to have a substantial effect on my data, although it may be present.

To test whether the volunteer bias had any gross effects on my sample, I compared the population of the volunteers to the population of the class as a whole. With respect to gender and class distributions, they were not significantly different. There was also not any significant difference between participants and

---

\(^1\) The data collection system was not ready at the beginning of the semester because of two reasons: first, I thought that I needed to improve its performance in order to reduce its load on the system, and second because I needed the cooperation of the key holder (see section A.3). It turned out that I had greatly overestimated the load on the system, and performance was never an issue at all. The biggest problem was that the key holder was extremely busy at the beginning of the semester.
nonparticipants with respect to assignment and test grades. Therefore, I concluded that volunteer bias is probably not an issue in this data.

A.2 The Data Collection System

The information about student work habits is gathered by the data collection system. The general architecture is very simple: new versions of the programs whose activity I wish to monitor (make, gdb, the ANT tools, and some test scripts) are placed in the students’ paths. These replacements are wrappers around the real programs, and behave identically to the original programs but also log the activity of the program.\(^2\)

The wrappers are described in more detail in section A.3.

The wrappers send their information to the logger, which is responsible for keeping a log of the data for future retrieval. For reasons of security and practicality, the logger generally runs on a different host than the wrapper. In the current implementation, the logger runs on the same machine that hosts the EECS web server—this is a powerful machine, and not heavily loaded, and has local disks that can be used to store the log files temporarily. To greatly simplify the communication protocol between the wrapper and the logger, I used the HTTP CGI POST protocol [Mor95]. The EECS WWW server sees each incoming connection from a wrapper as a request for a web page, and dispatches the appropriate CGI server to handle it. In this case, the appropriate CGI server is the logging daemon.

Using the HTTP protocol and an unmodified WWW server was a calculated risk— the HTTP protocol adds overhead, but completely encapsulates the network abstraction. This allowed me to update or modify the logger daemon without making any changes to the wrappers.

One potential problem with this approach, however, is that the HTTP timeouts are quite long. If the EECS WWW server was down or overburdened, the wrapper might have to wait for a long time (or forever) to communicate with the logger. To prevent this from having an impact on the user, I added a timeout to the wrapper so that if WWW server could not be contacted within two seconds, the attempt would be abandoned. In the worst case, a user’s make might appear to hang for two seconds after finishing, but no longer. In practice, I never received any complaints about this.\(^3\)

In tests before the start of the study, I was able to saturate the WWW server, however, by simulating a very heavy job load on the FAS machines. I was worried that this might happen in actual use, but apparently my estimates of how frequently the students would use the compiler were much higher than ever occurred. If I thought that the system was falling behind and losing messages, however, I could have reconfigured it to distribute the logging over several different machines (each running its own WWW server)–this capability is implemented and tested in the wrappers, although never used during actual data collection.\(^4\)

The logger generated a tremendous quantity of logs, which were then processed by a battery of PERL scripts to extract and summarize the appropriate information from the logs.

---

\(^2\)There only detectable differences are that the new programs run measurably (although imperceptibly) slower than the originals in some cases, and that the stdout and stderr streams are merged into stdout for make and as. To fix this at all would have been quite complicated, and to mimic the precise behavior of make with respect to stdout and stderr is probably impossible. Margo and I decided that it wasn’t worth the effort to try.

\(^3\)Unfortunately, it was impractical, given the current system, to obtain any measurements about how many messages were lost—the only way that the wrapper can indicate that it timed out while sending a message is to send another message, which would only make matters worse if the logger was already falling behind. However, my torture tests of wrapper/logger throughput convinced me that the system would successfully handle a load several times that which appears at any point in the logs, so I do not believe that the system dropped many messages.

\(^4\)A large source of anxiety on my part during the data collection was the reliance on the EECS WWW server. I was assured that it was reliable and carefully maintained, but as the semester progressed I realized that this was not the case, and that its host was not even administered by the EECS sysadmins! Eventually, the inevitable occurred—www.eecs suffered a catastrophic head crash and lost the entire partition that the web server and the logging daemon used, and was down for days. Luckily, this occurred after the data collection effort had ended.
A.3 The Anonymizer

One of the interesting and challenging aspects of the data collection facility is the need for anonymization and security of the data gathered by the database. This section describes the protocol I devised to provide this security.

I will begin by describing the purpose of the protocol, and my current implementation of the protocol. I will then outline the flaws in the current setup and how they might be exploited. In the last section, I will describe a more general framework that fixes these flaws to provide an extremely high level of anonymization.

A.3.1 The Need for Anonymization

In my study, I need to join records that contain information about students from several different sources. However, it is forbidden for me to know the actual identity of the student who is associated with each record.\footnote{When this protocol was being designed, I planned to teach a CS50 section and hence might actually be dealing with information about my own students that as a TF I would not ordinarily be allowed to access. Since I did not actually serve as a TF during the study, this requirement could probably have been relaxed.}

Similarity, I might desire to share my information with other people, in such a way that they can join their private records with my own, but without learning anything about the identity of each student that they didn’t already know.

For example, Professor Margo Seltzer gave me information about each student’s class, gender, and grades, in such a way that I could join these with records containing my records about their work habits, but without ever knowing the names of any of the students. I could also send my database of student activity records to Margo, and she would be unable to determine the identity of any of the students from any of the information I gave her.

Finally, not only is the identity of each student secret, but the set of students enrolled in the study is secret—enrollment is done in such a way that no single entity except the students themselves can know for sure the name of even a single person enrolled in the study.

As I will show, the current implementation is effective at keeping information during the data gathering phase, although it can be compromised during the data analysis phase, if I shared some specific information with Margo. So far, however, this has not been necessary and I do not anticipate that it will be.

The Protocol Parties

There are five parties in the current protocol:

- **The students.**
  
  All students wish to keep their identity secret. Some students may elect to share information about their work habits, but not at the cost of revealing their identity.\footnote{Of course, if none of the students whatsoever enrolled in the study, anyone with a class list would know the set of identities of all the nonparticipants— and if all the students enrolled, they would know the set of identities of all of participants. However, they still would not know who was who.}

- **The key holder.**
  
  The key holder is a disinterested third party. The job of the key holder is to generate (and remember) a single key that is used as the central key of the cryptographic system used by the protocol. In the current system, Paul Bergen (a member of the FAS computer support staff) is the key holder.

- **The log holder.**
The log holder has access to the logs of student activity generated by the monitoring and logging system. In the current system, I am the log holder.

The logs contain information about the identity of each student, but this information is encrypted via the keyholder’s key, which I do not know.

- **The grade holder.**

  The grade holder has access to the grades (and gender and class) of each student. The grade holders in the current system include anyone with access to this information—most notably Professor Seltzer, but also all the CS50 staff.

  There must be no way that a grade holder can learn anything more about a student than they would be able to learn from the grade files—and that includes grade holders who also happen to be log holders (which I thought would be the case for myself).

- **The analyzer.**

  The analyzer is responsible for analyzing the union of the information held by the grade holder and the log holder. The analyzer should not be able to learn anything about the identity of any of the students in the study, and should not reveal anything to anyone else that might make it possible for them to learn these identities.

  In this study, I was both the analyzer and the log holder.

The Trusted Programs

1. **The uid2anon program.**

   The uid2anon program is a simple program that determines the UNIX UID of the user running the program, combines this with the key held by the keyholder, encrypts the resulting string using DES, and prints it. The resulting string is called the AnonID of the user.

   Because the AnonID is computed for the user running the program, any user can find their own AnonID. Since the mappings between UIDs and AnonIDs is determined entirely by the key (and some salt built into the uid2anon program), there is no need to maintain an explicit mapping of UIDs to AnonIDs—every unique user of the FAS computer system is in effect given their own unique AnonID (whether or not they participate or are even aware of the study).

   In order to make the uid2anon program fast, the key is compiled into the program. Therefore, uid2anon must be installed execute-only. In the current system, the uid2anon program is created automatically by the key holder—the key holder runs a script that prompts him or her for a key, and then creates and compiles a tailor-made uid2anon program.

   The uid2anon program must be trusted to generate a unique AnonID for each user, and that the mapping between the UIDs and AnonIDs must be as difficult to invert (at least as difficult as DES).

   The uid2anon program itself is implemented by less than 100 lines of C. The script that builds the uid2anon program from a key is less than 200 lines of PERL.

2. **The wrapper.**

   The wrapper is the program that students actually execute when they run the commands that the system monitors. The wrapper does the following:

   (a) Determines if the student is enrolled in the study or not. A student enrolls in the study by setting an environment variable (usually via their .cshrc). If the environment variable is not set,
then the program that the student intended to run is executed as normal, and the wrapper exits. Otherwise, the wrapper continues to the next step.

(b) If the student is enrolled in the study, the wrapper determines, from the name of the command, what information about the command to gather. (For compilations, the wrapper gathers the error messages, while for other commands it might just note the fact that the command was run.) The information always includes the current time, directory, and the name of the command being run.

(c) The student’s AnonID is determined, by running the uid2anon program.\(^7\)

(d) The command is run, and the proper information is gathered.

(e) Identifying information about the student (or information that might identify the student, such as the full path to the current directory, etc) is removed from the information gathered. Where appropriate, it is replaced by the student’s AnonID.

(f) The information is sent to the logger. The logger simply accepts messages from the wrapper and logs them to a file—the logger runs on a separate machine and has no information about the activity being logged except what the wrapper chooses to tell it.

The wrapper must be trusted to:

- Correctly remove any identifying information other than the student’s AnonID from the information sent to the logger.
- Send the information only to the logger, and never store it in a public place.
- Send only valid information, in a format that the logger can understand.

The wrapper is implemented by approximately 1,000 lines of portable C.

3. The uid2anonList program.

The uid2anonList program is a companion to the uid2anon program, but is never publicly installed and can only be used by the key holder. uid2anonList takes a list of \(N\) UIDs and creates two maps: a map from UIDs to a random permutation of \(N\) unique numbers, and a map from AnonIDs to the same random permutation. If these maps are joined by the random numbers, a map from UIDs to AnonIDs would be created—but this join is never performed. The key holder runs uid2anonList at the end of the semester (after grades are filed) and sends the first map to the grade holder (who already knows the UIDs of each student) and the second map to the log holder (who already knows the AnonIDs of each student in the study).

uid2anonList must be trusted to create the maps correctly, but not share them with anyone except the correct recipients.

uid2anonList is implemented almost entirely as a 114-line PERL script and a short C program (created by the same script that creates uid2anon).

Note that the logger and the analysis tools do not need to be trusted in the same manner as these programs—they do not have access to the identity of the students, and so there is nothing that they can reveal.

\(^7\) Theoretically, uid2anon could be combined with the wrapper, since the wrapper is the only program that ever runs uid2anon. However, this can be impractical, because changes to the wrapper would require cooperation with the key holder, which in practice was difficult to coordinate. I updated or extended the wrapper several times during the semester (including porting it from PERL to C in order to increase its speed and decrease its use of system resources) but the uid2anon remained unchanged.
A.3.2 The CS50 Study Protocol

1. Before the study begins.
   (a) The keyholder runs the CreateKey script, which creates $\text{uid2anon}$ and $\text{uid2anonList}$, and installs $\text{uid2anon}$ as a publicly executable but unreadable program.
   (b) The wrapper is installed around all programs of interest.
       For CS50, the set of programs were make, gdb, aa, ad, and the ANT test suites.
   (c) The logger daemon is enabled.

2. At the start of the study period.
   Consent and information forms are distributed to the students. Students enroll/unenroll in the study by setting/unsetting an environment variable.
   No observer can tell whether a student is enrolled at any given moment except the student or anyone with direct access to the student’s environment.\(^8\)

3. During the study period.
   Logs are collected about student activity. The logs are kept on a system owned by the log holder, and are not publicly readable. The only information about the identity of the student who created each of the records is their AnonID, which cannot be connected to any UID by the log holder. The key holder, who can determine which UID corresponds to each AnonID, does not have any access to the logs.
   The log holder does not share the logs with anyone.

4. At the end of the study period.
   (a) The key holder runs $\text{uid2anonList}$ and sends the resulting UID → NID map to the grade holder, and the AnonId → NID map to the log holder. The grade holder and log holder respectively replace the UIDs and AnonIDs in their data with the NIDs.
   (b) The grade holder and log holder send their resulting files (with all identifying information in the form of NIDs) to the entity doing the analysis.\(^9\)
   (c) The entity doing the analysis joins the databases from the log holder and grade holder by NID, and then replaces each NID with a unique, random RID.
   (d) The entity doing the analysis uses the resulting RID data for their analysis.
      At this point, the data set can be revealed to the grade holder (or even the key holder) without loss of anonymity. The analyzer is the only person who knows the map from the NIDs to the RIDs, and he never reveals it (and can destroy it once the join is complete, to prevent it from ever being revealed).

---

\(^8\)Unfortunately, direct access to the student’s environment is often granted to TFS at the beginning of the semester, when TFSs in office hours are asked for helping fixing dot files or other configuration issues. During this time, the TFSs could, if they desired, determine whether or not a student was enrolled in the study at that moment.

\(^9\)For this study, I was both the log holder and the entity doing analysis.
A.3.3 The Psychology of Anonymization

I designed the system to meet both the letter and the spirit of complete anonymity requested by the committee on the use of undergraduates as subjects, and I believe that the degree of true anonymity is higher than that of similar experiments conducted elsewhere.

However, this degree of anonymity may actually have been overkill, at least in one respect. The participants will not trust the anonymizer any more than they trust that administrators of the system.\textsuperscript{10} Although we made an earnest attempt to explain what data was gathered and that it would be anonymous, this message apparently did not reach (or was not believed) by everyone.

I believe that it is possible that there were students enrolled in the study who might have preferred to not be enrolled, but did so because they felt that it would earn them good will. There were certainly students at the other end of the extreme, who declined to participate because they were afraid that we would put the data to nefarious use.\textsuperscript{11}

A.3.4 Weaknesses in the Implementation

- **Participant Identity Can Be Determined via Covert Channels**

  Although the identifiers themselves do not reveal any identifying information, the data that they are associated with could be used, by a careful attacker, to uniquely identify a particular student via instance deduction \cite{Schneier96}.

  The grade holder can potentially identify students by using the near uniqueness of grades. For example, if NID 42 is assigned to a student who received a unique grade on a particular assignment (i.e. they were the only sophomore female who got a 53 on assignment 3) the grade holder could work backwards from their grade files to determine the UID and true identity of NID 42.

  This problem could be reduced by quantizing the grades, to decrease the probability that a student’s grades uniquely identify them. This only reduces the problem, however, since the data would have be severely quantized before this attack would be defeated, and such quantization would profoundly weaken any analysis of the grades.\textsuperscript{12}

  The log holder can also deduce the identity of subjects, with some effort, by carefully monitoring the use of the FAS system. Over a long enough period of time, it is probably possible to correlate the list of people logged in at any given moment with the list of NIDs appearing in the logs at that time. (Students who work at odd hours, or over break, would be particularly easy to find.) I did not attempt this attack, but I suspect that it could yield a large amount of identity information if applied diligently.

- **Inappropriate Delegation of Trust**

  A key weakness of the current instance of the system was that the role of the implementer of the system and the log holder were both played by the same person—myself. I could have subverted the system in any number of ways and gathered more information than the protocol suggests. Of course, I did not

\textsuperscript{10}At least one student declined to participate in the study because he did not understand what data was gathered or what it was being used for. He had some very imaginative theories, which he shared with his friends, perhaps leading to several withdrawals from the study. (This was related to me by the student in March.)

\textsuperscript{11}Of course, if we were truly nefarious, we could have simply gathered an Orwellian collection of data about every CS50 student, without obtaining their consent at all. The technology to gather fine-grained information about everything that each user does is certainly within our grasp—only ethical considerations (and the rules of the University) prevent us from doing so. This, however, is not the sort of thing that you point out to a student who already doesn’t trust the system.

\textsuperscript{12}One reviewer suggested adding or subtracting a point at random from each score. This gives the attacker more work to do, but does not prevent identification. There are enough distinguishing characteristics about the grades that masking off a bit or two doesn’t help. In fact, it is necessary to “mask off” the lowest three bits (by dividing all scores by 8) before any pair of students have identical grades.
do so, because this information is uninteresting to my research— and because having this information in my possession would get me in all kinds of trouble. Therefore, I built the system with the purest intentions and did not leave any secret back doors that would enable me to discover the encrypted identities of the subjects in the log files. The lack of a back door caused me some amount of anxiety until final data was in my hands— if the keyholder had lost the key, there would have been no way to reconstruct the mappings.

Another weakness was that some sensitive parts of the system were installed in the 11b50 account, which is writable by many people, including the holder of the grades. In theory, any of these people could have written wrappers around my wrappers which extracted nearly any desired information. To avoid this, I regularly inspected the wrappers for changes, and I did not detect any.

Both of these problems could be avoided by limiting write-access of the wrappers to a trusted authority who could verify that the code did not contain any back doors, and could make sure that it was not replaced by an unauthorized copy. Unfortunately, trusted parties willing to inspect code, install, and monitor it are not easy to find. However, the body of trusted code is small, and therefore this is not a completely impractical approach for people who require higher levels of security.

- **Unencrypted Data Transfer and Storage**

  The communication between the wrappers and the log database was done in plain-text, using a text-based protocol that could be easily decoded. The log files themselves were unencrypted (though well protected from prying eyes), for easier browsing and processing. I assumed that the network between FAS and EECS was reasonably secure, and the EECS machines themselves were secure. Both of these assumptions turned out to be incorrect, unfortunately. The EECS machines were victims of a series of attacks and break-ins during the Fall and Winter. Several packet sniffers were discovered on the EECS subnet, and several accounts, including my own, were compromised. If the intruder was interested in sniffing the packets to the log database, or simply viewing the log files on the disk, they could have done so with ease. In fact, they could even have modified or deleted the log files, a fact that caused me great anxiety. However, I believe that the intruder was not interested at all in my data, and only in using my account as a springboard to more interesting accounts, so I believe that the log files were not stolen or modified.

  It is important to note that the translation between UID and AnonID takes place before the log entry messages are sent across the network to the logger. Therefore, even the most successful thief of my data would not be able to know more than I know about the identity of the subject in each log entry. There is no way for an intruder to learn the identity of any of the subjects in the study from the logs stored on the EECS machines or transmitted over the network.\(^\text{13}\)

\(^{13}\)Unfortunately, the grade files were kept on machines on the same network, and could have been compromised. The possibilities are chilling.
Appendix B

Analysis of the Study Data

B.1 Description of the Variables

The variables gathered by the study include the following:

- Grades for each of the homework assignments,
  Grades were provided for all of the students in the course, not just those students who participated in the study.
- Class, gender, and pass/fail status of each student in the course.
- For each student participating in the study, the time, commandline, and result of each compile and debug session, for each assignment. Whether the command was executed on a server or a workstation was also recorded.

From the trace of the activity of a student, *sessions* were identified by clustering the commands into groups. The clustering algorithm has two parameters: $g$, which specifies the maximum intra-cluster gap between events, and $r$, which specifies a fudge-factor to use to estimate how much time the user actually spent working in the session before and after their last logged event (generally spent editing, which is not logged).

Each $(g, r)$-session is defined to contain the longest possible time-ordered sequence of events such that no event is separated from the subsequent event by a gap of more than $g$ minutes. The length of the session is defined to be the difference between the time of the last event and the first, plus $r$ (the “roundup factor”).

For my analysis, I used a $(60, 15)$-session. Each session is therefore a sequence of events separated from each other by no more than an hour. A discussion of this choice of parameters is given in section B.1.1.

B.1.1 General Criticisms of these Variables

Criticisms of the methodology used by my study must include the following weaknesses in the variables my analysis is based upon:

- **Success in CS50 is hard to measure, and grades are hard to interpret.**
  The current method of measuring the success that a student has in learning the material in CS50 is based on a composite of their scores on the assignments and the exams. This method of assessment is
a somewhat questionable measurement of actual achievement (although I do believe that it is likely to be strongly and positively correlated with actual learning).

A better method might involve a better test of each student’s ability to program at the end of the semester, or might require tracking the student through several subsequent courses to see what patterns emerge.

It is frustrating that assignment grades, one of the most important outcome variables of my models, are at best an imperfect measure of achievement, and at worst can be somewhat misleading. There are several particular weaknesses in the assignment grades as a measure of achievement:

- **Variability of ratio of coding to written problems on different problem sets.**
  Some problem sets require a relatively small amount of coding and a larger number of pencil-and-paper or conceptual problems. For these assignments, it is hard to estimate how much of the score is due to programming habits and how much to other factors.

- **Variability of grading standards.**
  The grading standards themselves evolved through the course of the semester, so the same kinds of errors would receive different deductions on different assignments. This makes analysis across assignments very difficult, if not impossible.

- **Variability of grading practices and procedures.**
  There is a persistent belief that Tfs apply the grading policies inconsistently, and so the section that a student is in may have an effect on their assignment grades. I do not have the data necessary to test whether such an effect exists (or control for it if it does).

- **Deductions due to lateness or other procedural issues.**
  A student may lose a considerable number of points per assignment for procedural issues such as lateness, sloppiness, not printing all materials, or neglecting to submit various ancillary files. I do not have the “raw” scores before such deductions are taken, so I do not know how much of the variability in grades is explained by this.

- **A complete analysis of tool usage and work habits is impossible with the current monitoring methodology.**
  The algorithm for detecting sessions is too simplistic. It was based on an assumption about student work habits that I no longer believe is valid.

  The current session definition can result in a large number of short sessions for each student, separated by long periods of time. It may be that the students are editing for extended periods of time, or nulling over some aspect of the problem for very long periods of time between compilations or debug sessions. However, I have no way to tell whether they are doing this or whether they’ve gone to dinner or are watching TV for an hour.

  Although I cannot be sure, I strongly suspect that my initial assumptions about typical programming habits were quite incorrect. Unfortunately, these assumptions guided my design of the monitoring system. The problem is that students use the tools much less than I anticipated- I assumed that student activity would consist of a fairly regular edit-compile-debug pattern, and so it would be able to identify people working on their assignments by monitoring compile and debug activity. However, it now seems common for a student to spend an hour or more editing between compilations- but because I do not monitor editing sessions, I cannot tell whether the students are editing during this time or whether they are doing something completely unrelated to their assignments.
Ironically, the fact that students are not compiling (and using the error and warning messages generated by the compiler to guide their editing) as frequently as I expected is one of the more interesting results of the study.

- **What are students doing during all that time?**

  This criticism is related to the previous; it appears that the number of distractions available to students while they are trying to work on their homework is very large, and may play an important role in how students get their work done.

  When I attempted to learn how to program as an undergraduate, there was nowhere to do programming except in the basement of the Science Center, and in the terminal room there was little else to do besides programming. Today a student working on a computer has innumerable distractions—email, USENET, WWW, games, as well as their other coursework (much of which can be done on the computer as well). Computers today are not “workstations” so much as a concentrated focus of distractions. Students do not appear to work on their assignments in clearly delimited periods of time, but instead seem to work on several things at once.

  Any followup study should make every attempt to gather information about what other activities students are engaged in (or distracted by) at the same time as they are working on their CS50 assignments. Unfortunately, in the current study I cannot monitor the non-CS50 activity of a student in the same manner as I monitor their coursework.  

---

**B.2 Summary Statistics and Discussion of Grades**

A summary of the CS50 grades for 1997 for the students who participated in the study is shown in figure B.1. These grades are not significantly different than the grades for the entire class.

**B.2.1 Midterm, Exam, and Assignment Grades**

Histograms of the distributions of the scores on each hourly and the final exam for each participant in the study are shown in figure B.2. Histograms of the grades for the homework assignments are shown in figures B.3 and B.4.

**B.2.2 Time Spent on Assignments**

The amount of time spent on each assignment is computed by summing the session times, as described earlier.

A summary of the raw statistics of time spent by each student on each assignment is given in figure B.5. These numbers are based on estimates of how much time each student spent working at the computer on their programs, as described in section B.1.

Assuming that these numbers can be believed, one thing is clear— for most of the assignments, the majority of students are spending less than 10 hours per week working on the computer. This contradicts the persistent rumors that most students in CS50 spend most of the semester hacking during every waking moment. Instead, it shows that the amount of time spent is typically well within the 10-15 hour per week estimate that is given to the students. Of course, this table summarizes time that the logging system can

---

1OK, so I wasted a lot of time playing **hack**

2Monitoring activity that is not specific to CS50 would violate the terms of consent that the students agreed to when they agreed to participate in the study— and even if it didn’t, there are serious ethical considerations that would need to be addressed before this kind of activity could be passively monitored.
Figure B.1: Summary statistics for grades of participants in the CS50 1997 study.

<table>
<thead>
<tr>
<th>Assignment</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework 1</td>
<td>124</td>
<td>60.65</td>
<td>6.76</td>
<td>33</td>
<td>70</td>
</tr>
<tr>
<td>Homework 2</td>
<td>124</td>
<td>48.97</td>
<td>6.78</td>
<td>19</td>
<td>60</td>
</tr>
<tr>
<td>Homework 3</td>
<td>124</td>
<td>49.49</td>
<td>9.41</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>Homework 4</td>
<td>124</td>
<td>49.91</td>
<td>7.77</td>
<td>0</td>
<td>59</td>
</tr>
<tr>
<td>Homework 5</td>
<td>124</td>
<td>45.45</td>
<td>13.71</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>Homework 6</td>
<td>124</td>
<td>49.94</td>
<td>9.68</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>Homework 7</td>
<td>124</td>
<td>52.27</td>
<td>11.30</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>Homework 8</td>
<td>124</td>
<td>30.43</td>
<td>23.18</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>Homework 9</td>
<td>124</td>
<td>48.82</td>
<td>13.82</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>Final Project</td>
<td>124</td>
<td>18.70</td>
<td>3.92</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>Hourly 1</td>
<td>124</td>
<td>35.54</td>
<td>9.19</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>Hourly 2</td>
<td>124</td>
<td>33.83</td>
<td>5.72</td>
<td>19</td>
<td>45</td>
</tr>
<tr>
<td>Final Exam</td>
<td>124</td>
<td>136.60</td>
<td>18.73</td>
<td>74</td>
<td>176</td>
</tr>
</tbody>
</table>

measure—it may be the case that students are spending untold numbers of hours doing the reading for the course, or laboring offline on the design of their code.

B.2.3 Discussion of Distributions

The summary statistics given in figure B.5 are informative, but can be misleading because the nature of the distribution of time spent per assignment is too complex to express as a mean and standard deviation. Since students are given the opportunity to drop one assignment, nearly every assignment has one or more students who spend zero time on the assignment. In addition, since students can enroll or unenroll in the study at any point during the semester, I might have no record of any work done by a student for some assignments, even though they completed the assignment and I have records of their work for other assignments.

In addition, some assignments appear to spend extremely large amounts of time on a single assignment.

Figure B.6 shows stem-and-leaf diagrams of the raw distribution of time spent working on the computer by each student for assignments 4, 6, 7, and 9. The distribution of time spent on these assignments is reasonably typical of the other assignments, and show some consistency across at least these assignments.
Figure B.2: Histograms of hourly and final exam grades for students participating in the study.

<table>
<thead>
<tr>
<th>Midterm 1</th>
<th>Midterm 2</th>
<th>Final Exam</th>
</tr>
</thead>
<tbody>
<tr>
<td>50: Ix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46: Ixxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44: Ixxxxx</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>38: Ixxx</td>
<td>Ixxxxx</td>
<td></td>
</tr>
<tr>
<td>32: Ixxxxx</td>
<td>Ixxxxxxx</td>
<td></td>
</tr>
<tr>
<td>30: Ixxx</td>
<td>Ixxxxxxx</td>
<td>180:</td>
</tr>
<tr>
<td>28: Ixxxx</td>
<td>Ixxxx</td>
<td>170:</td>
</tr>
<tr>
<td>26: Ixx</td>
<td>xx</td>
<td>160: Ixxxx</td>
</tr>
<tr>
<td>24: Ix</td>
<td>x</td>
<td>150: Ixxxxxx</td>
</tr>
<tr>
<td>22: I</td>
<td>x</td>
<td>140: Ixxxxxxx</td>
</tr>
<tr>
<td>20: I</td>
<td>x</td>
<td>130: Ixxxxxxx x</td>
</tr>
<tr>
<td>18: I</td>
<td>x</td>
<td>110: Ixxxxxxx</td>
</tr>
<tr>
<td>16: I</td>
<td>x</td>
<td>100: Ixxx</td>
</tr>
<tr>
<td>14:</td>
<td>80:</td>
<td></td>
</tr>
<tr>
<td>12:</td>
<td>70: x</td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>60:</td>
<td></td>
</tr>
<tr>
<td>8:</td>
<td>50:</td>
<td></td>
</tr>
<tr>
<td>6:</td>
<td>40:</td>
<td></td>
</tr>
<tr>
<td>4:</td>
<td>30:</td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td>20:</td>
<td></td>
</tr>
<tr>
<td>0:</td>
<td>10:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:</td>
<td></td>
</tr>
</tbody>
</table>
Figure B.3: Histograms of assignment grades for assignments 2, 3, 4, and 5 for students participating in the study.

<table>
<thead>
<tr>
<th>Homework 2</th>
<th>Homework 3</th>
<th>Homework 4</th>
<th>Homework 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>60:</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>58:</td>
<td>XX</td>
<td>XXX</td>
<td>XXXXX</td>
</tr>
<tr>
<td>56:</td>
<td>XXXXX</td>
<td>XXXXXX</td>
<td>XXXXXXX</td>
</tr>
<tr>
<td>54:</td>
<td>XXXXX</td>
<td>XXXXXXXX</td>
<td>XXXXXXX</td>
</tr>
<tr>
<td>52:</td>
<td>XXXXXXX</td>
<td>XXXXXXXX</td>
<td>XXXXX</td>
</tr>
<tr>
<td>50:</td>
<td>XXXXX</td>
<td>XXXXXXXX</td>
<td>XXXXXXX</td>
</tr>
<tr>
<td>48:</td>
<td>XXXXXX</td>
<td>XXXXXXXX</td>
<td>XXXXXXX</td>
</tr>
<tr>
<td>46:</td>
<td>XXX</td>
<td>XXXX</td>
<td>XXXX</td>
</tr>
<tr>
<td>44:</td>
<td>XXX</td>
<td>xxx</td>
<td>XXXx</td>
</tr>
<tr>
<td>42:</td>
<td>XXX</td>
<td>XXX</td>
<td>xxx</td>
</tr>
<tr>
<td>40:</td>
<td>XXX</td>
<td>x</td>
<td>XXX</td>
</tr>
<tr>
<td>38:</td>
<td>x</td>
<td>xx</td>
<td>X</td>
</tr>
<tr>
<td>36:</td>
<td>X</td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>34:</td>
<td>XX</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>32:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>30:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>28:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>26:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>24:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>22:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>20:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>18:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>16:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>14:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>12:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>10:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>8:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>6:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>4:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>2:</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>0:</td>
<td>x</td>
<td>x</td>
<td>XX</td>
</tr>
</tbody>
</table>
Figure B.4: Histograms of assignment grades for assignments 6, 7, 8, and 9 for students participating in the study.

<table>
<thead>
<tr>
<th>Homework 6</th>
<th>Homework 7</th>
<th>Homework 8</th>
<th>Homework 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>60:</td>
<td>X</td>
<td>XXXx</td>
<td>XXXx</td>
</tr>
<tr>
<td>58:</td>
<td>XXX</td>
<td>XXXxxxxxxx</td>
<td>XXXxxxx</td>
</tr>
<tr>
<td>56:</td>
<td>XXXxxx</td>
<td>XXXxxxxxxx</td>
<td>XXXx</td>
</tr>
<tr>
<td>54:</td>
<td>XXXxxxxxxx</td>
<td>XXXxxxx</td>
<td>XX</td>
</tr>
<tr>
<td>52:</td>
<td>XXXxxxxxxx</td>
<td>XXXx</td>
<td>X</td>
</tr>
<tr>
<td>50:</td>
<td>XXXxxx</td>
<td>XXXx</td>
<td>XXX</td>
</tr>
<tr>
<td>48:</td>
<td>XXX</td>
<td>XX</td>
<td>X</td>
</tr>
<tr>
<td>46:</td>
<td>XXX</td>
<td>XX</td>
<td>X</td>
</tr>
<tr>
<td>44:</td>
<td>Xx</td>
<td>XX</td>
<td>Xx</td>
</tr>
<tr>
<td>42:</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>40:</td>
<td>Xx</td>
<td>Xx</td>
<td>Xx</td>
</tr>
<tr>
<td>38:</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>36:</td>
<td>x</td>
<td>Xx</td>
<td>Xx</td>
</tr>
<tr>
<td>34:</td>
<td>x</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>32:</td>
<td></td>
<td>x</td>
<td>Xx</td>
</tr>
<tr>
<td>30:</td>
<td>x</td>
<td>Xx</td>
<td>X</td>
</tr>
<tr>
<td>28:</td>
<td>x</td>
<td>Xx</td>
<td>X</td>
</tr>
<tr>
<td>26:</td>
<td>x</td>
<td>Xx</td>
<td>X</td>
</tr>
<tr>
<td>24:</td>
<td></td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>22:</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>20:</td>
<td></td>
<td>XXX</td>
<td>X</td>
</tr>
<tr>
<td>18:</td>
<td></td>
<td>XX</td>
<td>X</td>
</tr>
<tr>
<td>16:</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>14:</td>
<td>x</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>12:</td>
<td></td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>10:</td>
<td>x</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>8:</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>6:</td>
<td></td>
<td>Xx</td>
<td>X</td>
</tr>
<tr>
<td>4:</td>
<td></td>
<td>Xx</td>
<td>X</td>
</tr>
<tr>
<td>2:</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0:</td>
<td>X</td>
<td>XX</td>
<td>XXXxxxxxxx</td>
</tr>
</tbody>
</table>
Figure B.5: Summary statistics for time spent by 1997 CS50 students on each assignment.

<table>
<thead>
<tr>
<th>Assignment</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework 2</td>
<td>124</td>
<td>1.32</td>
<td>2.15</td>
<td>0</td>
<td>14.92</td>
</tr>
<tr>
<td>Homework 3</td>
<td>124</td>
<td>7.36</td>
<td>4.03</td>
<td>0</td>
<td>20.79</td>
</tr>
<tr>
<td>Homework 4</td>
<td>124</td>
<td>6.77</td>
<td>3.23</td>
<td>1.1</td>
<td>20.34</td>
</tr>
<tr>
<td>Homework 5</td>
<td>124</td>
<td>11.23</td>
<td>6.21</td>
<td>0</td>
<td>28.19</td>
</tr>
<tr>
<td>Homework 6</td>
<td>124</td>
<td>6.88</td>
<td>3.79</td>
<td>0</td>
<td>21.85</td>
</tr>
<tr>
<td>Homework 7</td>
<td>124</td>
<td>4.83</td>
<td>3.24</td>
<td>0</td>
<td>15.69</td>
</tr>
<tr>
<td>Homework 8</td>
<td>124</td>
<td>15.19</td>
<td>14.09</td>
<td>0</td>
<td>61.49</td>
</tr>
<tr>
<td>Homework 9</td>
<td>124</td>
<td>5.93</td>
<td>3.94</td>
<td>0</td>
<td>21.03</td>
</tr>
</tbody>
</table>

Figure B.6: Stem-and-leaf diagrams of the number of hours spent by each student on the computer for assignments 4, 6, 7, and 9.
Appendix C

Assignment Descriptions

This section provides a brief description of each assignment. For more information (and a copy of the assignments themselves), consult the CS50 assignment web page:

http://www.deas.harvard.edu/courses/cs50/assignments/assigns.html

C.1 Assignment 1

Assignment 1 is a “getting started” assignment. Students learn their way around the system, and write some algorithms. There is no real programming component, although students learn how to edit, compile and run a real program as part of the exercise.

C.2 Assignment 2

Assignment 2 is the first real programming exercise. It consists of three main pieces:

1. Fixing a piece of broken code.
   The students are given a specification and a program that does not meet the specification due to both syntactic and logical errors. The students must fix the code so that it meets the specification.

2. Writing a simple program to make a scalable “arrow” on the screen.
   The program must loop, asking the user for sizes, and then printing an “arrow” of the corresponding size on the screen. This exercise requires the use of loops and conditional control structures. This is the bulk of the assignment.


C.3 Assignment 3

Assignment 3 is the first challenging assignment for many of the students. It is considerably more time-consuming than earlier assignments.

The bulk of the assignment involves completing an implementation of a Hi-Q game. The students are given some scaffolding code, but most design and write the bulk of the Hi-Q code themselves. This exercise requires the use of arrays, and introduces functional decomposition.
C.4 Assignment 4

Assignment 4 involves randomness. The programming component consists of two exercises:

1. Building a Monte Carlo simulation to estimate the number of 3-character sequences that are valid English words (according to a function provided to the students, which consults the on-line dictionary).

2. Writing a program to simulate a casino slot machine. The machine must correctly simulate the probabilities of each payout, and keep track of how much money the user has won or lost.

Students are provided with some framework for the slot machine, and have already seen example code that implements a Monte Carlo simulation to estimate the value of π. My impression is that most students do not find this assignment particularly challenging, but it does reinforce their programming and design skills.

C.5 Assignment 5

Assignment 5 is a departure into data representation, machine architecture, and assembly language programming. Students write two short assembly language programs for the ANT\(^1\) architecture, and then implement most of an ANT virtual machine (some scaffolding is provided).

This assignment does not introduce many new C concepts, but does require a substantial amount of time to program. For the many students who decided to not attempt assignment 8, this was the longest and most difficult assignment of the course.

Good design is very important for this assignment-- a bad design can lead to a very long and complex (or buggy) implementation. Many of the students suffered because of this. A design review phase, similar to that used on assignment 7 (and nearly all of the CS51 assignments this year) would be very helpful for this assignment, because it would give the students an opportunity to revise their designs with feedback from a TF before investing large amounts of time coding a needlessly complex design.

C.6 Assignment 6

Assignment 6 introduces pointers, dynamic memory allocation, and linked lists.

Students implement a library of doubly-linked list routines, according to a set specification, and implement a program to simulate the Josephus problem. (Whether or not they use their linked list library to solve the Josephus problem is up to them, but it is encouraged-- the purpose of a library is to be useful!)

A lot of scaffolding code is provided for this assignment, and the students do not have to write very many lines of code. Many students still find this challenging, however, because they find pointers and dynamic memory allocation difficult topics to master.

C.7 Assignment 7

Assignment 7 reinforces dynamic memory allocation and binary trees. The assignment is to write a program that plays the game of animals. No scaffolding is provided, although a detailed algorithm is given in the assignment. Students are required to design their program before implementing it, and urged to submit their designs for review by their TF. Points are deducted for improperly modularized code.

\(^1\)http://www.eecs.harvard.edu/~ellard/ANT.
The amount of code required for this assignment is greater than assignment 6, but many of the students find this assignment relatively easy. It is not unheard of for a student to type in their code directly from the design and have it working almost immediately. These students are given a superb lesson in the importance of design discipline.

C.8 Assignment 8

Assignment 8 is the largest programming assignment of the year (and this year, perhaps the largest in the history of the course). The assignment is to write an assembler for the ANT architecture.

This assignment was unreasonably difficult for many students. The scope of the assignment was too broad, and the topic not explained in enough detail. The most crucial issues are understanding what the scaffolding code does, coming up with a proper design, and understanding what each component must do. Many of the designs I saw were needlessly complex or contained many redundant modules (i.e. separate functions to handle each opcode, each with its own subfunctions to parse the arguments of that opcode, etc). Some of these students wrote an incredible amount of code for this assignment, and spent days working on it.

I believe that the difficulty of this assignment could be reduced by having a design review, similar to assignment 7, and spending more time in lecture talking about the specific algorithms used.\(^2\) At best, however, this problem requires approximately 1,000 lines of code to solve-- a very large program for students at this level to write in a week or two.

Because of the difficulty of assignment 8, its due date was delayed twice, until it was eventually due 5 days later than it would have ordinarily been due.

C.9 Assignment 9

Assignment 9 explores state-space search. The students write a program that allows the computer to determine whether a given H-Q configuration has a solution-- and if so, what moves lead to it. The students are not provided with an algorithm, but use of recursion is broadly suggested.

The amount of code required for assignment 9 is not large-- perhaps somewhere between the amount of code required for assignments 6 and 7. However, the algorithm requires some insight to solve. Students who devise a correct algorithm and understand it clearly can finish the coding for this assignment relatively quickly, while students who do not see the trick and try to hack something together do not make much progress.

The grading for this assignment was done on a relatively easy scale (which helps to explain why the mode score for this assignment was a perfect score), and the written exercises for this assignment were worth half of the points. In addition, the due date was pushed back (to leave time for students to work on it after assignment 8), and unfortunately for many students this meant that it was now due after they intended to leave for Winter Break! I believe that this hard deadline, coupled with the fact that students were exhausted (and out of late days) after assignment 8, contributed to the number of students who did relatively badly on this assignment.

As noted in chapter 3, the relationship between score and time spent and the other variables was different for this assignment than the others-- this distribution of grades for this assignment is difficult to interpret.

\(^2\)One source of confusion in this assignment was that there are several valid ways to solve some of problems, such as backpatching-- and different TEs apparently taught different methods. This meant that students who did not have a firm understanding of the algorithm their TE suggested could be completely baffled by contradictory advice from their friends or other TEs. It is very important for all of the TEs to agree on one approach-- not to claim that this is the “right” or “best” approach, but simply to avoid this sort of confusion.
C.10 The Final Project

The final project is chosen by the student, with guidance from his or her TF. Students are given considerable latitude in their choice of project--as long as it is related to what they are learning in CS50 and are neither too trivial nor too large ambitious. The suggested amount of work is approximately equal to 2 ordinary assignments.

Because the choice of projects is up to each student, I did not attempt to analyze the data from the projects. It would be interesting, however, to see if there are any patterns to the kinds of projects that students choose, or whether there is a relationship between how long students spend on their projects and how long they spend on their other assignments, or what their exam scores are.
Appendix D

Lessons Learned

Although I didn't find answers to as many of my research questions as I would have liked, I did learn some useful things. Since some of these may be useful to future researchers in this area, I have written some of them down.

• Get section information.

    Although this may weaken the degree of anonymity, it would be very useful to know exactly when a student's assignments are due. It would also be interesting to know the set of students in each section, to investigate whether there are significant differences between sections in grades or student work habits.

• Don’t rely on dot files, because people change them.

    The enrollment of a student in the study is represented by an environment variable that is set in the user’s .cshrc. Unfortunately, a standard panacea for many problems on the FAS systems is to copy over a new .cshrc. Thus, students trying to solve problems in their environment often inadvertently unenrolled themselves from the study.

    The difficulty is finding something that is visible to the student’s environment but invisible from the outside. Creating a stub dot file in the the user’s home directory would probably be a better solution, although its permissions would have to be set carefully in order to protect anonymity.

• Better results could be obtained with small changes to the course.

    This is not meant as a criticism of the amount of cooperation and help I received from Professors Seltzer and Chen–They cooperated in every appropriate manner. The problem was that I didn’t know what to ask of them. Here are some ways that the course could have been made easier to monitor, without having an impact on the students or teaching staff:

    – Mandate the names of the directories used for each assignment, and make sure that they are unique. This makes it easier for the logger to figure out which assignment students are working on.
    – Collect more information about student backgrounds as part of the survey, such as whether they have a computer in their room, or whether they have any roommates who have taken or are taking CS50.\(^1\)

\(^1\) Much of this information is actually available, but whether this information may be gathered surreptitiously is an ethical question.
• Use the right tools for the right job.

Finding the right tools to analyze the data was a very time-consuming task, and I won’t claim to have gotten it right. Many of the tools that are common and familiar, such as Microsoft Excel or Microsoft Access, are not capable of handling the large number of records generated by a study of this size, as I learned the hard way.

In the end, I used SAS on the mainframe at the Graduate School of Education to perform all of my statistical analysis of this data. This SAS installation is slow and clunky (and text-only, so I could not produce the flashy charts and graphs that would ordinarily appear in a paper of this kind), but SAS is a very useful and powerful tool.

To perform any nontrivial statistical analysis of a data set of this size and complexity requires a good statistical package, and these packages take time to learn, so budget plenty of time for this.
Bibliography


[Par] Allen Parrish. Syntax error reduction patterns among student programmers. An unpublished article, intended for eventual publication on SIGCSE or a similar journal. Draft provided via personal communication 4/6/98. This is a followup to [PLCM97].


