Gender Bias in Programing Code Reviews

Karin Brodd, Steve Yang, Luke Doolittle, Kari Ross
W241: Experiments and Causality
UC Berkeley School of Information
{kbrodd, luke.doolittle, steve.yang, kari.ross}@berkeley.edu
April 24th, 2018

1 - ABSTRACT

A survey was conducted in an attempt to determine if gender bias exists in the opinion of code written by software developers. The survey design varied the perceived gender of the author of a piece of code along with the quality of that code and then asked the participants a number of questions about qualities of that code. This data was then analyzed along with information about the demographics of the survey participants. The survey results did not provide evidence for the hypothesis that womens code is judged more harshly than mens. However the survey did show some heterogeneous effects of gender between code of differing quality: women tended to be judged less harshly on code of poorer quality, but that difference did not exist in good quality code.

2 - INTRODUCTION

Wage discrimination lawsuits [1]; a manifesto arguing psychological and biological differences to be less able to be skilled engineers [2]; hard data showing one of the “best places to work” award employer has gender bias in their pay structure [3] . There is a continued discussion on the bias and inequality in the tech and engineering industry, giving rise to the need for a continued look at where the bias lies, in order to find solutions that work, rather than just make continued talking points.

2.1 Why an Experiment

The ability to manipulate a single variable, gender, without any of the other confounding variables, was a driving factor in the design of this study. What would happen if in a biography, the pronoun “he” was replaced with “she”? Could this small change be the cause of a different rating or review of code?

2.2 Hypothesis

The hypothesis under investigation is that gender bias exists in code reviews. Specifically that code written by a female will be rated less favorably than code written by a male. We also hypothesize that there is interaction between code quality and gender for the

rating of code reviews.

3 EXPERIMENTAL DESIGN

In order to test the hypothesis of gender bias in code reviews, a multifactor 2x2 experimental design was created, with a high quality / low quality code treatment and a female / male biography treatment associated with the code. After receiving the treatment, the subject was then asked to review the code and rate it on four different common software metrics.

3.1 Treatment Assignment

The multifactor 2x2 experimental design is shown below in Figure 1.

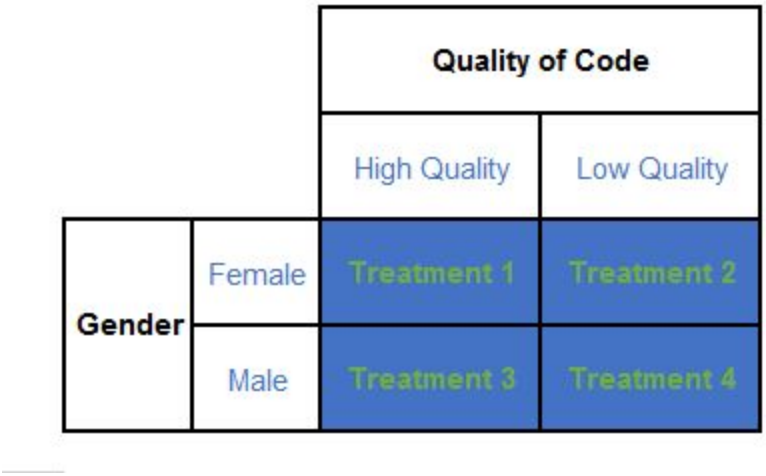


Figure 1: 2x2 Treatment Assignment

The methodology for creating the high quality versus low quality coding examples was based on software engineering principles of:

- Readability
  - how easily is the code understood by someone who didn't author the code
  - Do variables make intuitive sense
- Maintainability
  - Is the code structured such that it can be altered if requirements change
- Optimization / efficiency
  - bigO notation
  - Brute force implementation versus an optimization
- Correctness
  - does the code perform the task specified in the requirements

The premise of the experiment was that experimental subjects were presented the code to review and the requirements of the code. The code's approachability was around an average level of programming: it wasn't trivial but not overly complex. The requirements involved a simple merge and sorting of two arrays. The actual code used for treatment is shown in Appendix A of this paper.

The gender treatment was created by writing fictional biographies for the male and female software engineers. The biography included personal pronouns referring to the engineer, the company they were employed by, and the school they attended. The only change between the two gender treatments was the use of "he" or "she" pronouns in the biography. Names were not used, in order to reduce the possibility of a non-native English speaker confusing the gender of names. The biography used for treatment is shown in Appendix B.

3.2 Experiment Subjects

The experimental subjects were self-selected from Mechanical Turk with the task of doing a code review. The test subjects were not necessarily professional software engineers but there was the requirement of programming and specifically python experience. Compensation was $1 per HIT. There was no geographical location restriction on the experimental subjects: many large

In order to assure that the experiment subjects were from a population of coders who would be qualified to do code reviews, a survey of 5 questions was formulated to determine a base level of programming competence. The survey questions are listed in Appendix C.

3.3 Outcome Measures

Subjects are asked to rate the code on a 5 point scale (presented as a Likert scale) on 4 different criteria related to the overall quality of the code, which was the basis of the code creation.

- Correctness: The code meets the requirements that were specified.
- Efficiency: The code is written in the most efficient way possible.
- Readability: The code is easy to read and understand.
- Maintainability: The code would be easy to make modifications to if needed.

3.4 Manipulation Survey Questions

Subjects were given a post treatment survey in order to determine if subjects noticed and retained the gender of the code writer, as this was the main treatment we examined. The survey was three multiple choice questions asking about information contained in the treatment biography asking the gender of the engineer, what school they attended, and who they worked for. The results are discussed in Section 5.1.3 Manipulation Checks.

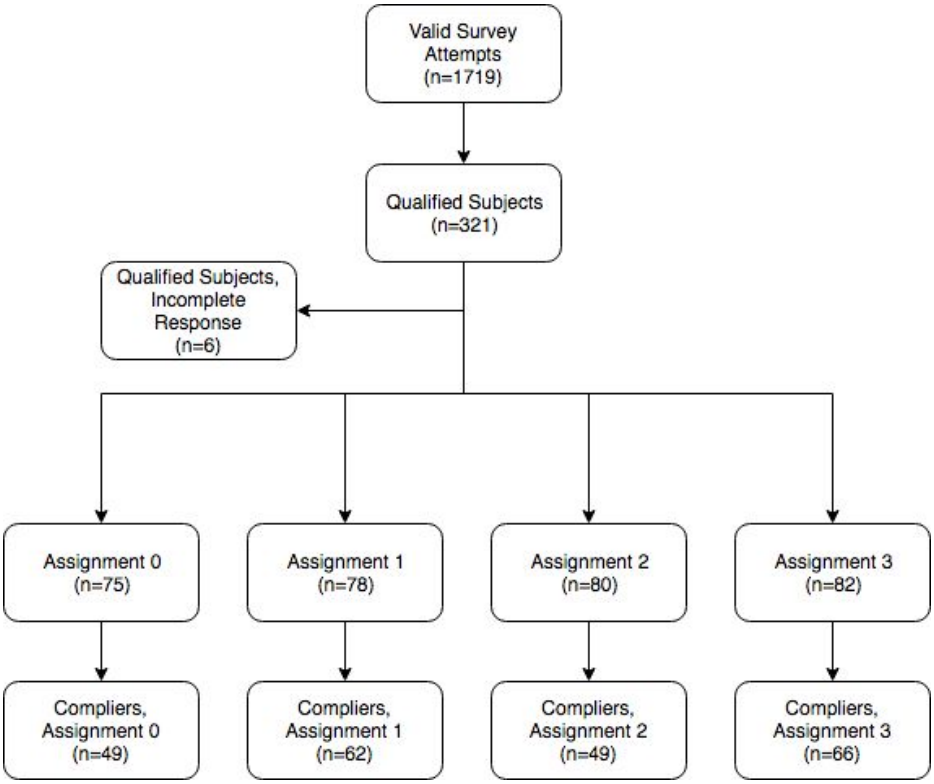
3.5 Randomization Design

The randomization of assignment occurs through Qualtrics but we did cursory check by performing an F-test on a set of covariates. We asked 3 demographic questions of the subject (work experience, gender, formal education) and several compliance questions about the conditions present in the assignment for the survey (the coders gender, employer, and education institute). The F-test revealed that all of the covariates did not display a statistically significant difference with the exception of the gender compliance question. In that question the treatment assignments that had a male developer listed as the code author had a higher rate of compliance (~80%) than the treatment assignments that had a female listed as the author (~50%).

4- Data Acquisition

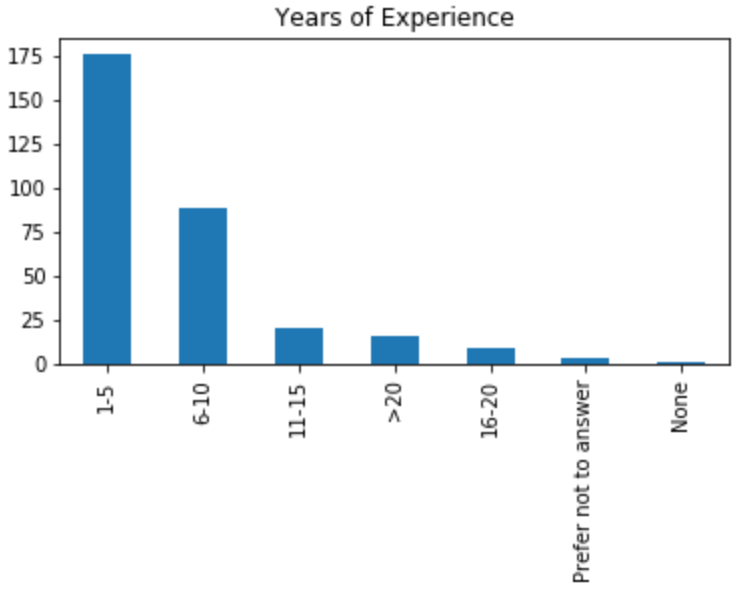
4.1 Survey Flow

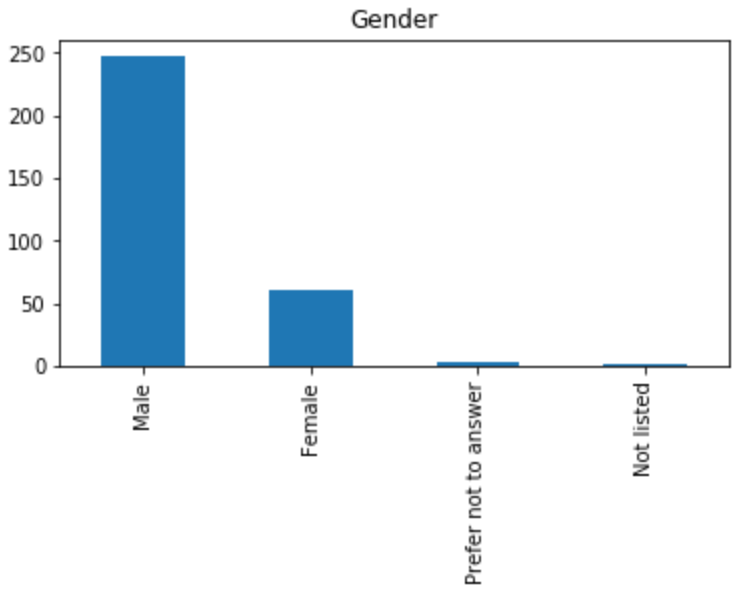
Below is a diagram showing the participant count in each step of the survey. We began with 1719 participants of which 321 correctly answered the qualifying questions. Of those qualified subjects 6 did not finish the survey and a number were assigned to each treatment condition. Of those in each treatment condition a certain number passed the qualifying questions.

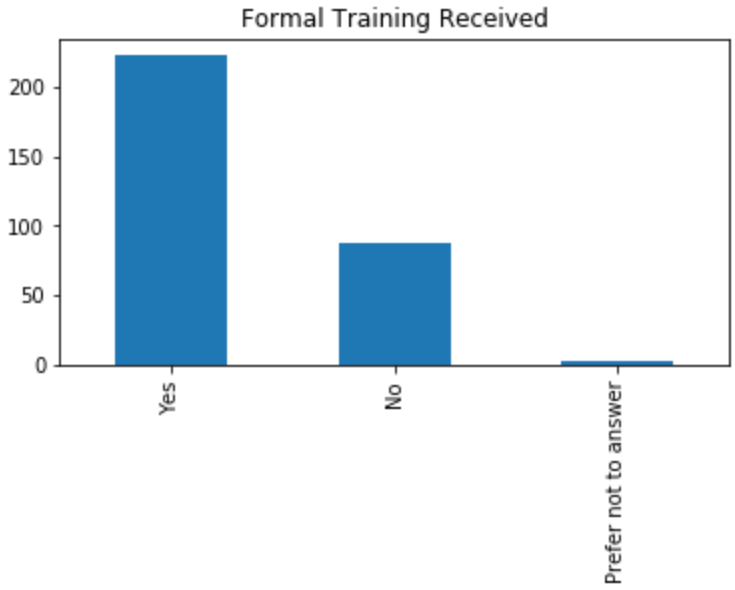


4.2 Distribution of Demographics

Below are figures showing the distribution of the answers to the demographic questions.







4.3 Randomization Check

As was mentioned in section 3.4 we performed a randomization check via an F-Test against the 3 demographic questions and the 3 qualifying questions. The results of the F-Test are present in the table below.

Covariate	**P-Value**
Years of Experience (1-5)	0.93
Years of Experience (6-10)	0.92

Years of Experience (11-15)	0.56
Years of Experience (16-20)	0.26
Years of Experience (>20)	0.67
No Formal Education	0.33
Has Formal Education	0.24
Gender Female	0.33
Gender Male	0.79
Gender Not Listed	0.39
Compliance Question 1	0.68
Compliance Question 2	0.01
Compliance Question 3	0.61

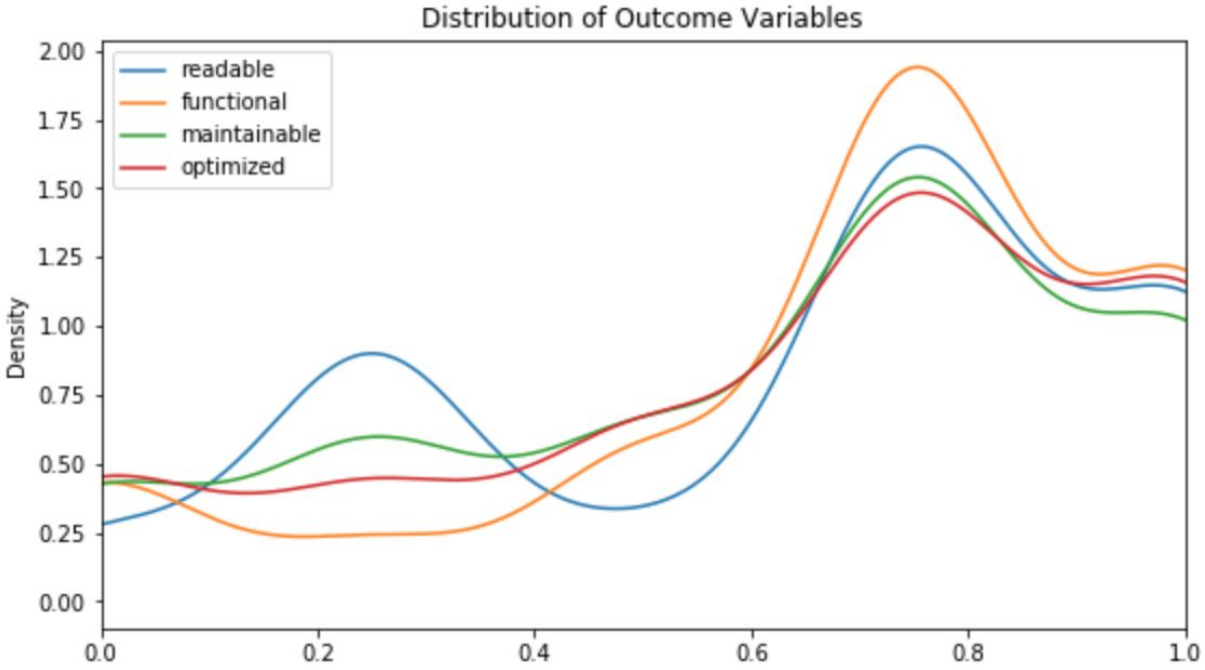
5 Analysis

5.1 Descriptive Statistics

5.1.1 PCA

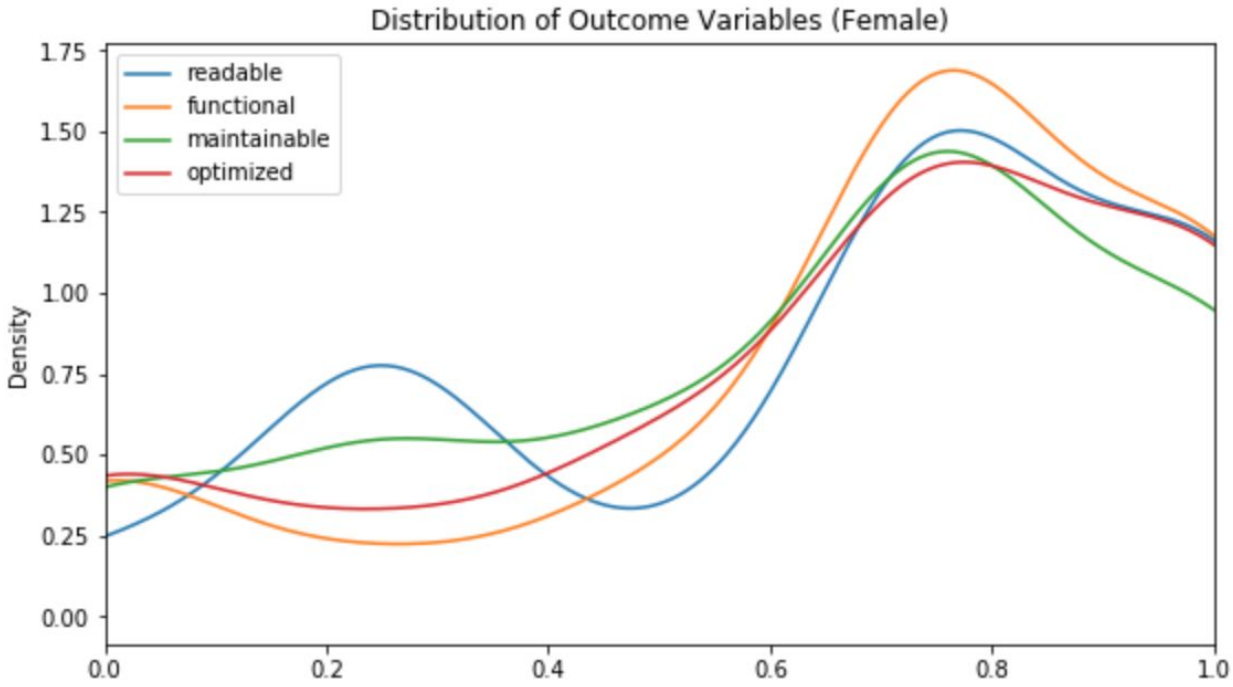
Since we collected data on multiple outcome measures, we performed PCA to determine if we can reduce the dimension of the outcome into a single component. After analyzing the data, for two main reasons we decided against using PCA for dimensionality reduction. First, PCA makes interpretability difficult. That is, it's hard to quantify the magnitude of change caused by the treatment. Another reason is that, unsurprisingly, the amount of variance that is captured by the first component is relatively low. One reason for this is that not all features of judging code coincide. In other words, code that subject's judge as very readable, do not necessarily also need to be rated as very optimized, for example. The total amount of variance captured by a single component was less than half of the total variance in the four dimensions.

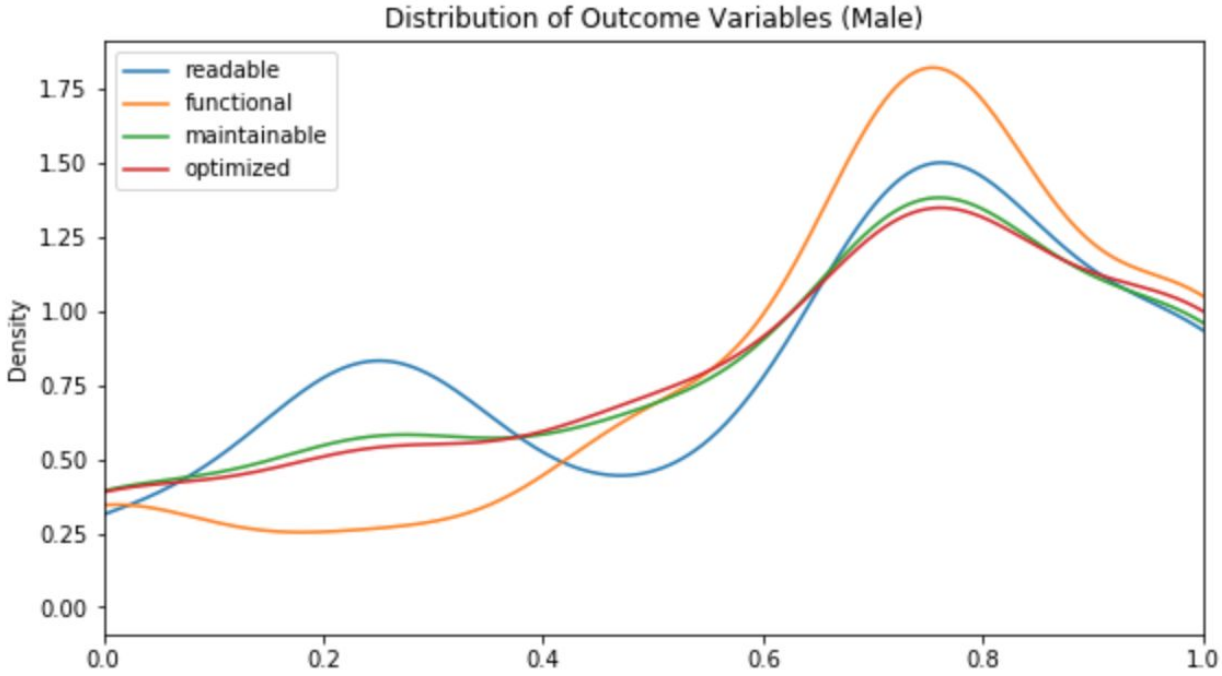
5.1.2 Distribution of Outcome



While PCA showed that not all outcome measures move together, we do see from the figure above that all four measures share a similar distribution. We also see that the distribution is bimodal in nature with peaks at the values that coincide with “Disagree” and “Agree”.

Next, we examined the distribution by the gender assignments to see if that was the cause of the two peaks.





Even with the separation of gender assignment, we see the same bimodal distribution. It seems likely that the peaks may just be a feature of the way subjects tend to answer on the Likert scale.

5.1.3 Manipulation Checks

We asked respondents to recall attributes about the code author after answering the primary outcome survey questions. This served as a manipulation check to assess the effectiveness of stimulus. Although we were only interested in the respondents' recollection of gender, we also asked about the other attributes that were planted in the stimulus. The average rate of recollection for gender, company, and university was 0.72, 0.80, and 0.61, respectively. Keep in mind, we actually varied stimulation for the first attribute, while the second two remained static.

Next, we performed a T-Test on each of the attributes by gender assignment. We fail to reject the null for *company* and *university.* However, we noticed an interesting difference in manipulation effectiveness with gender recollection. While 80% of respondents who were assigned to *male* were able to correctly recall the code author's purported gender, only 63% of respondents who were assigned to *female* were able to correctly recall the gender. This shows that the stimulus was not as effective as it initially appeared. Instead, it seems apparent that the default guess for many of the subjects when he or she could not recall was to answer *male*. For this reason, we decided to estimate the treatment effect by using an *Intent-to-Treat* estimator.

5.2 Regression Analysis

We opted to use an *Intent-to-Treat* estimator to measure out treatment effect. There are several benefits of an ITT analysis. One such benefit is the ability to produce unbiased estimators even while ignoring the issue of noncompliance. We start off with a "baseline" regression in which we regressed each outcome measure independently on assigned gender and code quality along with other covariates to increase precision. Then, we ran the same regression, while adding an interaction term for gender and code quality.

5.2.1 Baseline Regression

We specified the following model for each outcome measure.

$$Y_i = \alpha + \beta_1 \text{gender}_i + \beta_2 \text{quality}_i + \gamma_i + u_i$$

- α: intercept
- β_1: The effect of being assigned to female
- β_2: The effect of being assigned to good code
- γ: Demographic Fixed Effects
- gender_i: Female = 0; Male = 1
- quality_i: Good Code = 1; Bad Code = 0

Below we show the results from the regression and calculated heteroskedasticity-robust confidence intervals -- displayed below in parentheses.

Table 1:

	Dependent variable:			
	readable	functional	maintainable	optimized
	(1)	(2)	(3)	(4)
Assignment - Gender	0.048	0.010	0.010	0.024
	(−0.018, 0.113)	(−0.058, 0.078)	(−0.059, 0.080)	(−0.049, 0.097)
Assignment - Code Quality	0.250***	0.101***	0.196***	0.039
	(0.186, 0.314)	(0.034, 0.169)	(0.127, 0.265)	(−0.034, 0.113)
Constant	0.732***	0.807***	0.585***	0.851***
	(0.394, 1.070)	(0.458, 1.156)	(0.232, 0.939)	(0.476, 1.225)
Demographic Fixed Effects	Yes	Yes	Yes	Yes
Observations	307	307	307	307
R^2	0.190	0.039	0.105	0.028
Adjusted R^2	0.177	0.023	0.090	0.012
Residual Std. Error (df = 301)	0.284	0.300	0.306	0.322
F Statistic (df = 5; 301)	14.159***	2.437**	7.048***	1.764
Note:				*p<0.1; **p<0.05; ***p<0.01

It is apparent that while we see various coefficients with statistical significance across different measures, *readable* appears to be the most useful. This seems reasonable, since both code samples that we displayed were functional, but had different degrees of readability.

Also, to reiterate the magnitude of these estimates, we note that the dependent variable is a value between 0 and 1, corresponding to a five-point Likert scale. Thus, an increase of 0.25 is approximately the difference between "Agree" and "Strongly Agree", for example.

The following interpretations were made based on the *readable* outcome measure. We saw statistically significant evidence that those who were shown the good quality code were more likely to grade the code more favorably. These particular results are somewhat trivial, but it serves as sort of a manipulation check. Also, the gender of the survey respondent had a statistically significant effect on outcome measures. Note this is not displayed in the findings above, but instead rolled into the row labeled “Demographic Fixed Effects”.

Finally, we did not see any statistical evidence of our primary treatment of interest. This is most likely due to the fact that there are heterogeneous treatment effects, which we show evidence of in the next section.

5.2.2 Interaction of Gender and Code Quality

Next, we ran a regression with the following model specification and again calculated heteroskedasticity-robust standard errors.

$$Y_i = \alpha + \beta_1 \text{gender}_i + \beta_2 \text{quality}_i + \beta_3 (\text{gender}_i \dot{} \text{quality}_i) + \gamma_i + u_i$$

- α: intercept
- β_1: The effect of being assigned to female
- β_2: The effect of being assigned to good code
- β_3: The effect of interacting female and good code
- γ: Other covariates
- gender_i: Female = 0; Male = 1
- quality_i: Good Code = 1; Bad Code = 0

Table 2:

	Dependent variable:			
	readable	functional	maintainable	optimized
	(1)	(2)	(3)	(4)
Assignment - Gender	0.114**	−0.017	0.071	0.004
	(0.013, 0.214)	(−0.119, 0.084)	(−0.032, 0.174)	(−0.099, 0.107)
Assignment - Code Quality	0.317***	0.074	0.257***	0.019
	(0.230, 0.403)	(−0.019, 0.166)	(0.163, 0.351)	(−0.080, 0.119)
Interaction - Gender*Quality	−0.135**	0.056	−0.123*	0.040
	(−0.261, −0.008)	(−0.080, 0.192)	(−0.260, 0.014)	(−0.106, 0.186)
Constant	0.705***	0.819***	0.560***	0.859***
	(0.368, 1.041)	(0.469, 1.168)	(0.207, 0.914)	(0.483, 1.235)
Demographic Fixed Effects	Yes	Yes	Yes	Yes
Observations	307	307	307	307
R^2	0.202	0.041	0.114	0.029
Adjusted R^2	0.186	0.022	0.096	0.010
Residual Std. Error (df = 300)	0.282	0.300	0.305	0.322
F Statistic (df = 6; 300)	12.655***	2.140**	6.436***	1.517
Note:				*p<0.1; **p<0.05; ***p<0.01

We see that *readable* and *maintainable* have statistically significant treatment effects. There are also clear signs of heterogeneous treatment effects.

For *readable,* the first thing we note is the change in direction of the estimated treatment effect depending on the quality of code that is shown. On average, we saw that when subjects

are judging lower quality code, purportedly female-authored code receives a more favorable score. However, this effect disappears with higher quality code; that is, there was no statistical difference between purportedly male-authored or female-authored code. While the magnitude of each coefficient are 0.11 and -0.14, respectively, the two must be combined to see the causal effect. The sum of the two is approximately -0.02, but based on the standard errors, we don't even need a formal test to determine that this value is statistically insignificant.

We also noticed a similar trend for *maintainable* in the interaction term, but do not see signs of significance when we present poor code quality. We do not see evidence of differences between treatments in the other two outcome measures.

6 Conclusion

The results did not support the hypothesis that the code authored by a woman would be rated less favorably than code authored by a man. If anything these results suggested that with code of poor readability, women may be judged less harshly than men. The results did however suggest a heterogeneous effect of gender between code of good quality and poor quality, where the advantage women have when poor code was being reviewed disappeared when the code was of good readability. One possible interpretation of this result may be that the reviewers perceived the female coder to be less threatening, or felt a need to protect her from harsh criticism, when the code was of poor quality. This could have led to the reviewer to be less critical of the poor quality code on average when the coder was female, but not when the code was of good quality.

The results of this experiments had a few weaknesses. Firstly, the rate at which subjects could recall the gender of the coder showed that a fairly low percentage of subjects were either paying close attention to the bio or picking up on the language cues, and thus were not receiving the intended treatments. Additionally, the difference in the rate at which subjects could correctly recall the gender of the coder (male coders were correctly recalled at a higher rate than female coders) suggests that subjects who did not recall the gender were just guessing (or assuming) male by default. This fact likely shrunk the size of the intent to treat effect, as a large number of our subjects were treating the code as being written by a male coder or some coder of unknown gender regardless of their assigned treatment group.

Because MTurk does not provide the ability to target workers who had the particular Python skills that were desired, it was necessary to rely on a set of qualification questions in the survey to filter subjects. However, because compensation was based on completion on the survey (workers were not paid if they did not pass qualification), some workers may have been motivated to get past the qualification questions even if they didn't possess any coding skills. This could have been accomplished either by research, or making multiple attempts at the survey. The end result was that some of our results were from under qualified workers who would be simply adding extra noise to our data rather than adding any meaningful results.

6.1 Future Work

While the outcome measures looked at the subjects' perceptions of the code's readability, maintainability, correctness, and efficiency, the code quality treatments only truly manipulated the code's readability (and to some extent it's maintainability as a side effect of the readability). A future experiment that might be attempted could be to come up with multiple code quality treatments that sought to manipulate the other outcome measures. This experiment would potentially provide the ability to observe the interaction between the quality of the code and gender on multiple dimensions. Additionally, the questions regarding the efficiency of the code (in terms of bigO notation) was an advanced difference in code and might not be suitable for this type of experiment.

Another idea for an subsequent experiment would be one which made the treatment of male versus female code more evident to the subject. The goal of such an experiment would be to increase the number of subjects who we successfully manipulated into believing that they reviewing the work of a male or female coder. As an example of a possible experimental design, a portrait of the purported code author might be presented to the subject. If this approach were to be used, care would need to be taken to make sure that new biases were not introduced based on the appearance of the individual in the portrait. For example, whether or not the individual in the portrait was smiling, or wearing glasses could affect the subjects perception. To address this issue, multiple portraits of individuals over a wide variety of physical attributes for each gender could be used and assigned at random within the appropriate treatment.

7 Appendices

Appendix A: Software Engineer Treatment

Below is the biography used to manipulate the gender in the treatment. Note, there are 3 instances of the male / female pronoun he/she.

> "The author of this code is currently working as a Software Engineer. She has 10 years of experience, and is currently working at Lyft for the past 2 years. She obtained a degree in Computer Science from Rice University prior to entering the industry and graduated with a GPA of 3.6/4.0.
>
> She is tasked with the following requirements.
>
> Write a function that returns …"

Appendix B: Code Treatment

The requirements that went with the code review were that:

The code would take in 2 sorted arrays and would return the intersection of the two arrays, in a single, sorted array.

The statement of the array's being sorted was important because it allowed for the high-quality of code to execute much faster than low-quality code. The execution time optimization could be considered a more advanced treatment in the code, but it was deemed necessary in order to have the code treatment be more sophisticated than what a compile would catch for syntax errors.

The high quality code also followed consistent naming notation, as well as coding standards for spacing and comments. The low-quality code was purposely written with non-descriptive names which would also be hard to do text-editor searches (it's difficult to search for the variable name "X"). These are common blunders which will production level code to be rejected or reviewed poorly.

High Quality Code Treatment

```
#=========================================================================
# Function Name:    intersectionTwoSortedArrays(A,B)
# Function Inputs:  Two Sorted Arrays, A and B
# Function Purpose: Returns an array which contains the intersection
#                       of Arrays A and B. The returned array is sorted
# Error Handling:   Function will throw an assertion
#                        if both arrays are not sorted in ascending order
#                   no error handling for data types
# Run Time Approx:  O(len(A) + len(B))
#                   This is reduced from a brute force of O(len(A)*len(B)
#                   by exploiting the nature of sorted lists
# Author:
#=========================================================================
def intersectionTwoSortedArrays(A,B):
    """ Returns the intersection of array A and array B.
        If there is no intersection of the arrays, returns an empty array"
        Requires arrays A and B to be sorted in ascending order.
    """
    # Basic Error handling of Inputs
    assert A == sorted(A), "The first array was not sorted"
    assert B == sorted(B), "The second array was not sorted"

    # Initalization of array of intersection elements and
    # Initalization of array indices
    intersectionAB = []
    i, j = 0, 0

    # While loop through shortest list
    while i < len(A) and j < len(B):
        # if/elif/else statement exploits the nature of the sorted list
        # by advancing through input arrays simultaneously
        # Reduces the time complexity from brute force of O(mn) to O(m+n)
        if A[i] == B[j]:
            if i == 0 or A[i] != A[i - 1]:
                intersectionAB.append(A[i])
                i += 1
                j += 1
        elif A[i] < B[j]:
            i += 1
        else: # A[i] < B[j] so advance index for Array B
            j +=1

    return intersectionAB
```

Low Quality Treatment

```
def codingProblem(X, XX):
    return[a for i, a in enumerate(X) if (i == 0 or a!= X[i-1]) and a in XX]
```

Appendix C - Programming Skill Quiz

The base level of programming required to be included in the survey was assessed by the programming quiz shown below.

Question 1: Indicate the type of expression below

3.14

- ○ int
- ○ float
- ○ bool

Question 2: Indicate the type of expression below

True

- ○ int
- ○ float
- ○ bool

Question 3: Select the Output from a program

Below is a Short Python Program

```
num = 5
if num > 2:
    print num
    num -= 1
print num
```

Select what prints out from running the program

- [ ] 5, 4, 3, 2
- [ ] 5, 4
- [ ] 5, 4,3, 2, 1

Question 4: Below is a code for a python dictionary

```
animals = { 'd' :  ['donkey'], 'e': ['elephant'], 'f': ['frog']}

animals['d'].append('dog')
animals['g'] = ['giraffe']
```

How many keys are in the dictionary

- ○ 2
- ○ 3
- ○ 4

Question 5: Return from a Python Dictionary

```
animals = { 'd' :  ['donkey'], 'e': ['elephant'], 'f': ['frog']}

animals['d'].append('dog')
animals['g'] = ['giraffe']
```

What is returned from the expression

```
animals['d'][1]
```

- ○ giraffe
- ○ donkey
- ○ dog

References

1. Greenway, R. (2017, July 26) Gender Discrimination Lawsuits at Silicon Valley Tech Companies. Retrieved from https://www.nbcbayarea.com/news/local/Gender-Discrimination-Lawsuits-in-Silicon-Valley--436347823.html on 04/22/2018
2. Molteni, M. and Rodgers, A. (2017, August 15) The Actual Science of James Damore's Google Memo. Retrieved from https://www.wired.com/story/the-pernicious-science-of-james-damores-google-memo/ on 04/20/2018
3. Cao, S. (2018, April 16) Salesforce CEO Marc Benioff Discusses the Unexplained Gender Pay Gap in Tech. Retrieved from http://observer.com/2018/04/salesforce-ceo-marc-benioff-unexplained-gender-pay-gap/ on 04/23/2018