ORDINARY LEAST SQUARES REGRESSION OF ORDERED CATEGORICAL DATA: INFERENTIAL IMPLICATIONS FOR PRACTICE

by

BETH R. LARRABEE

B.S., Kansas State University, 2009

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Statistics
College of Arts and Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2011

Approved by:

Major Professor
Dr. Nora Bello

Abstract

Ordered categorical responses are frequently encountered in many disciplines. Examples of interest in agriculture include quality assessments, such as for soil or food products, and evaluation of lesion severity, such as teat ends status in dairy cattle. Ordered categorical responses are characterized by multiple categories or levels recorded on a ranked scale that, while apprising relative order, are not informative of magnitude of or proportionality between levels. A number of statistically sound models for ordered categorical responses have been proposed, such as logistic regression and probit models, but these are commonly underutilized in practice. Instead, the ordinary least squares linear regression model is often employed with ordered categorical responses despite violation of basic model assumptions. In this study, the inferential implications of this approach are investigated using a simulation study that evaluates robustness based on realized Type I error rate and statistical power. The design of the simulation study is motivated by applied research cases reported in the literature. A variety of plausible scenarios were considered for simulation, including various shapes of the frequency distribution and different number of categories of the ordered categorical response. Using a real dataset on frequency of antimicrobial use in feedlots, I demonstrate the inferential performance of ordinary least squares linear regression on ordered categorical responses relative to a probit model.

Table of Contents

List of Figures v
List of Tables xi
Acknowledgements xii
Dedication xiv
Chapter 1 - Literature Review 1
1.1 Introduction 1
1.2 Ordered Categorical Responses 2
1.3 Statistically Sound Methods for Ordered Categorical Data 4
1.3.1 Probit 4
1.3.2 Proportional Odds Logistic 5
1.3.3 Multinomial Model 7
1.4 Interval Methods of Practical Use 8
1.4.1 T-test 9
1.4.2 ANOVA 10
1.4.3 Ordinary Least Squares Linear Regression 12
1.5 Summary 13
Chapter 2 - Ordinary Least Squares Regression of Ordered Categorical Data 14
2.1 Introduction 14
2.2 Data Simulation Methods 16
2.3 Analysis 24
2.4 Results 25
2.4.1 Type I Error 25
2.4.2 Statistical Power 28
2.4.3 Inferences on Slope 36
2.5 Case Study Based on Real Data 39
2.6 Discussion 41
2.7 Future Directions 43
2.8 Summary 45

2.9 Final Remarks 45
References 47
Appendix A - Appendix A: Empirical Distributions of Slope 50
Appendix B - Appendix B: Simulation Code for $\beta^* = 0$ Condition 66
Appendix C - Appendix C: Simulation Code for $\beta^* = 1$ Condition 116

List of Figures

Figure 2-1 Realization of a simulated ordered categorical response with 7 levels and a Uniformly-Shaped frequency distribution. Histogram is based on a single simulated Monte Carlo replication with 300 observations. 20

Figure 2-2Realization of a simulated ordered categorical response with 7 levels and a Bell-Shaped frequency distribution. Histogram is based on a single simulated Monte Carlo replication with 300 observations.....21

Figure 2-3 Realization of a simulated ordered categorical response with 7 levels and a Triangularly-Shaped frequency distribution. Histogram is based on a single simulated Monte Carlo replication with 300 observations. 22

Figure 2-4 Realization of a simulated ordered categorical response with 7 levels and a Exponentially-Shaped frequency distribution. Histogram is based on a single simulated Monte Carlo replication with 300 observations. 23

Figure 2-5 Empirical Type I error for inference on H_0) $\beta = 0$ based on ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels and Uniform, Bell, Triangular, or Exponentially shaped frequency distributions. Each scenario is represented by 4000 Monte Carlo replications. The figure represents settings where the explanatory covariate consisted of 51 levels ($x_i^{(51)}$) ranging from -50 to 50 in intervals of 2. Bounds on $\boldsymbol{\alpha}$ correspond to 2.5th and 97.5th percentiles of a binomial distribution with probability of 5% and size = 4000, expressed as a proportion.26

Figure 2-6 Empirical Type I error for inference on H_0) $\beta = 0$ based on ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels and Uniform, Bell, Triangular, or Exponentially shaped frequency distributions. Each scenario is represented by 4000 Monte Carlo replications. The figure represents settings where the explanatory covariate consisted of 5 levels ($x_i^{(5)}$) ranging from -50 to 50 in intervals of 25. Bounds on $\boldsymbol{\alpha}$ correspond to 2.5th and 97.5th percentiles of a binomial distribution with probability of 5% and size = 4000, expressed as a proportion. ..27

Figure 2-7 Empirical statistical power for correctly rejecting H_0) $\beta = 0$ based on a probit or ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels with a Uniformly-Shaped frequency

distribution. Slope on the latent variable used to generate the data is 1. The figure represents settings where the explanatory covariate consisted of either 51 levels ($x^{(51)}$) ranging from -50 to 50 in intervals of 2 or 5 levels ($x^{(5)}$) ranging from -50 to 50 in intervals of 25. 30

Figure 2-8 Empirical statistical power for correctly rejecting H_0) $\beta = 0$ based on a probit or ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels with a Bell-Shaped frequency distribution. Slope on the latent variable used to generate the data is 1. The figure represents settings where the explanatory covariate consisted of either 51 levels ($x^{(51)}$) ranging from -50 to 50 in intervals of 2 or 5 levels ($x^{(5)}$) ranging from -50 to 50 in intervals of 25. 31

Figure 2-9 Empirical statistical power for correctly rejecting H_0) $\beta = 0$ based on a probit or ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels with a Triangularly-Shaped frequency distribution. Slope on the latent variable used to generate the data is 1. The figure represents settings where the explanatory covariate consisted of either 51 levels ($x^{(51)}$) ranging from -50 to 50 in intervals of 2 or 5 levels ($x^{(5)}$) ranging from -50 to 50 in intervals of 25. 32

Figure 2-10 Empirical statistical power for correctly rejecting H_0) $\beta = 0$ based on a probit or ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels with an Exponentially-Shaped frequency distribution. Slope on the latent variable used to generate the data is 1. The figure represents settings where the explanatory covariate consisted of either 51 levels ($x^{(51)}$) ranging from -50 to 50 in intervals of 2 or 5 levels ($x^{(5)}$) ranging from -50 to 50 in intervals of 25. 33

Figure 2-11 Empirical Type statistical power for inference on H_0) $\beta = 0$ based on ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels and Uniform, Bell, Triangle, or Exponentially shaped frequency distributions. Ordered categorical variables generated from a latent variable with slope of 1. Each scenario is represented by 4000 Monte Carlo replications. The figure represents settings where the explanatory covariate consisted of 51 levels ($x_i^{(51)}$) ranging from -50 to 50 in intervals of 2. 34

Figure 2-12 Empirical Type statistical power for inference on H_0) $\beta = 0$ based on ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels and Uniform, Bell, Triangle, or Exponentially shaped frequency distributions. Ordered categorical variables generated from a latent variable with slope of 1. Each scenario is represented by 4000 Monte Carlo replications. The figure represents settings where the explanatory covariate consisted of 5 levels ($x_i^{(5)}$) ranging from -50 to 50 in intervals of 25. 35

Figure 2-13 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 2 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 37

Figure 2-14 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 2 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 38

Figure A-1 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 50

Figure A-2 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Bell-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 51

Figure A-3 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Triangularly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 52

Figure A-4 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and an Exponentially-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 53

Figure A-5 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 54

Figure A-6 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Bell-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 55

Figure A-7 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Triangularly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 56

Figure A-8 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Exponentially-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 57

Figure A-9 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 58

Figure A-10 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Bell-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 59

Figure A-11 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Triangularly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 60

Figure A-12 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and an Exponentially-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 61

Figure A-13 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 62

Figure A-14 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Bell-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 63

Figure A-15 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Triangularly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 64

Figure A-16 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and an Exponentially-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2. 65

List of Tables

Table 2-1 Probabilities used to define latent-scale thresholds separating sequentially-ranked levels of an ordered categorical response with a Uniformly-Shaped frequency distribution. 18

Table 2-2 Probabilities used to define latent-scale thresholds separating sequentially-ranked levels of an ordered categorical response with a Bell-Shaped frequency distribution. 18

Table 2-3 Probabilities used to define latent-scale thresholds separating sequentially-ranked levels of an ordered categorical response with a Triangularly-Shaped frequency distribution. 19

Table 2-4 Probabilities used to define latent-scale thresholds separating sequentially-ranked levels of an ordered categorical response with an Exponentially-Shaped frequency distribution. 19

Acknowledgements

In reflecting on my time here at K-State, it strikes me how many people have helped me along the way. Dr. Loughin showed me his data cloud and wrote me a reference, Dr. Gadbury helped me with R, Dr. Boyer helped me network, Dr. Higgins answered questions for me even when I wasn't in his class, Dr. Dubnicka worked with me extensively in regard to theory in addition to offering me more personal advice, Robert taught me the Cholesky decomposition, Karen taught me order statistics, and Troy helped with matrix concepts. I could go on and on in this fashion. I think if I were to properly acknowledge everyone, this section might be as long as the report. Instead, let me briefly mention my role models and a few people that have really helped me a lot along the way.

First and foremost, Dr. Bello is an amazing major professor and role model. She tirelessly worked with me on this project. She was receptive to my questions in regard to subject matter, R code, writing, presenting, and even what to wear on an interview. She allowed me the opportunity to try and do things for myself and encouraged me to grow by offering suggestions for improvement. She was a great support to me through every stage of this process and I feel very fortunate to have had a major professor that was there for me so fully. It is her level of attention to detail, quality of work, and constant professionalism that I should strive to attain. The department is lucky to have her.

I would also like to mention Dr. Murray, the next of my role models. Dr. Murray has played an important part in the careers of many of us. I appreciate her taking the time to explain things in a clear fashion and for guiding me through my consulting projects, regardless of the kind of complication I was encountering. I am grateful to have had the opportunity to work on "real" problems and to have had her as my supervisor during this process. It is through watching Dr. Murray that I may learn how to be an excellent consultant. She explains things to clients in such a way that they can understand, she is sensitive to how she is impacting the researchers as people, she is careful in her documentation, and her time management skills are something to be applauded.

Seth Demel has been essential to my functioning in this department. He tutored me extensively in Theory I, Theory II and Theory of Linear Models. He was there to comfort me

when I fell and congratulate me when I succeeded. He is a true friend and I am thankful for all he has done for me.

Finally, I would like to extend my gratitude to my family. My mother and my sister have been helping me with my daughter, Maya, since she was born. During the final weeks of my graduate work they allowed Maya to live with them so that I could stay at school until late in the night and come back directly in the morning. I am not confident that I could have finished in a timely fashion without their support. I am appreciative that my daughter is also understanding of my situation as she has handled the whole situation with a grace and understanding that far surpasses her age.

It is said that it takes a community to raise a child. It certainly has to raise mine. I think the same could be said of "raising a graduate student." Thank you to everyone that has been a part of my supporting community during my academic infantile stage, you have been great parents.

Dedication

I would like to dedicate this work to my daughter, Maya, the one person who can always make me smile. May she have happiness all of her days. I love you sunshine-passion-flower.

Chapter 1 - Literature Review

1.1 Introduction

Ordered categorical responses are prevalent in modern practice. For example, in medicine, endometrial cancer may be indicated with an ordered categorical response such that patients are classified as in the diseased state or not in the diseased state (Xu et al., 2007). In sensory analysis such outcomes may be utilized to describe preferences with the categories "excellent, very good, good, neutral, poor, and very poor" (Snell, 1964). In public health, ordered categorical responses have been used to study student tobacco and health knowledge by forming categories of correct responses to a survey (Hedeker & Gibbons, 1994). In nutrition, they may be employed to examine macronutrient and micronutrient intake by grouping a continuous variable into quintiles (Xu et al., 2007). In genetics, they may be used to study phenotype as it relates to genotype by looking whether or not animals with specific genetics have 0, 1, 2, 3, 4, or more dead fetuses at a various time points of gestation (Yi, Samprit, & Yandell, 2007). Within psychology, ordered replies may be used to study oppositional defiant disorder by measuring an individual's reported maladaptive behavior on a frequency of occurrence scale consisting of the categories, "never in the past month", "1-2 times in the past month", "3-4 times in the past month", "2-4 times per week", "1 time per day", "2-5 times per day", "6-9 times per day", and "10 or more times per day" (Taylor, Burns, Rusby & Foster, 2006). The final score on the Quality of Life in Schizophrenia Survey is also composed of ranked levels, namely, "severely compromised quality of life", "moderately compromised quality of life", and "unaltered quality of life" (Abreu, Siqueira, Cardoso, & Caiaffa, 2008). In environmental science, quality of evaluation of greenhouse emissions may be recorded as A for the highest quality, most direct assessments through D for fully estimated assessments (*U.S.,* n.d.). NIH research funding is evaluated using a crude scale of the integers 1 through 5 for less competitive grants and a more refined scale between 1 through 5, moving in .1 increments, for more competitive grants (Johnson, 2008).

The objective of the following review of the literature is to discuss the unique characteristics of ordered categorical data and to examine statistical methods utilized to infer upon these data.

1.2 Ordered Categorical Responses

Ordered categorical responses can be conceptualized as being a discretization of an underlying continuous latent variable, usually with a normal distribution (Abreu et al., 2008; Agresti, 1990; Liu & Agresti, 2005; McCullagh, 1980; Winship & Mare, 1984). Let

$$y_i^* = \omega^* + \beta^* x_i + \varepsilon_i^* \qquad \text{Equation 1}$$

where y_i* is the i^{th} realized value of a latent variable defined in the range (-∞, ∞), ω* is an intercept parameter, β* is a slope parameter, x_i is the i^{th} fixed value for x, and ε_i* is the error associated with the i^{th} observation. Let the ordered categorical realization of Y* be denoted by Y, whereby y_i assumes values $j = 1,..., J$. Let there be $J+1$ cutpoints τ_m, $m = 0,... , J$, so that

$$y_i = \begin{cases} 1 & if\ \tau_0 = -\infty \le y_i^* < \tau_1 \\ 2 & \quad if\ \tau_1 \le y_i^* < \tau_2 \\ \vdots & \qquad \vdots \\ J & if\ \tau_{J-1} \le y_i^* < \tau_J = \infty \end{cases}$$

It then follows that:

$$P(y_i = j) = P(\tau_{m=j-1} < y_i^* < \tau_{m=j})$$

that is, the probability that $y_i = j$ is the probability that y_i* takes a value between $\tau_{m=j-1}$ and $\tau_{m=j}$.

One may then see that ordered categorical frequency distribution functions may assume a variety of shapes dependent on the placement of the cutpoints τ. According to Javaras and Ripley (2007), ordered categorical data that are generated as human reactions to a survey tend to be influenced by response styles and are frequently asymmetrical. An exemplification of this may be seen in outcomes from quality of life scales (Abreu et al., 2008) and in responses to antimicrobial use in feedlot cattle (McIntosh et al., 2009). One can imagine a variety of scenarios where ordered categorical variables might have frequency distributions that assume any number of shapes. In fact, in the paper by McIntosh et al. (2009), the probability distribution functions of various ordered categorical responses to antimicrobial use in feedlot cattle assumed a range of symmetric and skewed shapes.

It should be noted that, in contrast to continuous variables which have an infinite number of possible levels that they may assume, ordered categorical responses may only assume a finite number of levels. For illustrative purposes, I may examine previously mentioned examples in relation to their number of levels. Endometrial cancer was indicated with a 2-level ordered

categorical response: disease absent or disease present (Xu et al., 2007). The final score on the Quality of Life in Schizophrenia had an additional level, being composed of 3 ordered levels (Abreu et al., 2008). Quality of measurements of green house emissions (U.S., n.d.) were studied using a 4-level ordered categorical response, while micronutrient and macronutrient intake were examined with a 5-level ordered categorical response (Xu et al., 2007). A 6-level ordered categorical response was utilized to investigate stillbirth phenotype in mice (Yi et al., 2007), the survey for student tobacco and health knowledge has 7 levels (Hedeker & Gibbons, 1994), while oppositional defiant disorder was studied using an 8-level outcome (Taylor et al., 2006). According to Johnson (2008), NIH research funding is evaluated using a 10-level scale for less competitive grants and a 50-level scale for more competitive grants. The numerical recording of ordered categorical responses, especially when the ordered categorical responses have many levels, may potentially lead to confusion as to the theoretically sound methods for data analysis and inference.

Ordered categorical responses are unique among discrete variables. While categorical in nature, the ordering of their categories contains additional information about the process of interest relative to outcomes with nominal categories. The ranking information in ordered categorical responses may be considered, at least conceptually, to resemble that of interval data. However, with interval data, the numeric representation of observations is reflective not only of relative ranking but also of the magnitude of the distance between observations. In contrast, with ordered categorical data, the numeric representation of the observations is only reflective of rank; the distances between contiguous categories may not be proportional (Long, 1997).

Failure to recognize this subtlety seems to be rather common in practice, whereby inferences for ordered categorical responses are often based on statistical methodology developed for interval data (Liu & Agresti, 2005). In fact, one may find examples in the literature where this has occurred. Before being analyzed by Yu as ordered categories, the previously mentioned phenotype example was analyzed with interval level techniques by Rocha, Eisen, Seiwerdt, Vleck and Pomp (2004). McIntosh et al. (2009) analyzed the numerical scores assigned to the categories “always”, “often”, “sometimes”, “rarely” and “never” as though they were realizations of a continuous variable. Russel and Boboko (1992) set up a scenario based on willingness to revise a manuscript as measured by a 5-point scale ranging from “very

unmotivated" to "unmotivated," where not only did they wish to employ linear regression, but to investigate interaction effects as well.

1.3 Statistically Sound Methods for Ordered Categorical Data

A number of sound statistical techniques methodologies have been developed, and are available, for inference on ordered categorical responses. These methods allow one to make inferences about ordered categorical responses in a way that acknowledges the discreteness of the response and takes advantage of the additional information contained in the ranked order of the response categories. Amongst those statistical models most frequently encountered in practice are the probit model and the proportional odds logistic model. If, instead, one chooses to disregard the ordering and strictly treat the response categories as nominal, the multinomial model can serve as an alternative choice (Abreu et al., 2008; Agresti, 1990; Liu & Agresti, 2005; McCullagh, 1980; Winship & Mare, 1984). I now review these models in further detail.

1.3.1 Probit

When working with the probit model, as is discussed by Long (1997), I assume that the error (ε_i*) and the latent variable (y*) follow the formula in equation 1, so that the error follows a standard normal distribution with mean zero and variance one. This probability density function is:

$$\phi(\varepsilon^*) = \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon^{*2}}{2}}$$

The cumulative density function is:

$$\Phi(\varepsilon^*) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt$$

The probit model then uses an inverse normal cumulative distribution function, also known as the probit link function, to map the probability scale (0,1) onto a scale on (-∞, ∞). The standard normal cumulative distribution function is used to determine the probability of the ordered categorical response in each ordered category. In illustration,

$$P(y_i = 1 \,|x_i) = P(\tau_0 = -\infty \leq y_i^* \leq \tau_1) = \Phi(\tau_1 - \omega^* - \beta^* x_i)$$

$$P(y_i = 2 \,|x_i) = P(\tau_1 \leq y_i^* \leq \tau_2) = \Phi(\tau_2 - \omega^* - \beta^* x_i) - \Phi(\tau_1 - \omega^* - \beta^* x_i)$$

.

.

.

$$P(y_i = J-1\,|x_i) = P\left(\tau_{J-2} \le y_i^* \le \tau_{J-1}\right) = \Phi\left(\tau_{J-1} - \omega^* - \beta^* x_i\right) - \Phi(\tau_{J-2} - \omega^* - \beta^* x_i)$$

$$P(y_i = J\,|x_i) = P\left(\tau_{J-1} \le y_i^* \le \tau_J\right) = 1 - \Phi\left(\tau_{J-1} - \omega^* - \beta^* x_i\right)$$

where all notation remains the same as it was previously defined during the delineation of the latent variable conceptualization. In estimating model parameters, τ_1 is usually set to zero to ensure model identifiability. It is worth noting that fitting this model necessitates estimation of J+1 parameters, namely J-1 threshold parameters τ_m, an intercept ω, and a slope, β. Estimating this number of parameters may pose practical difficulties if the data are sparse in any of the categories (Liu & Agresti, 2005). Additionally, practitioners may have a difficult time interpreting the parameters of this model in a way that is meaningful for their audiences, causing hesitance in use. As is further explained in Long (1997), if an appeal to a latent normal variable seems reasonable, one is faced with the task of explaining what, in a practical sense, it means for Y* to increase by β standard deviations for a unit increase in $\boldsymbol{x}$ while both Y* and it's variance remain unknown. It is from the analysis of partial change in probability that I get the meaning of β separate from the latent variable. That is, the change in estimated probability that the j^{th} category is observed as X increases by 1 unit is:

$$\begin{aligned}\{P(Y = j|X+1) - P(Y = j|X)\} \\ = \Phi\left\{\hat{\tau}_{m=j} - \hat{\omega} - \hat{\beta}(X+1)\right\} - \Phi\left\{\hat{\tau}_{m=j-1} - \hat{\omega} - \hat{\beta}(X+1)\right\} \\ - \Phi\left\{\hat{\tau}_{m=j} - \hat{\omega} - \hat{\beta}X\right\} + \Phi\left\{\hat{\tau}_{m=j-1} - \hat{\omega} - \hat{\beta}X\right\}\end{aligned}$$

If one wishes to make additional comments while remaining on the ordered categorical scale, one is faced with an onerous task. A thorough analysis will utilize a variety of techniques as no one is sufficient to capture the nonlinear relationship in the probabilities. These approaches include investigating predicted probabilities and their ranges, both in tabular and graphic format, and looking at partial and discrete changes in probability.

1.3.2 Proportional Odds Logistic

A popular type of model that has parameters with a somewhat clearer interpretation is the proportional odds logistic model. In the set of all models created specifically for ordered categorical responses, this model is the most frequently used (Liu & Agresti, 2005). As one may learn from Abreu et al. (2008) and Agresti (1990), the proportional odds logistic model enables a practitioner to estimate and discuss odds ratios as well as probabilities. Odds are defined as the

probability of success divided by the probability of failure. I may specify a success to be a realization of the ordered categorical variable *y* in category j or less whereas a failure is being a realization of the ordered categorical variable *y* in any remaining category. This results in an odds model of the form:

$$odds_{\mathrm{m}} = \frac{P(Y \leq j \mid x)}{1 - P(Y \leq j \mid x)} = \frac{P(Y \leq j \mid x)}{P(Y > j \mid x)} = \omega_{\mathrm{m}} + \beta x \qquad \omega_1 < \omega_2 < \cdots < \omega_{J-1}$$

where $m = 1, \ldots, J\text{-}1$ with J being the number of ordered categories, ω_{m} are dichotomy specific intercept parameters, and where β is the common slope parameter. An odds ratio (OR) for a predictor's effect on the response is:

$$OR = \frac{\frac{P(Y \leq j \mid x_1)}{P(Y > j \mid x_1)}}{\frac{P(Y \leq j \mid x_2)}{P(Y > j \mid x_2)}} = \frac{odds_1}{odds_2}$$

where $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are different levels of the predictor variable; for example, exposed and non-exposed.

In the proportional odds model, the OR is assumed to be the same for all cumulative dichotomies of the categories of the response variable. This is commonly known as the proportional odds assumption. According to Agresti (1990), the model structure implies

$$\log(OR) = \beta(\boldsymbol{x_1} - \boldsymbol{x_2})$$

so that the log of the OR is proportionate to the distances between explanatory variables for all intercepts ω_m. Note that it is possible to choose different thresholds τ_i such that the underlying latent variable $\boldsymbol{y^*}$ is regrouped into different categories of $\boldsymbol{y}$. This will not affect β, but will instead only impact the intercepts ω_m. As is mentioned in Long (1997), the error distribution on the latent variable is assumed to have a logistic distribution with mean zero and variance $\pi^2/3$. Its probability distribution function is then

$$\lambda(\boldsymbol{\varepsilon}) = \frac{e^{\varepsilon}}{[1 + e^{\varepsilon}]^2}$$

and cumulative distribution function

$$\Lambda(\boldsymbol{\varepsilon}) = \frac{e^{\varepsilon}}{1 + e^{\varepsilon}}$$

The above restraints result in simultaneously fitting *J-1* response curves that are identical in all aspects but their intercept terms. The differing intercepts result in curves that are shifted along the x-axis by $(\omega_{m-1} - \omega_m)/\beta$ units (Agresti, 1990).

The interpretation of parameter estimates from this model may be more easily understood by practitioners than those from the probit model. This may be due to the higher frequency of application of this model; however, neither this ease of comprehension nor the technique's frequency of use has surpassed that associated with ordinary least squares linear regression (Liu & Agresti, 2005). As an additional impediment to a researcher choosing this model, fitting it still requires a minimum of *J* parameters to be estimated. This means that sparse data in any of the categories of $\boldsymbol{y}$ may still create model fitting complications. To further exacerbate the model-fitting process, one must contend with the fact that it may not be reasonable to impose the proportional odds assumption. That is, assuming that the response curves in the logit link scale share the same slope may be too restrictive; the log odds between groups of consecutive categories may not all be the same. As a consequence, I may need to allow for different slope parameters for each response curve β_m , ($m = 1, \ldots, J$) in the logit scale. Note that when building a model with multiple covariates, the proportional odds model insists that all covariates meet the proportional odds assumption. In contrast, a statistical methodology called the partial proportional odds model allows some covariates to adhere to the proportional odds assumption while allowing others to deviate from it (Abreu et al., 2008). When working with a model with multiple covariates, instead of the single covariate modeling primarily discussed in this paper, this model may provide an option that is more parsimonious than a multinomial regression but more flexible than a full proportional odds regression. The partial proportional odds model is not heavily utilized in practice (Abreu et al., 2008).

1.3.3 Multinomial Model

The multinomial model is what is frequently defaulted to if the proportional odds model assumptions are not reasonably satisfied (Long, 1997). The multinomial model continues to model the log of the odds to a linear form, but it does so by simultaneously comparing each category to all remaining categories. This model may be expressed in terms of odds or in terms of probabilities. The multinomial model does not look at cumulative probability, but can instead

be expressed as the probability of a given category. As per Long (1997), the probability expression of this model is:

$$Probability(y_i = j \mid \boldsymbol{x_i}) = \frac{e^{\boldsymbol{x_i}[\omega_j + \beta_j + c_j]}}{\sum_{m=1}^{J} e^{\boldsymbol{x_i}[\omega_m + \beta_m + c_m]}}$$

where y_i is the i^{th} ordered categorical observation, j is the category value currently under inspection, $\boldsymbol{x_i}$ is the i^{th} column of the design matrix $\boldsymbol{x}$ where $\boldsymbol{x}$ contains a row of ones and then subsequent rows of covariate values, and ω_j and β_j are, respectively, the intercept and the slope corresponding to the j^{th} category, m is an indicator of ordered category ($m = 1, \ldots, J$), and c is a constant. I must add a constraint to make the model identifiable. If I set an α-β combination to zero, say ω_1 and β_1, then our model becomes:

$$Probability(y_i = j \mid \boldsymbol{x_i}) = \frac{e^{\boldsymbol{x_i}[\omega_j + \beta_j]}}{\sum_{m=1}^{J} e^{\boldsymbol{x_i}[\omega_m + \beta_m]}}$$

where $\boldsymbol{x_i}$ is a design matrix with a row of ones for the intercept and a row of covariates; all other parameters retain their interpretation. This model becomes cumbersome to interpret very quickly and is the most susceptible to sparse cells of all the models discussed thus far. Oftentimes, researchers choose not to implement any of the above methods in search of techniques that they are more accustomed to employing and that are numerically tractable under conditions of exiguous category counts (Liu & Agresti, 2005).

1.4 Interval Methods of Practical Use

Thus far techniques that acknowledge the categorical nature of ordered categorical data have been discussed. However, many of these statistical methods are often underutilized by practitioners, who in turn favor techniques designed for interval data (Liu & Agresti, 2005). Previous studies have examined the robustness of making inferences about ordered categorical responses based upon statistical methods developed for continuous data. The limitation of doing so relies on the assumption of normality of the error term, which is often not satisfied (Long, 1997).

1.4.1 T-test

The *t*-test is commonly employed when two groups need to be compared. While this test has been developed for continuous data, it has also been used for ordered categorical responses (Boneau, 1960). The t-test statistic is:

$$t = \frac{\bar{y}_1 - \bar{y}_2}{\sqrt{\frac{\sum y_1^2 - n_1\bar{y}_1 + \sum y_2^2 - n_2\bar{y}_2}{n_1 + n_2 - 2}\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \qquad \textbf{Equation 2}$$

where $\bar{y}_i$ *i*s the mean for the i^{th} category, y_i is an observation from group *i*, and n_i is the number of observations in group *i* . The mean is defined as:

$$\bar{y}_i = \frac{\sum_{i=1}^{n_i} y_j}{n_i}$$

This test compares the result of equation 2 to a critical point on the student's *t*-distribution having *n* - 2 degrees of freedom. Underlying assumptions for t-test based inference include: 1) that the 2 groups being sampled from come from a normal distribution, or that the sample sizes are relatively large so as to have normality of the distribution of the mean via the central limit theorem, 2) that the variances of those distributions are homogeneous, and 3) that the observations are independent.

Research on the inferential robustness of the *t*-test has focused on cases of violation of assumptions of normality and/or equality of variance, rather than violation of independence of observations, as this may be induced by design. Gayen (1949) developed theory that suggested that a one-sample *t*-test with small sample sizes is robust to violations of normality if the population the data was drawn from could be well represented by the third or fourth approximation of an Edgeworth series. Boneau (1960) furthered this investigation of the *t*-test with Monte Carlo simulations and looked at small sample size scenarios with various combinations of distribution, sample size, and variance. He found the *t*-test based inference to be robust to violations of assumptions of normality and heterogeneity of variance, even with small sample sizes. This was evidenced by empirical distributions of *t* that resembled their theoretical counterparts.

Heeren and D'Agostino (1987) computed the sampling distributions of the *t*-test statistic applied to ordered categorical responses with a number of different probability distributions. The ordered categorical responses had 3, 4, or 5 levels and sample sizes used ranged between 5

and 20. They found their calculated significance to be close to their specified nominal significance across combinations of distribution and number of categorical response.

1.4.2 ANOVA

Analysis of Variance (ANOVA) techniques may be conceptualized as a generalization of the *t*-test to the case where comparisons between more than two groups are of interest. The simple one-way fixed effects ANOVA, as discussed by Kutner, Nachtsheim, Neter and Li (2005), assumes a model:

$$y_{ij} = \mu + \omega_i + \varepsilon_{ij} \qquad \varepsilon_{ij} \sim independent\ N(0, \sigma^2)$$

where y_{ij} is the j^{th} observation in the i^{th} treatment group, μ is the overall mean, ω_i is the distance of the i^{th} group mean from μ, ε_{ij} is the error associated with the ij^{th} observation and σ is the standard deviation of the error. Note that the above model has built into it the same assumptions as the *t*-test, namely normality, homogeneity of variance and independence of the error terms, and thus the observations. The assumptions of normality and homogeneity of variance are of most interest here.

Hsu and Feldt (1969) conducted a computer simulation to discover how well ANOVA performs when used on 5, 4, 3 or 2-point ordered categorical responses. Responses for any given point scale were generated from treatment populations that came from one of three distribution. The distributions in these scenarios were intended to represent different levels of skew, platykurtosis, and variance. For an ordered categorical response with 5 levels, the study showed that ANOVA had excellent Type I error (α) provided that all observations were generated from a single distribution, regardless of which distribution this was. That is, no adjustment to the utilized α level was required to maintain nominal α. This occurred with as few as 11 observations per treatment group. If observations were generated from different distributional scenarios, an adjustment of utilized significance to .06 was suggested to maintain a nominal α = .05. With the 4-point scale, the ANOVA produced, on average, empirical α within .5% of nominal α if observations came from a single distributional scenario. In contrast, when different scenarios were used such that there were differences in the variances across treatment populations, the average difference between the nominal significance level and the empirical significance level ranged between 0.23% and 0.80%. With the 3-point scale, Type I error did better than for the 4-point scale but not as well as for the 5-point scale. Justification for the

superior performance with an even number of points was not provided. The difference between the nominal and empirical α was less than a half percent regardless of combination of distributional scenario used. In the binary condition, the estimated difference in α became as large as 3.3%, but the average difference decreased with larger sample sizes.

Glass, Peckman, and Sanders (1972) compiled an extensive review of the work on robustness of ANOVA. For continuous response, they found that non-normality and skewness of the error distribution had very little effect on the Type I error (α) or the power of the fixed effects ANOVA *F*-test statistic regardless of whether data were balanced or unbalanced. They stated that empirical α was less than nominal α when distributions were leptokurtic and more than nominal α when distributions were platykurtic for both balanced and unbalanced sample sizes. In regard to the effect of kurtosis on power, they stated that empirical power tended to be less than theoretical power when the distributions were platykurtic, but more than theoretical power when the distributions were leptokurtic. This effect was more pronounced for smaller sample sizes, and it was present regardless of whether the design was balanced or unbalanced. They also explained that heterogeneity of variance had very little effect on Type I error (α) when the data were balanced, with the effect being within a few hundredths when power is considered as a decimal between zero and one. However, there seemed to be a trend toward a slight increase over the nominal α level. When the data were unbalanced, the heterogeneity of variance had a greater impact on empirical α. Furthermore, with small sample sizes drawn from populations with greater variance, empirical α was greater than the nominal α level. In turn, with small sample sizes drawn from populations with less variance, empirical α was decreased compare to the nominal α value.

Arnold (1980) further validated these conclusions stating that the ANOVA F was robust to violations of normality if the number of groups was small and the sample size in each group was high. Reed and Stark (1995) did further computer simulation to provide evidence of the robustness of the ANOVA with equal sample sizes and 3, 4, or 6 groups across a variety of distributions. They found that the distributions sampled from had very little effect for inference in terms of realized Type I error (α), but noted that if the distribution being sampled from had longer tails, like the Cauchy distribution, the ANOVA *F*-test was conservative. They also found the ANOVA F-test statistic to work comparably to, or better than, a variety of nonparametric statistics when sampling from a panoply of distributions and a wide assortment of equal and

unequal sample sizes. Akritas and Papas (2004) informed that the ANOVA *F*-test was asymptotically correct for balanced data even without normality due to a convergence in distribution. Additionally, they found, via computer simulation, that the ANOVA *F*-test statistic was robust with unbalanced data if the group sizes were large.

1.4.3 Ordinary Least Squares Linear Regression

As is discussed by Kutner et al. (2005), ANOVA can be considered a special case of ordinary least squares regression where the predictor variable is categorical, rather than continuous. Also, an ANOVA model does not put restrictions on the group means that they follow a certain functional form (i.e. linear, quadratic, cubic, etc.), unlike ordinary least squares linear regression. In fact, if the categorical predictor variable has no inherent ordering, the levels of the predictor variable may be rearranged in their positioning on the x-axis, rendering meaningless any inference on a functional form of the relationship between predictor and response.

In general terms, when both the dependent response and the independent explanatory variables are continuous, their relationship can be modeled using linear regression where the addition of a slope constraint is something that can be informative. In this circumstance I have the model:

$$y_i = \omega + \beta x_i + \varepsilon_i \qquad independent\ \varepsilon_i \sim N(0, \sigma^2)$$

where y_i is the i^{th} observation of *Y*, ω is the intercept parameter, x_i is the i^{th} fixed value of the predictor variable, β is the slope parameter, or rate of change in Y per unit increase in $\boldsymbol{x}$, and ε_i is the error associated with the i^{th} observation. Note that the error is independently and identically normally distributed with mean zero and constant variance σ^2.

Given the robustness of ANOVA to the type of violations of assumptions common with ordered categorical data, the question may arise as to whether or not the more general form of ordinary least squares linear regression may exhibit the same qualities of robustness when applied to ordered categorical data. Showing this robustness of ordinary least squares linear regression would be of practical importance as these techniques are frequently used in practice (Liu & Agresti, 2005). To the author's best knowledge, the inferential implications in regard to Type I error (α) and statistical power of analyzing ordered categorical data with ordinary least squares linear regression are unknown.

It should be noted that the normal distribution upon which ordinary least squares linear regression and ANOVA techniques are based is a continuous probability distribution. As such, any data that is not continuous, such as ordered categorical data, by definition violates the underlying normality assumption. Furthermore, while many probability mass functions for ordered categorical variables have a general bell shape similar to a normal distribution, it is also common for them to be skewed (Abreu et al., 2008; Javaras & Ripley, 2007). Further research on the robustness of ordinary least squares linear regression in cases where model assumptions are violated is needed to better inform decisions on OLSLR-based inference on ordered categorical data.

1.5 Summary

Ordered categorical responses are discrete categorical variables that are commonly encountered in practice and that contain additional information on the relative ranking of the categories. A number of statistical methods are frequently used to make inference on this type of data. The greater ease of implementation and interpretation of the statistical methods that assume a continuous and normal nature of the data frequency may sway practitioners towards implementing these techniques on ordered categorical responses despite potential violation of basic model assumptions. It is well-described that both the *t*-test statistic and the ANOVA *F*-test statistic are robust to these violations of assumptions in a variety of scenarios and under certain circumstances. However, the inferential consequences of using ordinary least squares linear regression for inference on ordered categorical responses remain unknown.

Chapter 2 - Ordinary Least Squares Regression of Ordered Categorical Data

2.1 Introduction

Many researchers are faced with the challenge of analyzing ordered categorical responses (Liu & Agresti, 2005). Examples of interest in agriculture include quality assessments, such as for soil (De Groote et al., 2010) or food products (Hernandez et al., 2005), evaluation of lesion severity, such as teat ends status in dairy cattle (Raubertas & Shook , 1982), and antimicrobial use in feedlot cattle (McIntosh et al., 2009). For instance, McIntosh (2009) worked with ordered categorical responses to describe the frequency of antimicrobial use in feedlot cattle as "never", "rarely", "sometimes", "often" and "always". In that study, categories were enumerated from 1 to 5, respectively, to reflect their natural ordering based on frequency of use. Enumeration of ranked categories is commonly observed for ordered categorical data. However, such enumeration is only indicative of order and should not be confused with magnitude of the distance, or proportionality, between consecutive categories of the response.

The probit regression model and the proportional odds logistic regression model are examples of models developed specifically to take advantage of the information available in the ordering of the realized response categories response while acknowledging its discrete categorical nature (Long, 1997). However, fitting a probit model or a proportional odds model may be complicated by technical limitations such as data sparsity (Liu & Agresti, 2005). Additionally, interpretation of parameter estimates from these models can be perceived by the disciplinary scientist as more mathematically involved and less intuitive than that of ordinary least squares linear regression (OLSLR). Familiarity with OLSLR may be accredited to its frequent implementation in the sciences as well as to the rather straight-forward interpretation of slope parameters in terms of "rate of change". It may be based on a combination of these issues that Liu and Agresti (2005) assert that OLSLR is a frequent choice for the analysis of ordered categorical responses.

The main concern of using OLSLR to fit ordered categorical responses is violation of model assumptions, which may call into question any subsequent inference. Ordinary least squares linear regression assumes that errors be mutually independent and normally distributed

with mean zero and constant variance (Kutner, Nachtsheim, Neter & Li, 2005). This, in turn, implies that the response variable is continuous in nature and symmetrically distributed around a mean with the same such variance parameter as the distribution of the error. In modeling ordered categorical responses with OLSLR, the underlying assumption is that the ranking of the categories nominally identified as 1,2,3... bear additional information on the proportionality of the distance between categories (Long, 1997). For instance, in the example discussed in the first paragraph, category 2 (labeled as "rarely") would indicate twice as frequent antimicrobial use than category 1 (labeled as "never") and, subsequently, category 4 (labeled as "often") would indicate twice as frequent antimicrobial use compared to category 2 (labeled as "rarely"). While "often" is clearly more frequent use than "rarely" or "never", the assumption of proportionality between categories is highly questionable and may not be supported in the context of a given application. The inferential implications of assuming proportionality between ranked ordered categories are not known.

In addition, particular features of an ordered categorical response may impact the quality of inference. For example, the categories of ordered categorical variables may have asymmetrically distributed frequencies (Abreu et al., 2008; Javaras & Ripley, 2007) and may not have a constant variance (Long, 1997).

Furthermore, the number of levels, or number of categories, of an ordered categorical response is relevant when fitting a variety of models. Liu and Agresti (2005) claim that when fitting a proportional odds model, there is little gain in efficiency when using more than 4 levels. Agresti (1990) states that when utilizing weighted least squares to fit a mean response linear model to ordered categorical data with independent multinomially distributed error, using a large number of levels will help to ensure estimates of means within range of the ordered categories. This suggests that the number of levels of an ordered categorical outcome might be an important factor to consider in the functionality and practical application of fitting OLSLR to ordered categories.

The objective of this study is to investigate the implications for statistical inference of fitting OLSLR to ordered categorical responses. Using simulation studies, I evaluate empirical Type I error (α) and statistical power for inference on ordered categorical responses under a variety of scenarios. These simulation scenarios are motivated by subject-matter applications encountered in the scientific literature (Abreu et al., 2008; De Groote et al., 2010; Hernandez et

al., 2005; Javaras & Ripley, 2007; McIntosh et al., 2009; Raubertas & Shook, 1982; Russell & Bobko, 1992; Xu et al., 2007; Yi et al., 2007; *U.S.,* n.d.).

2.2 Data Simulation Methods

Using simulation, I investigated the inferential properties of OLSLR as applied to ordered categorical responses. I evaluated ordered categorical responses with frequency distribution of uniform, belled, triangular, or exponential shape. I also considered multiple numbers of categories (or levels) of the ordered categorical response, specifically, 2, 3, 4, 5, 7, and 10 levels. For each of the 24 scenarios (4 frequency distribution shapes x 6 number of categorical levels of the ordered categorical response), 4,000 Monte Carlo replicates were produced. For each replicate, I generated 300 realizations of a normally distributed latent random variable according to the following equation:

$$y_i^* = \omega^* + \beta^* x_i + \varepsilon_i$$

where y_i^* is the i^{th} observation on the latent normal scale (i = 1, …, 300); ω^* and β^* are the intercept and slope parameters in the latent scale; x_i is the known covariate value corresponding to the i^{th} observation and ε_i is the error associated with the i^{th} observation and is assumed to be independently and identically normally distributed with mean zero and constant variance σ^2 (i.e. $\varepsilon_i \sim$ i.i.d N(0, σ^2)). Realizations of the ordered categorical responses y_i were generated by discretizing the latent continuous variables y_i^* using arbitrary thresholds τ between neighboring ranked categories. These thresholds were defined using category-specific probabilities as listed in tables 2-1 to 2-4. These probabilities represent areas under the normal density curve for non-overlapping, sequentially-ranked neighboring ordered categorical levels, such that the frequency distribution of the resulting ordered categorical response was uniform, belled, triangular or exponential in shape, respectively. For instance, from table 2-3, I use the latent variable y_i^* to generate an ordered categorical response y_i with 4 levels such that

$$y_i = \begin{cases} 1 & if\ -\infty \le y_i^* < \tau_1\ where\ P(-\infty \le y_i^* < \tau_1) = 0.1 \\ 2 & if\ \tau_1 \le y_i^* < \tau_2\ where\ P(\tau_1 \le y_i^* < \tau_2) = 0.2 \\ 3 & if\ \tau_2 \le y_i^* < \tau_3\ where\ P(\tau_1 \le y_i^* < \tau_2) = 0.3 \\ 4 & if\ \tau_3 \le y_i^* < \infty\ where\ P(\tau_3 \le y_i^* < \infty) = 0.4 \end{cases}$$

whereby $-\infty < \tau_1 < \tau_2 < \tau_3 < \infty$. For illustrative purposes, figures 2-1 to 2-4 depict histograms of the frequency distribution of one replicate of a simulated ordered categorical

response with 7 levels assuming uniform, belled, triangular and exponentially-shaped frequency distributions.

I simulated each scenario using $\beta^* = 0$ and $\beta^* = 1$ in order to assess Type I and Type II errors. For simulation purposes, x_i had a range of (-50, 50) and was allowed to assume 2 settings on each scenario. In one setting, namely, $\boldsymbol{x}^{(51)}$, x_i was allowed to take 1 of 51 possible integer values ranging from -50 to 50 in intervals of 2. In the second setting, namely, $\boldsymbol{x}^{(5)}$, x_i was allowed to take one of 5 possible integer values ranging from -50 to 50 in intervals of 25. The variance of the error (ε_i) was set such that the empirical and theoretical power for testing H_0) β^* =0 using OLSLR was approximately 0.80. This resulted in an error variance (σ^2) of 31648.41 in setting $\boldsymbol{x}^{(51)}$ and 47089 in setting $\boldsymbol{x}^{(5)}$. All simulation computations were conducted using the statistical software R Version 2.9.2 (R Development Core Team, 2009).

Table 2-1 Probabilities used to define latent-scale thresholds separating sequentially-ranked levels of an ordered categorical response with a Uniformly-Shaped frequency distribution.

Probability an observation takes value j.	Number of levels of the ordered categorical response.					
j	2	3	4	5	7	10
1	1/2	1/3	1/4	1/5	1/7	1/10
2	1/2	1/3	1/4	1/5	1/7	1/10
3	.	1/3	1/4	1/5	1/7	1/10
4	.	.	1/4	1/5	1/7	1/10
5	.	.	.	1/5	1/7	1/10
6	.	.	.	.	1/7	1/10
7	.	.	.	.	1/7	1/10
8	.	.	.	.	.	1/10
9	.	.	.	.	.	1/10
10	.	.	.	.	.	1/10

Table 2-2 Probabilities used to define latent-scale thresholds separating sequentially-ranked levels of an ordered categorical response with a Bell-Shaped frequency distribution.

Probability an observation takes value j.	Number of levels of the ordered categorical response.					
j	2	3	4	5	7	10
1	1/2	1/4	1/6	1/9	1/16	1/30
2	1/2	2/4	2/6	2/9	2/16	2/30
3	.	1/4	2/6	3/9	3/16	3/30
4	.	.	1/6	2/9	4/16	4/30
5	.	.	.	1/9	3/16	5/30
6	.	.	.	.	2/16	5/30
7	.	.	.	.	1/16	4/30
8	.	.	.	.	.	3/30
9	.	.	.	.	.	2/30
10	.	.	.	.	.	1/30

Table 2-3 Probabilities used to define latent-scale thresholds separating sequentially-ranked levels of an ordered categorical response with a Triangularly-Shaped frequency distribution.

Probability an observation takes value j.	Number of levels of the ordered categorical response.					
j	2	3	4	5	7	10
1	1/3	1/6	1/10	1/15	1/28	1/55
2	2/3	2/6	2/10	2/15	2/28	2/55
3	.	3/6	3/10	3/15	3/28	3/55
4	.	.	4/10	4/15	4/28	4/55
5	.	.	.	5/15	5/28	5/55
6	.	.	.	.	6/28	6/55
7	.	.	.	.	7/28	7/55
8	.	.	.	.	.	8/55
9	.	.	.	.	.	9/55
10	.	.	.	.	.	10/55

Table 2-4 Probabilities used to define latent-scale thresholds separating sequentially-ranked levels of an ordered categorical response with an Exponentially-Shaped frequency distribution.

Probability an observation takes value j.	Number of levels of the ordered categorical response.					
j	2	3	4	5	7	10
1	1/3	1/7	1/15	1/31	1/127	1/1023
2	2/3	2/7	2/15	2/31	2/127	2/1023
3	.	4/7	4/15	4/31	4/127	4/1023
4	.	.	8/15	8/31	8/127	8/1023
5	.	.	.	16/31	16/127	16/1023
6	.	.	.	.	32/127	32/1023
7	.	.	.	.	64/127	64/1023
8	.	.	.	.	.	128/1023
9	.	.	.	.	.	256/1023
10	.	.	.	.	.	512/1023

Figure 2-1 Realization of a simulated ordered categorical response with 7 levels and a Uniformly-Shaped frequency distribution. Histogram is based on a single simulated Monte Carlo replication with 300 observations.

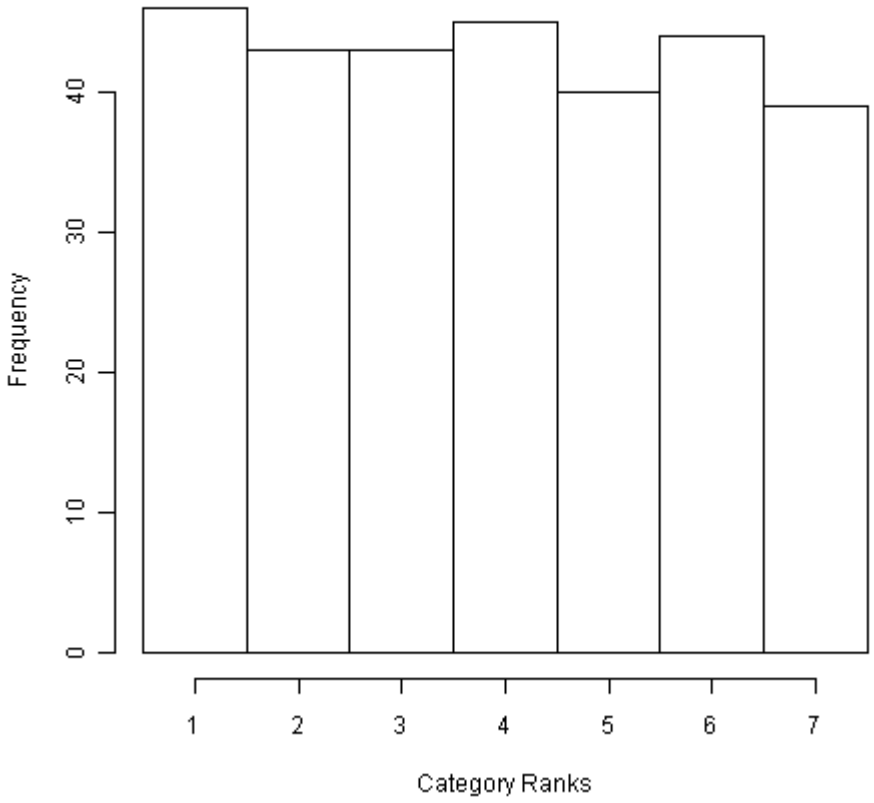


Figure 2-2Realization of a simulated ordered categorical response with 7 levels and a Bell-Shaped frequency distribution. Histogram is based on a single simulated Monte Carlo replication with 300 observations.

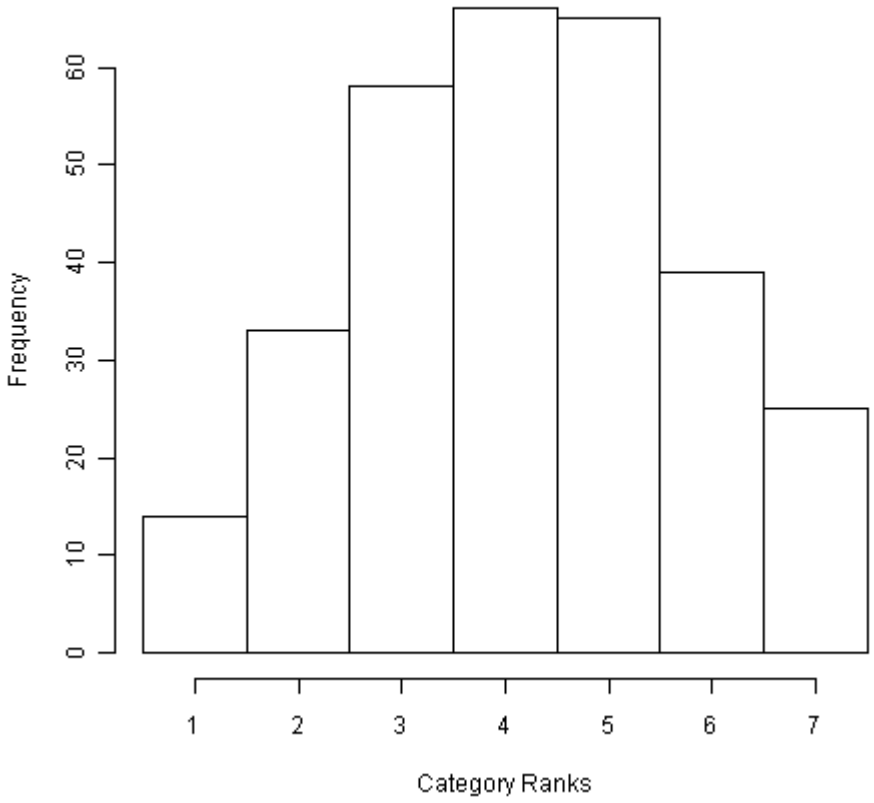


Figure 2-3 Realization of a simulated ordered categorical response with 7 levels and a Triangularly-Shaped frequency distribution. Histogram is based on a single simulated Monte Carlo replication with 300 observations.

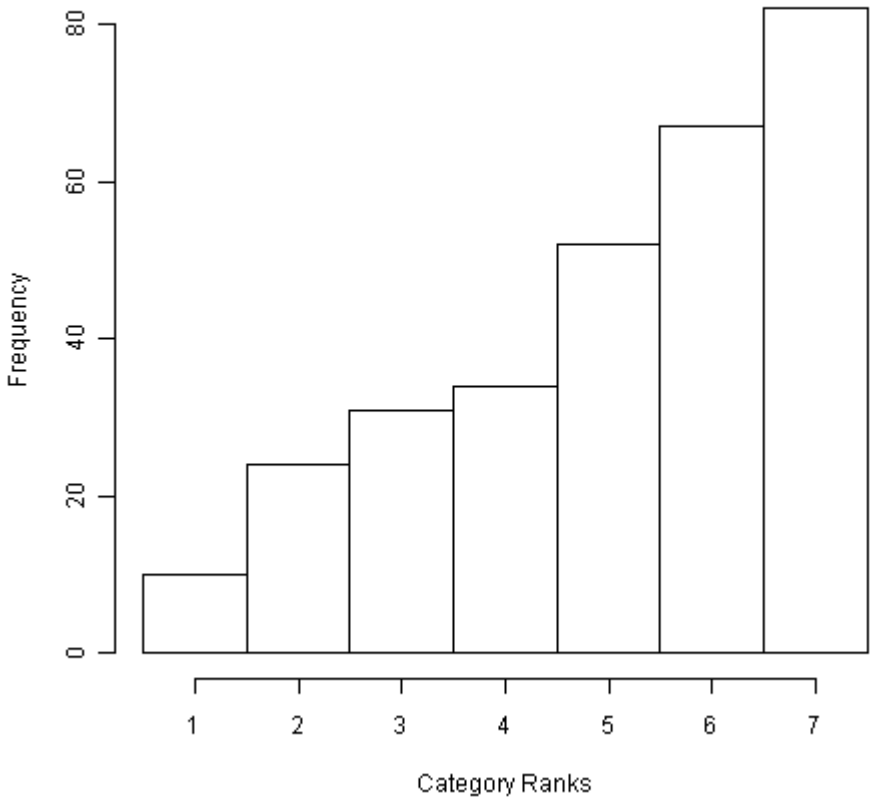
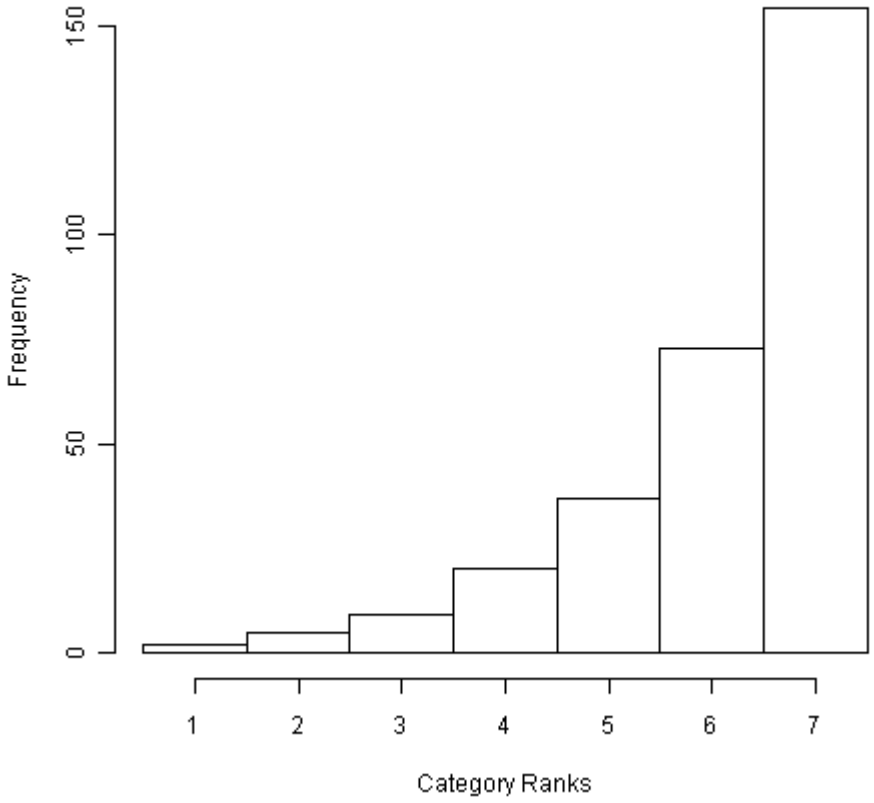


Figure 2-4 Realization of a simulated ordered categorical response with 7 levels and a Exponentially-Shaped frequency distribution. Histogram is based on a single simulated Monte Carlo replication with 300 observations.



2.3 Analysis

The following linear model was fit to each Monte Carlo replication:

$$y_i = \omega + \beta x_i + \varepsilon_i \qquad \varepsilon_i \sim i.i.d.N(0, \sigma^2)$$

where y_i is the observed level of the ordered categorical response on the i^{th} subject (i = 1,.. 300) , whereby the levels are enumerated from 1 to J, $\omega$and β are the intercept and slope parameters, respectively, and ε_i is the error associated with the i^{th} observation and is assumed to be $i.i.d.N(0, \sigma^2)$. I note that this model assumes that the enumeration of the realized ordered response categories is indicative not only of relative ranking of, but also of proportionality between consecutive levels of the ordered categorical response.

In each Monte Carlo replication, I use OLSLR to estimate β and then test the null hypothesis H_o) $\beta = 0$ based on a t test statistic with (n – 2) degrees of freedom where n is the sample size (n=300). The t test statistic is given by the formula:

$$t = \frac{b - \beta}{s(b)}$$

where b is the point estimate of β and $s(b)$ is the estimated standard error of b. The test statistic is then compared with a critical value given by the 1- α percentile of a t-distribution with n-2 degrees of freedom. For each Monte Carlo replication, I record whether the null hypothesis is either rejected or failed to reject at a level of significance given by the Type I error α = 0.05.

According to frequentist theory (Hogg, McKean, & Craig, 2005), probability is the long term relative frequency of a hypothetically repeatable event. Type I error, or α error, is the probability I conclude on a linear association between x and Y when this association does not exist. That is, Type I error (α) is the probability I incorrectly reject the null hypothesis (i.e. reject H_0 when it is true). In order to assess the robustness of OLSLR for inference on ordered categorical responses, empirical Type I errors (α) were computed based on the simulation study previously described. For a given scenario (i.e. combination of frequency distribution shape and number of ordered categories in the response) and $\boldsymbol{x}$ setting, the relative frequency of incorrect rejections of the null hypothesis $\beta = 0$ was computed by counting the number of Monte Carlo replications for which the null hypothesis was incorrectly rejected and dividing it by the total number of Monte Carlo replication that were analyzed for that scenario and $\boldsymbol{x}$ setting, namely 4,000). The relative frequency of incorrectly rejected null hypotheses was interpreted as an empirical Type I error rate (empirical α).

Type II error, or β error, is the probability one fails to reject the null hypothesis when it is false. Statistical power is then defined as (1 – Type II error); that is, the probability I reject the null hypothesis when the null hypothesis is false. In order to investigate the performance of OLSLR in regard to statistical power for inference on ordered categorical responses, empirical powers were computed based on the simulation study. For a given scenario and $\boldsymbol{x}$ setting, empirical power was computed as the proportion of Monte Carlo replicates (out of 4000) for which the null hypothesis $\beta = 0$ had been correctly rejected.

2.4 Results

2.4.1 Type I Error

Figures 2-5 and 2.6 illustrate empirical Type I error rates for all scenarios under the $\boldsymbol{x}^{(51)}$ or $\boldsymbol{x}^{(5)}$ setting, respectively. Across the frequency distributions shapes considered in this study, the empirical Type I error remained close to the nominal value of 0.05, regardless of the number of ranked levels of the ordered categorical response. The maximum and minimum empirical Type I error rates were 0.043 and 0.05475, corresponding to scenarios with 4-level ordered categorical variables generated in $\boldsymbol{x}^{(5)}$ with an exponentially-shaped frequency distribution and 7-level ordered categorical variables generated in $\boldsymbol{x}^{(51)}$ with a uniformly-shaped frequency distribution, respectively. In fact, the frequency of empirical Type I errors for all scenarios were within probabilistic expectation based on a binomial distribution with probability given by a nominal Type I error rate of 5% and size given by the number of Monte Carlo replicates for each scenario. Therefore, the observed deviations from the nominal Type I error of 0.05 may be attributed to random variability.

Figure 2-5 Empirical Type I error for inference on H_0) $\beta = 0$ based on ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels and Uniform, Bell, Triangular, or Exponentially shaped frequency distributions. Each scenario is represented by 4000 Monte Carlo replications. The figure represents settings where the explanatory covariate consisted of 51 levels ($x_i^{(51)}$) ranging from -50 to 50 in intervals of 2. Bounds on $\widehat{\alpha}$ correspond to 2.5th and 97.5th percentiles of a binomial distribution with probability of 5% and size = 4000, expressed as a proportion.

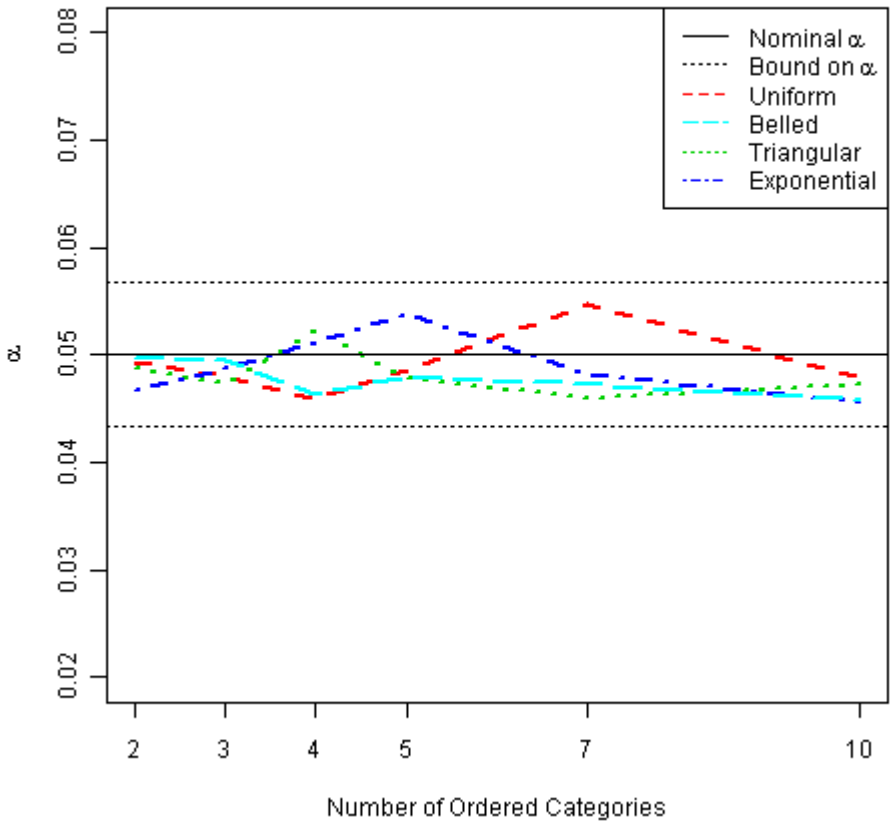
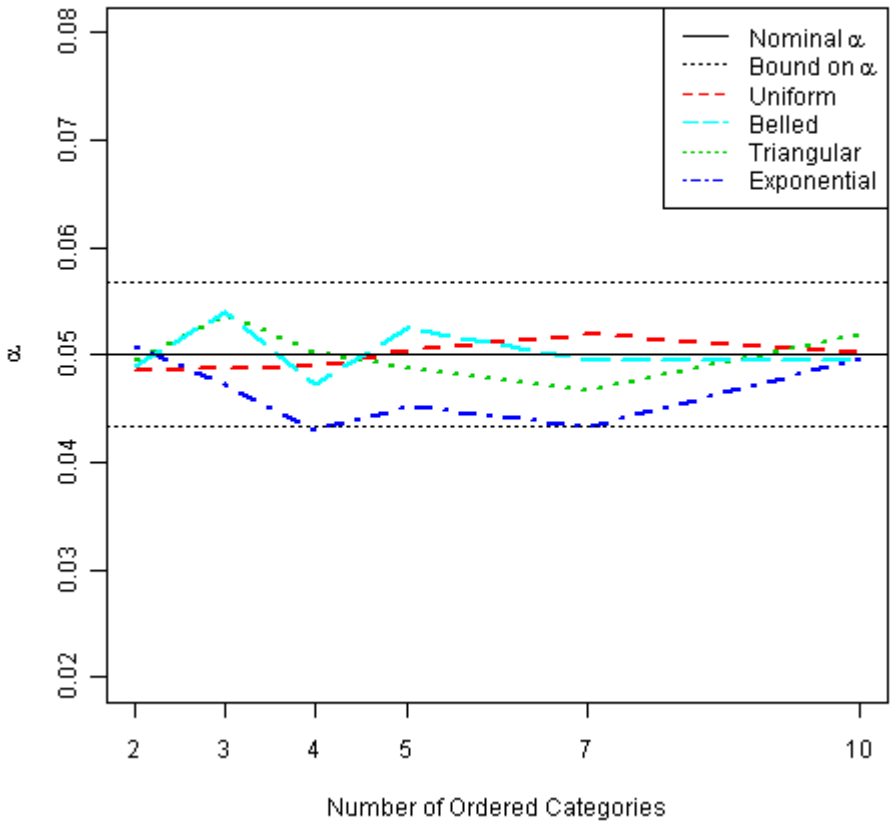


Figure 2-6 Empirical Type I error for inference on H_0) $\beta = 0$ based on ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels and Uniform, Bell, Triangular, or Exponentially shaped frequency distributions. Each scenario is represented by 4000 Monte Carlo replications. The figure represents settings where the explanatory covariate consisted of 5 levels ($x_i^{(5)}$) ranging from -50 to 50 in intervals of 25. Bounds on $\widehat{\alpha}$ correspond to 2.5th and 97.5th percentiles of a binomial distribution with probability of 5% and size = 4000, expressed as a proportion.



2.4.2 Statistical Power

Based on the simulation study, I compared statistical power for OLSLR-based inference on β fitted on the ordered categorical responses to statistical power for a statistically sound technique applied to the same data. Specifically, probit analyses were utilized. Statistical power of OLSLR fitted on the latent variable is also provided as a reference. Figures 2-7 to 2-10 illustrate empirical statistical power as a function of the number of categorical levels with uniform, belled, triangular or exponential frequency distributions of the ordered categorical response, respectively. Overall, across scenarios of distributional shapes and $\boldsymbol{x}$ settings, statistical power for OLSLR-based inference on β was weakest when the categorical response consisted of 2 levels. Minimal statistical power across distributional shapes ranged from 0.5615 for the scenario with an exponentially-shaped distribution and $\boldsymbol{x}^{(51)}$-setting to 0.617 for the scenario with a bell-shaped distribution and $\boldsymbol{x}^{(5)}$-setting. Statistical power overall ranged from 0.5615 to 0.7945. In general terms, as the number of levels of the ordered categorical response increased, so did empirical power. However, the rate of increase in empirical power decreased as the number of levels of the response increased and empirical power appeared to level off after a number of levels specific to each frequency distribution shape. For the uniform, bell, and triangularly shaped scenarios empirical power for both OLSLR and probit-link models analysis approached the 0.80 reference line as the number of levels of the response increased. Interestingly, for each $\boldsymbol{x}$-setting, the empirical power for OLSLR was comparable to that of probit-link models, especially for ordered categorical variables with uniformly-, bell- and triangular-shaped frequency distributions.

Empirical power for the scenario with an exponentially-shaped frequency distribution was the weakest across all numbers of levels of the ordered categorical response. Also, this shape scenario deviated from the general trend of increasing power as the number of the levels of the ordered categorical response increased. Empirical power for the exponentially-shaped scenario increased until 5 levels of ordered response, peaked at 0.69750 for the $\boldsymbol{x}^{(51)}$ setting and 0.69050 for the $\boldsymbol{x}^{(5)}$ setting, and then generally decreased beyond this point. The $\boldsymbol{x}^{(51)}$ scenario had more empirical power at 10 levels of ordinal response than at 7, but not as much empirical power as at 5 levels of ordinal response. This was a similar pattern to what was observed for probit analysis of these data. However, the probit analysis had more power at higher numbers of

ordered category than did OLSLR. For comparison, peak values for the probit also occurred at 5 levels of response and assumed the values 0.71200 and 0.70575, for $\boldsymbol{x}^{(51)}$ and $\boldsymbol{x}^{(5)}$, respectively.

Of particular interest was the comparison of statistical power between OLSLR-based inference and that based on the probit model across simulated scenarios. A normal approximation to the binomial distribution was used to construct confidence intervals for all differences in empirical power between the probit models and OLSLR, as well as between different distribution shape scenarios for OLSLR. A Bonferroni correction for multiple testing was used to control the family-wise Type I error rate at .05.

For scenarios comprising uniform, belled, and triangular-shaped frequency distributions and either $\boldsymbol{x}$ setting, regardless of the number levels of the ordered categorical response, there was no significant evidence for a difference in empirical power between probit models and OLSLR. Lack of evidence for a difference between OLSLR and probit model inference was also apparent for scenarios with exponentially shaped frequency distributions and 7 or less levels of the ordered categorical response. However, OLSLR-based empirical power of 10-level ordered categorical variables with an exponentially shaped frequency distribution was significantly less than that under a probit analysis.

I also compared OLSLR-based empirical power between scenarios comprising different shapes of frequency distribution, while keeping the $\boldsymbol{x}$-level setting and number of levels constant (Figures 2-11 and 2-12). Overall, empirical power based on OLSLR for ordered categorical responses with a belled, uniform or triangular frequency distribution were not significantly different from one another within a level of ordered categorical response and $\boldsymbol{x}$-level setting. The one exception to this is the comparison of the belled-shape and triangular-shape comparison for the 4-level ordered categorical variable. The difference in statistical power between was significantly different from zero. While the uniform, belled, and triangular shape scenarios had comparable powers for most levels of the response, the exponential shape scenario became statistically different from the other shape scenarios after 4 levels of response.

In summary, empirical power based OLSLR fitted to a categorical response with uniform, bell, and triangular shapes of frequency distributions were comparable to each other and to the empirical power from a probit analysis of the same data. The more categories of ordered categorical response, the greater the empirical power of OLSLR but the weaker that power in relation to the probit power. The ordered categorical variables with an exponentially shaped

distribution had the weakest power relative to the other shape scenarios, and this weakness became statistically significant as the number of response levels increased.

Figure 2-7 Empirical statistical power for correctly rejecting H_0) $\beta = 0$ based on a probit or ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels with a Uniformly-Shaped frequency distribution. Slope on the latent variable used to generate the data is 1. The figure represents settings where the explanatory covariate consisted of either 51 levels ($x^{(51)}$) ranging from -50 to 50 in intervals of 2 or 5 levels ($x^{(5)}$) ranging from -50 to 50 in intervals of 25.

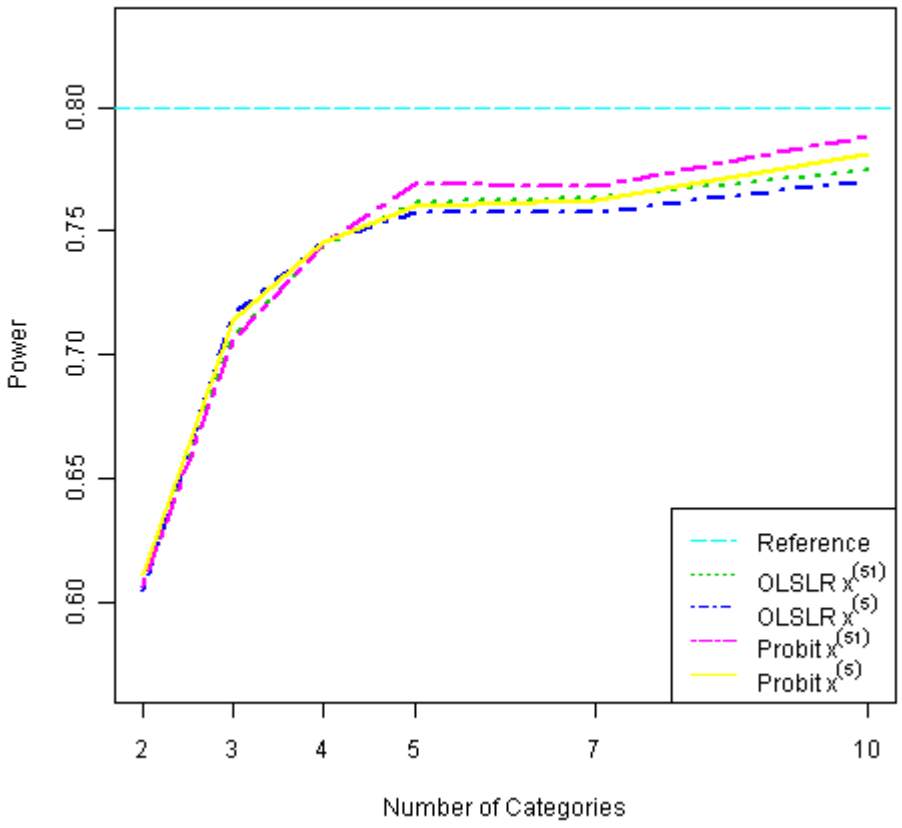


Figure 2-8 Empirical statistical power for correctly rejecting H_0) $\beta = 0$ based on a probit or ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels with a Bell-Shaped frequency distribution. Slope on the latent variable used to generate the data is 1. The figure represents settings where the explanatory covariate consisted of either 51 levels ($x^{(51)}$) ranging from -50 to 50 in intervals of 2 or 5 levels ($x^{(5)}$) ranging from -50 to 50 in intervals of 25.

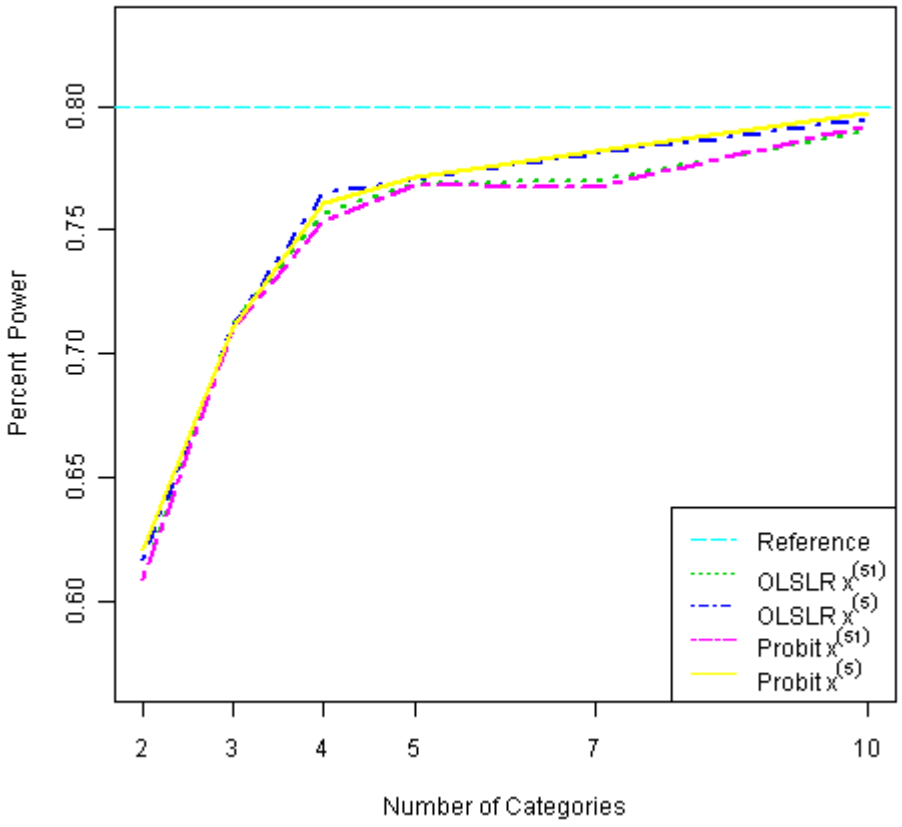


Figure 2-9 Empirical statistical power for correctly rejecting H_0) $\beta = 0$ based on a probit or ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels with a Triangularly-Shaped frequency distribution. Slope on the latent variable used to generate the data is 1. The figure represents settings where the explanatory covariate consisted of either 51 levels ($x^{(51)}$) ranging from -50 to 50 in intervals of 2 or 5 levels ($x^{(5)}$) ranging from -50 to 50 in intervals of 25.

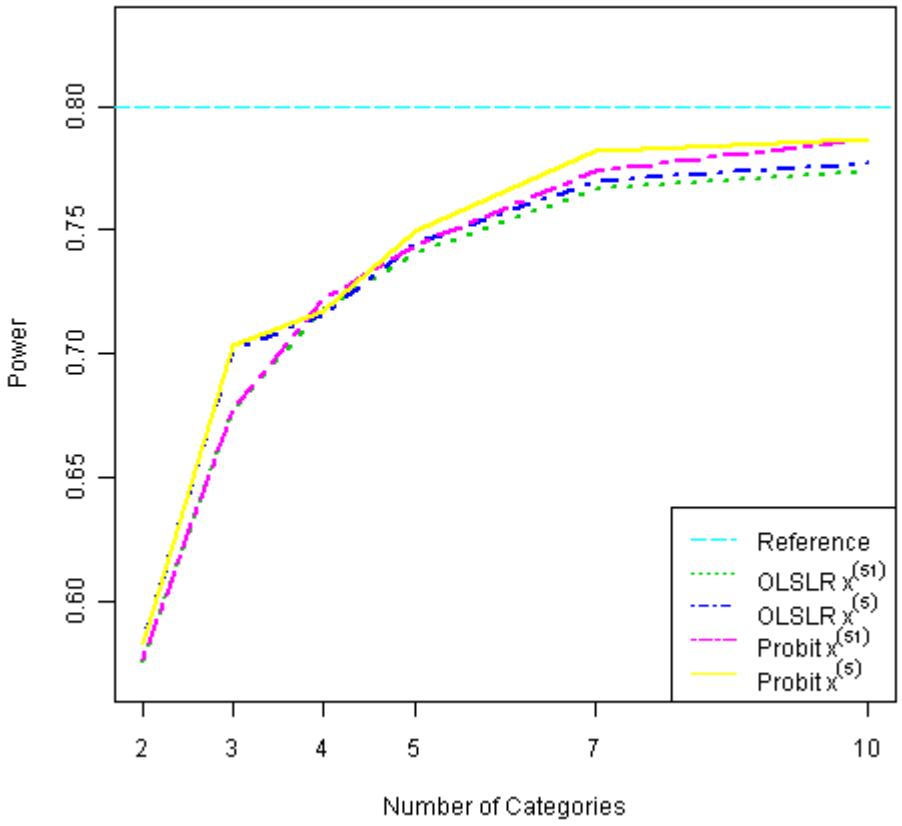


Figure 2-10 Empirical statistical power for correctly rejecting H_0) $\beta = 0$ based on a probit or ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels with an Exponentially-Shaped frequency distribution. Slope on the latent variable used to generate the data is 1. The figure represents settings where the explanatory covariate consisted of either 51 levels ($x^{(51)}$) ranging from -50 to 50 in intervals of 2 or 5 levels ($x^{(5)}$) ranging from -50 to 50 in intervals of 25.

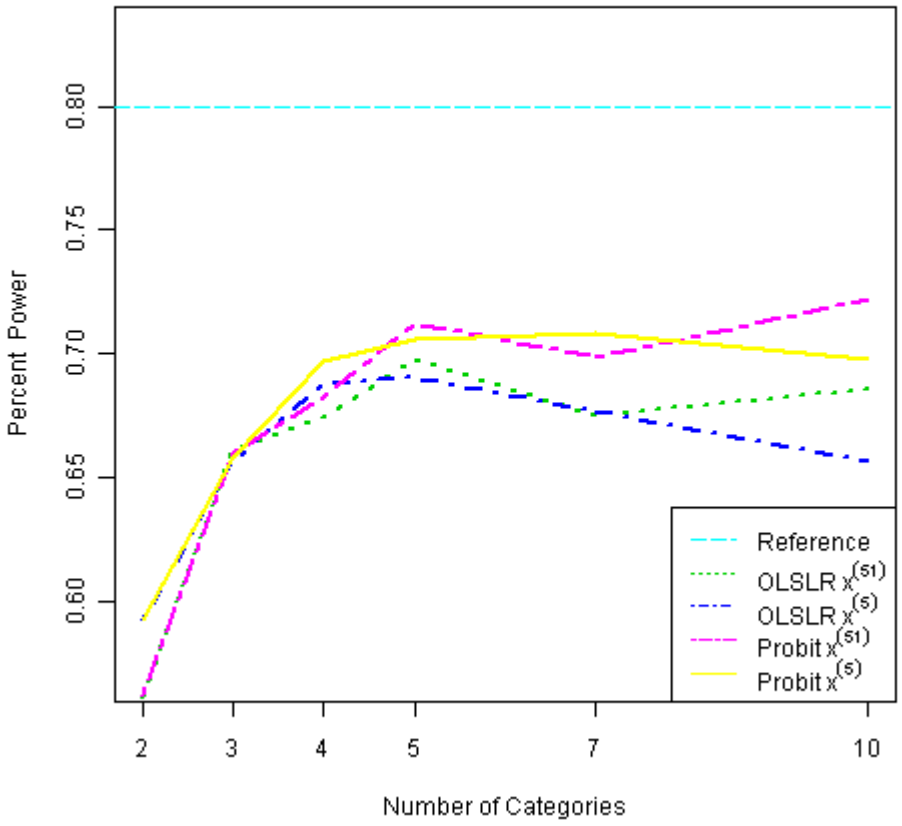


Figure 2-11 Empirical Type statistical power for inference on H_0) $\beta = 0$ based on ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels and Uniform, Bell, Triangle, or Exponentially shaped frequency distributions. Ordered categorical variables generated from a latent variable with slope of 1. Each scenario is represented by 4000 Monte Carlo replications. The figure represents settings where the explanatory covariate consisted of 51 levels ($x_i^{(51)}$) ranging from -50 to 50 in intervals of 2.

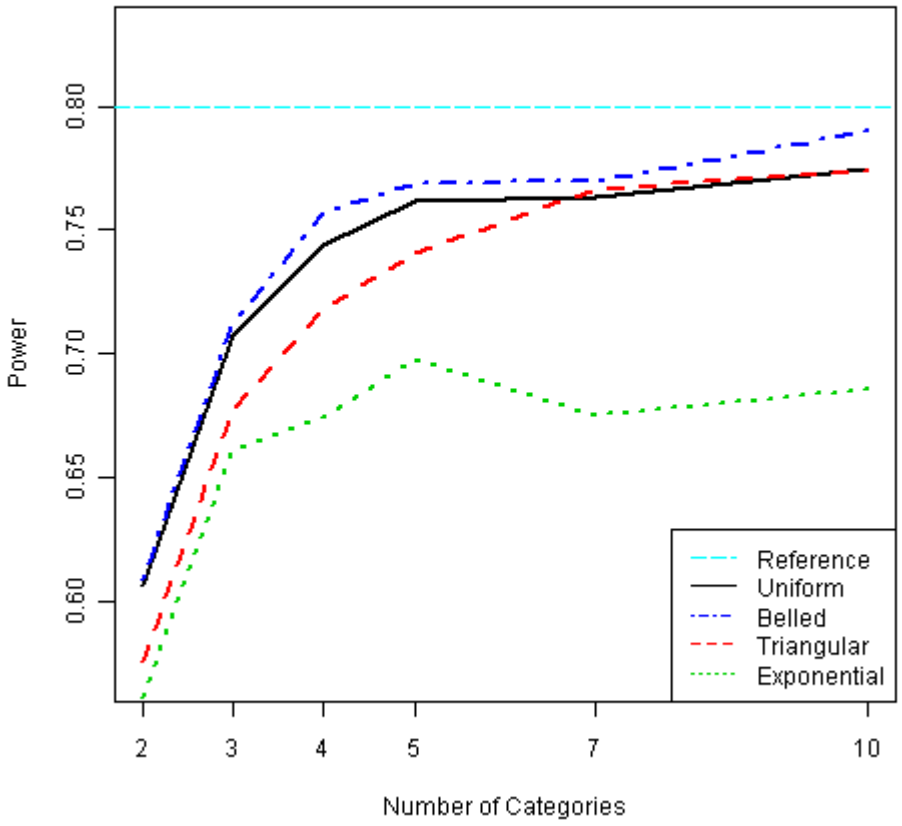
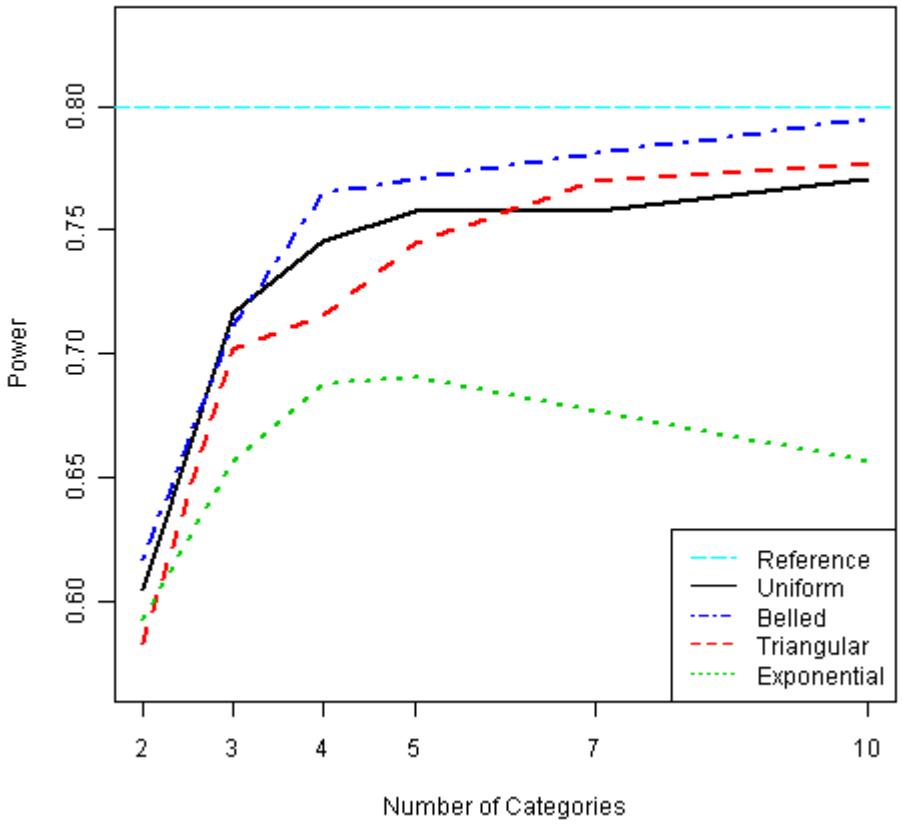


Figure 2-12 Empirical Type statistical power for inference on H_0) $\beta = 0$ based on ordinary least squares linear regression fitted to an ordered categorical response characterized by increasing number of levels and Uniform, Bell, Triangle, or Exponentially shaped frequency distributions. Ordered categorical variables generated from a latent variable with slope of 1. Each scenario is represented by 4000 Monte Carlo replications. The figure represents settings where the explanatory covariate consisted of 5 levels ($x_i^{(5)}$) ranging from -50 to 50 in intervals of 25.



2.4.3 Inferences on Slope

Following up the evaluation of inferential properties of OLSLR as applied to ordered categorical data, one may inquire as to the interpretation of the parameter estimate b in the context of the research problem. To address this question, I collected point estimates b for all 4000 Monte Carlo replications in a given scenario and used them to describe the empirical sampling distributions of b. Figure 2-13 and 2-14 illustrate the empirical sampling distribution of b for 2 representative scenarios. When the ordered categorical responses were generated based on $\beta^* = 0$, the empirical sampling distributions of b were observed to be bell-shaped and centered about zero for all simulated scenarios. In fact, in all cases, the lower and upper quartiles of the sampling distributions of b had negative and positive signs, respectively. In contrast, when the ordered categorical response was generated based on $\beta^* = 1$ in the latent scale, the sampling distributions of b remained bell-shaped but none contained the value 1. To illustrate, the lower quartile and the upper quartile of the empirical sampling distribution of b were -0.001135 to 0.001111 for the ordered categorical variables generated with $\beta^* = 0$ in setting $\boldsymbol{x}^{(51)}$ that had a uniformly-shaped frequency distribution and 2 levels of response. They were 0.002966 to 0.005166 for the ordered categorical variables generated in the equivalent condition but with $\beta^* = 1$. Additional examples of the sampling distribution of b for other arbitrarily selected scenarios may be seen in Appendix A.

It is worth noticing that the magnitude of the OLSLR estimate b of the slope parameter β is not directly comparable to the slope parameter in the latent scale β^*, which was used to generate the data. This may be partially explained by the difference in scales, namely the latent continuous scale of the data generation process versus the ordered categorical scale used for inference. In addition, it may be noted that the OLSLR assumption of proportionality across levels of the ordered categorical response was not necessarily supported by the data generation process, which further impairs meaningful interpretation of the OLSLR slope parameter b as a rate of change. Note that if the distance between consecutive ranks were meaningful, that is, if the order of categories reflected the true spacing of the construct up to some proportionality constant, then it might be possible to use the OLSLR-based slope estimate b to approximate β^* up to the proportionality constant.

Figure 2-13 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 2 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

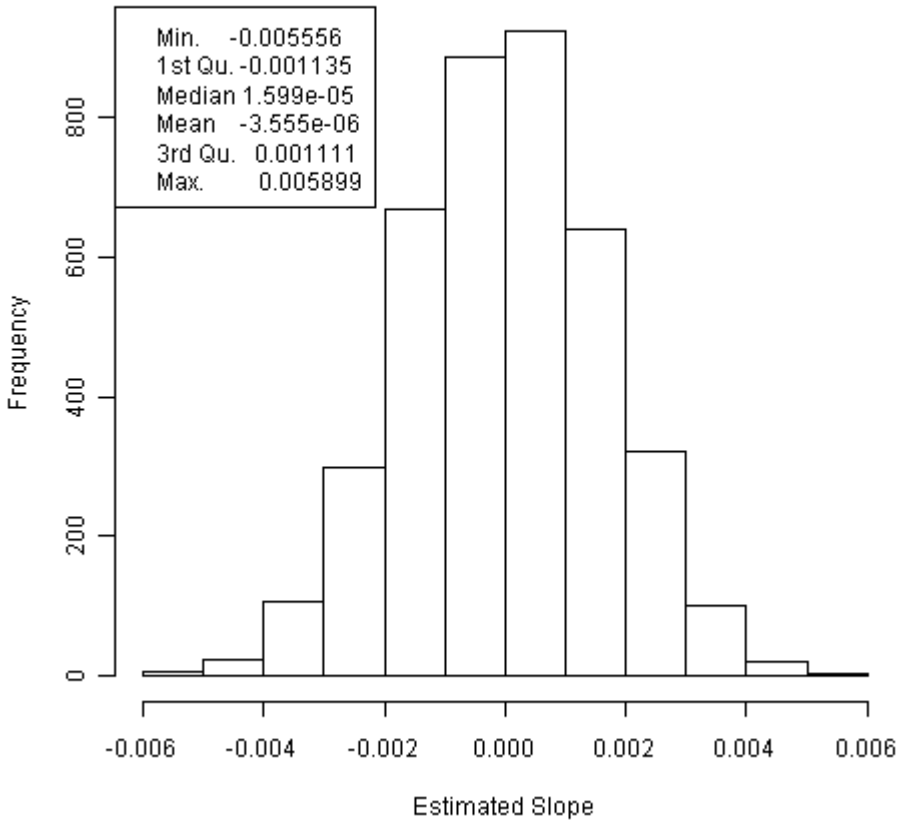
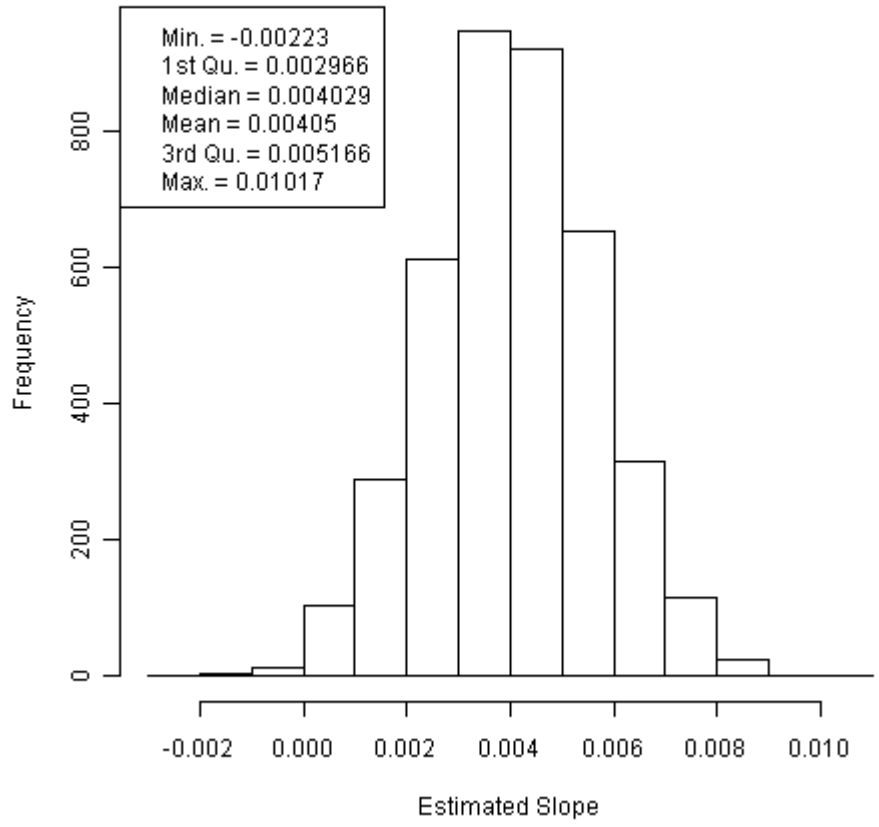


Figure 2-14 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 2 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.



2.5 Case Study Based on Real Data

To demonstrate the application of OLSLR to a real-world problem, data used by McIntosh et al. (2009) were analyzed. In that study, the researchers evaluated use of antimicrobials in feedlot cattle as it relates to Icek Ajzen's theory of planned behavior (Ajzen, n.d.). This theory states that behavioral beliefs, normative beliefs, and belief about perceived behavioral control come together to form an intention that will then result in an action (Ajzen, n.d.). The action of interest for that study was frequency of antimicrobial use in feedlot cattle for various groups of bovine. For this case study, I will discuss data from only one of these groups of cattle and for feedlot operators as well as veterinarians. McIntosh et al. (2009) wished to know how frequency of antimicrobial in feedlot cattle changed in relation to behavioral, normative and control beliefs. They used these beliefs to motivate questions for a 20 page survey that was distributed to veterinarians and feedlot operators in the Midwestern US. The individuals surveyed were asked to answer questions about their motivating beliefs and to rate their frequency of antimicrobial use as denoted by the categories "never", "rarely", "sometimes", "often" and "always." These response categories were labeled 1 to 5, respectively, to indicate ranking of increasing frequency of use.

The research question was addressed by fitting theoretically sound statistical models to the data; in particular, I chose to work with a probit regression model. I also choose to work with the larger data set from which the McIntosh research group's data came. This larger data set included the 269 feedlot operators and veterinarians that returned the survey, instead of just the 103 veterinarians used by McIntosh et al., and was provided by Dr. H. M. Scott (H.M. Scott, personal communication, June, 2009). Probabilities of frequency of antimicrobial use categories were modeled to a continuous covariate measuring an aspect of behavioral beliefs from the theory of planned behavior. The assumptions associated with this model were checked and considered to be reasonably satisfied. The estimated model coefficient of 0.6076 for the predictor was significant ($p < .001$) and had a standard error of 0.0739. This estimated coefficient tells us that for a one unit increase in the covariate, the z-score increases by 0.6076. The category never was set to baseline. The intercepts for "always", "often", "some of the time", and "rarely" were -1.1738, 0.2573, 1.3349 and 3.1811, respectively. One may use the estimated coefficient in combination with the various intercepts to compute a z–score. This z-score may be used to determine predicted probabilities from the standard normal distribution. For example,

the predicted probability of always using antimicrobials, when the covariate has been set to one standard deviation beneath its mean, at -0.999999999, was 0.0374. The predicted probability of always using antimicrobials when the covariate is set to its mean (.0000000009948218) was 0.1202. If instead I set the covariate to one standard deviation above its mean, at 1.0000000009948218, the predicted probability of always using antimicrobials is 0.2856.

A more complete model was also constructed using model selection procedures with all possible covariates. More specifically, backward, forward and stepwise selection, along with a model building process guided by the theory of planned behavior were all used to generate a set of final models. The final model suggested by these techniques agreed and was utilized as the final model. This final model included the above covariate from the single variable model in addition to a continuous explanatory variable for an aspect of perceived behavioral control and a 5-level categorical explanatory for an aspect of normative beliefs. This final model suggested that as the belief that use of antibiotic for certain cattle would be both better for the cattle and would be profitable (combined with valuing healthier cattle and having more funds) increased, the use of antibiotics in these cattle would increase (behavioral beliefs, $p < .0001$). This model also suggested that as the belief that treating these cattle was necessary increased the greater the frequency of antimicrobial use in these cattle (perceived behavioral control, $p < .0001$). Finally, it was suggested that as those close to a person had greater expectations for cattle to be treated (combined with how important those expectations were) the greater the use of antibiotics (normative belief, $p < .0324$).

McIntosh et al. (2009) used OLSLR to analyze their subset of the data. I chose to follow suit with the larger data set. Ordinary least squares regression was employed to model the rank associated with the frequency of antimicrobial use categories on the behavioral belief predictor. This predictor had a significant coefficient of -0.447450369 ($p < .001$) with a standard error of 0.04874177. This coefficient is telling us that for a unit increase in the covariate, I decrease our category by 0.447450369. Note that as categories are discrete, one may not move by a fraction of a category. Also notice that the coefficients from the OLSLR model and the probit model have unrelated meanings and that they assume different values. This suggests that an interpretation of slope from OLSLR is something that may not be sensible.

The remaining variables from the theoretically sound model were then included into an OLSLR model for purposes of comparison. Ordinary least squares linear regression suggested

the same conclusions as the theoretically sound model in regard to significance of the coefficients for the aspects of behavioral belief ($p < .0001$), perceived behavioral control ($p < .0001$) and normative beliefs ($p < .0409$).

2.6 Discussion

In this study, I evaluated the inferential implications of analyzing ordered categorical data with OLSLR. This was done for ordered categorical data that had 1 of 4 frequency distribution shapes, 2, 3, 4, 5, 7, or 10 categories of the response, and in 2 $\boldsymbol{x}$ settings ($\boldsymbol{x}^{(51)}$ and $\boldsymbol{x}^{(5)}$). It was found that OLSLR was generally robust to violations of normality. Arnold (1980) found that ANOVA was robust to violations of normality in the response variable if the number of categories of the discrete predictor was low but the number in each category was high. Linear modeling with continuous covariates may be thought of as having infinitely many ordered categories on the x-axis where the number in each category is at most one and the distances between categories is meaningful. This thought process combined with Arnold's results imply that OLSLR applied to ordered categorical data might be especially problematic when employed with continuous covariates. While the $\boldsymbol{x}^{(51)}$ covariate in our study was only nearly continuous, the analyses in that setting did not suggest that having a continuous covariate would present any special burden in modeling ordered categorical responses with OLSLR.

In fact, for Type I error, there did not appear to be any differences between $\boldsymbol{x}$-settings. Type I error was within the range that I would expect by chance for all except the 4-level exponential should the true α really have been .05 in all scenarios. Hsu and Feldt (1969) found the robustness of ANOVA in relation to Type I error increased as the number of categories on the response increased with the odd numbers of levels having superior performance. This agrees with what I found in that Type I error was robust and that there was a scenario with an even number of categories of ordinal response that behaved differently. It should be noted that both Hsu and Feldt (1969) and I only used 2 and 4 categories for our even number of response categories. It would be interesting to investigate whether our observed difference in performance for even and odd numbers of response category is coincidental or the manifestation of some underlying, yet to be understood, process. If this significant α was reflective of some underlying process not wholly attributed to even and odd numbers of outcomes, the fact that the exponential distribution was the most extreme in skew and kurtosis may contribute to the explanation.

In examining frequency distribution shapes, it is noted that as the number of levels of the ordered categorical response decreases, the impact of skew and kurtosis on the shape of the frequency distribution can be expected to be attenuated likely due to a more rectangular shape supported by fewer levels. In contrast, as the number of levels of the ordered categorical response increase, the frequency distribution shape can be expected to reflect skew and kurtosis in more pronounced non-rectangular shapes granted by the larger number of levels. In the leptokurtic scenarios considered in this study, namely exponentially shaped frequency distribution, I did not see departures of the empirical Type I error from its nominal value regardless of the numbers of levels of the response. This may be interpreted as considerable robustness of OLSLR-based inference on ordered categorical responses to skew and kurtosis, at least in the scenario considered herein. Indeed, Reed and Stark (1995) have suggested that the sampling distribution of the responses may have little effect on the robustness of ANOVA. Glass, Peckman, and Sanders (1972) also stated that the skew of a distribution may have little effect on Type I error. They have also claimed that leptokurtosis may decrease empirical α, but that platykurtosis may have increase α. For the platykurtotic scenario (uniformly-shaped frequency distribution), the evidence did not support departures of the empirical alpha from its nominal value. In general, my findings for OLSLR-based inference on ordered categorical responses were generally consistent with those for ANOVA.

Despite pervasive robustness for Type I errors across scenarios considered in this study, results on empirical power were mixed. Scenarios with frequency distributions of uniform, belled and triangular shape performed comparably to one another in terms of empirical statistical power, whereby power showed an overall increase with number of levels of the ordered categorical responses. The exponential shape scenario is the only shape scenario that decreased power after 5 categories of ordered categorical response.

Glass, Peckman, and Sanders (1972) have claimed that leptokurtosis may have a positive impact on power, but that platykurtosis may have a detrimental impact on it. This was not supported for OLSLR. Instead, I found that the less skewed and less kurtotic the frequency distribution was, the greater the power. For instance, the bell-shaped distribution (no skew, no excess kurtosis) tended to perform the best and yielded the greatest power. The scenarios with a triangular (moderate skew) or uniform (moderate platykurtosis) shape were not significantly different from the bell scenario. The exponential scenario (high skew and high kurtosis) tended

to have the worst power and truly distinguished itself as the poorest after 5 levels on the ordered categorical response. As was mentioned previously, the frequency distribution shapes are more similar when the response has few levels and differences in shape only become apparent with increasing number of levels; this may be a partial explanation for the increasingly poor performance of the exponential scenario as the number of categories of the response increase.

Response style and wording of a scale can impact the distributional shape of the response (Javaras & Ripley, 2007). It is possible there may be scenarios, such as with the analysis of scales from politically charged topics, where individuals choose to avoid stigma and ostracization by choosing non-polarized scale values. Such an answering scheme would result in a highly leptokurtotic response distribution. Additionally, many attitude scales produce responses that are skewed (Abreu et al., 2008; Javaras & Ripley, 2007). It would be interesting to investigate the performance of OLSLR in relation to power for frequency distributions that are only leptokurtic and not skewed and for distributions with more moderate combinations of skew and kurtosis as to better address these practical applications. Without this investigation, it remains unknown whether or OLSLR may reasonably be implemented.

It should be noted that the interpretation of slope is only meaningful when the distances between the ranks is proportionate to the distances between the categories. When this is true, it may be possible to use the OLSLR estimate of slope to learn the change in x required to transition from one response category to the next.

2.7 Future Directions

While normal theory techniques appear generally robust, there remains room for further inquiry. For example, this research found promising results for one non-zero level of slope and variance combination s per 1 of 2 $\boldsymbol{x}$ settings with variances such that latent power was approximately 0.80. It would be interesting to see how OLSLR would perform for different combinations of slope and variance. It is reasonable to assume that the proportions used here may not be representative of what will be encountered in practice. It may be speculated that OLSLR may not perform as well for data generated from models that have more variation with the same slope, from models that have steeper slope with more variation or from models with less slope and more variation. It may be that OLSLR is even more robust to violations of normality for data generated from models that have greater slope with similar variance, similar

slope but less variance, or a very steep slope and very little variation. Further elucidation of OLSLR functionality could be determined by exploration of assorted slope and variance combinations.

The number of x settings considered should be expanded upon. I considered 1 range of x, with either 5 or 51 possible values and built models with a single covariate per model. It would be informative to consider both wider and narrower ranges of x. The distances of the x values to their center are used in the computation of b. As I am using b to determine whether or not β is equal to zero, the x distances that might feasibly be encountered in practice become relevant. It would also be useful to consider including more than 1 covariate with multiple combinations of continuous and categorical predictors as well as interaction terms. The case study provided is an example of using multiple covariates, and it suggests that OLSLR for such models might be reasonable to fit. Russell and Boboko (1992) have mentioned the need for investigation into interaction models for ordered categorical responses analyzed with OLSLR. Another consideration in relation to $\boldsymbol{x}$ is that our models only considered a first order term. I did not investigate the inferential implications of fitting OLSLR for ordered categorical responses when the underlying latent variable included quadratic, or higher, order effects. These realms of inquiry remain open to investigation.

Finally, I have only considered the case where I have a single outcome variable free from any complicated structure. It would be interesting to see how robust OLSLR is when applied to data with more than one response variable as it may be desired to see how the responses behave when taken together. For example, with the case study example, there were multiple responses not considered. I focused on the behavior of feedlot operators and veterinarians in relation to their use of antimicrobials as it pertains to en masse treatment. These individuals were also questioned about their behavior for acutely ill cattle, for chronically ill cattle, and for subtheraputic use in at-risk cattle (H. M. Scott, personal communication, June, 2009). It would be useful to know if employing OLSLR for this more complicated scenario is a reasonable thing to do.

It would also be useful to investigate scenarios where maximum likelihood estimation, which is equivalent to least squares techniques when normality is achieved, is useful. For example, I may wish to what the inferential implications are for utilizing procedures for analyzing linear mixed models when our dependent variable is an ordered categorical response

measured over time instead of a continuous variable. This situation may occur in agriculture when measuring responses such as severity of lesions, condition of an udder, or some sort of other scaled biological measure at multiple times throughout the course of a study.

2.8 Summary

This research investigated the inferential implications of using OLSLR on ordered categorical responses. It appears that OLSLR may be a reasonable choice for investigating the hypothesis of $\beta = 0$, even when the response variables are ordered categorical and violate the assumption of normality. It was found that Type I error remained reasonable across all shapes of frequency distribution, $\boldsymbol{x}$-setting, and number of levels of response category scenarios considered. Power of OLSLR on the ordered categorical scale remains comparable to that of a probit analysis, especially for non-exponentially shaped scenarios. In general, the exponential shape scenario had the weakest power whereas the remaining shape scenarios were comparable to one another and approached that of OLSLR on the latent scale as the number of outcome categories increased.

While determining if a nonzero slope exists seems like something that can be accomplished with OLSLR on ordered categorical data, the slope itself should not be something considered of interest. The estimate of slope on the ordered categorical scale is not meaningful because the rankings of the ordered categorical variables do not themselves grant us any information about the distances between them. This means the magnitude of the slope estimate is a reflection of our choice of distances between the ranks.

A data analysis with non-simulated data was done to demonstrate OLSLR in practical application. The ordinary least squares linear regression analysis of a 5-level ordered categorical variable agreed with a theoretically sound analysis in terms of the significance of the covariates under inspection for both the single covariate model and the 3 covariate model. The predictor in the single covariate model was a continuous variable. The model with 3 covariates had 2 continuous predictors and 1 5-level categorical predictor.

2.9 Final Remarks

While OLSLR appears robust to violations of normality, it is advised that methods with sound statistical foundations be employed, as there are additional advantages for using

theoretically appropriate techniques. For example, with the probit analysis, a practitioner may comment on the standardized slope of the underlying latent variable. The disciplinary scientist may form odds ratios and get the probability of being in a certain category with the proportional odds logistic regression. Furthermore, the robustness of OLSLR outside of the scenarios seen in this study is still not known. Given the advantages of theoretically sound techniques combined with the uncertainty that remains in regard to OLSLR of ordered categorical responses, it is advised that OLSLR be considered as an alternative for inference on ordered categorical responses only in certain situations. Specifically, that it only be employed where statistically sound techniques become impossible to implement due to technical complications such as data sparsity.

References

Abreu, M. N. S., Siqueira, A. L., Cardoso, C. S., & Caiaffa, W. T. (2008). Ordered categorical logistic regression models: Application in quality of life studies. *Cadernos De Saúde Pública, 24*, s581-s591.

Agresti, A. (1990). *Categorical data analysis*. New York: Wiley.

Ajzen, Icek. (n.d.). *Theory of planned behavior* Retrieved 3/21/2011, 2011, from http://www.people.umass.edu/aizen/tpb.html

Alali, W. Q, Scott, H. M., Harvey, R. B., Norby, B. Lawhorn, D. B. and Pillai, S. D. (2008). Longitudinal Study of Antimicrobial Resistance among Escherichia coli Isolates from Integrated Multisite Cohorts of Human and Swine. *Applied and Environmental Microbiology, 74(12),* 3672.

Arnold, S. F. (1980). Asymptotic validity of F tests for the ordinary linear model and the multiple correlation model. *Journal of the American Statistical Association, 75*(372), pp. 890-894.

Arostegui, I., Nunez-Anton, V., & Quintana, J. M. (2010). Statistical approaches to analyse patient-reported outcomes as response variables: An application to health-related quality of life *Statistical Methods in Medical Research,* doi:10.1177/0962280210379079

Boneau, C. A. (1960). The effects of violations of assumptions underlying the t test. *Psychological Bulletin, 57*(1), 49-64. doi:DOI: 10.1037/h0041412

De Groote, H., Rutto, E., Odhiambo, G., Kanampiu, F., Khan, Z., Coe, R., & Vanlauwe, B. (2010). Participatory evaluation of integrated pest and soil fertility management options using ordered categorical data analysis. *Agricultural Systems, 103*(5), 233-244. doi:10.1016/j.agsy.2009.12.005

Gayen, A. K. (1949). The distribution of `Student's t in random samples of any size drawn from non-normal universes. *Biometrika, 36*(3/4), pp. 353-369.

Glass, G. V., Peckham, P. D., & Sanders, J. R. (1972). Consequences of failure to meet assumptions underlying the fixed effects analyses of variance and covariance. *Review of Educational Research, 42*(3), 237.

Hedeker, D., & Gibbons, R. D. (1994). A random-effects ordered categorical regression model for multilevel analysis. *Biometrics, 50*(4), pp. 933-944.

Heeren, T., & D'Agostino, R. (1987). Robustness of the two independent samples t-test when applied to ordered categorical scaled data. *Statistics in Medicine, 6*(1), 79-90. doi:10.1002/sim.4780060110

Hernandez, P., Guerrero, L., Ramirez, J., Mekkawy, W., Pla, M., Arino, B., & Blasco, A. (2005). A bayesian approach to the effect of selection for growth rate on sensory meat quality of rabbit. *Meat Science, 69*(1), 123-127. doi:10.1016/j.meatsci.2004.06.013

Hsu, T., & Feldt, L. S. (1969). The effect of limitations on the number of criterion score values on the significance level of the F-test. *American Educational Research Journal, 6*(4), pp. 515-527.

Javaras, K. N., & Ripley, B. D. (2007). An "Unfolding" latent variable model for likert attitude data. *Journal of the American Statistical Association, 102*(478), 454-463. doi:10.1198/016214506000000960

Johnson, V. E. (2008). Statistical analysis of the National Institute of Health peer review system. *Proceedings of the National Academy of Sciences, 105,* 11076–11080.

Kutner, M., Nachtsheim, C. J., Neter, J., & Li, W. (2005). *Applied linear statistical models.* Boston: McGraw-Hill Irwin.

Liu, I., & Agresti, A. (2005). The analysis of ordered categorical data: An overview and a survey of recent developments. *Test, 14*(1), 1-73.

McCullagh, P. (1980). Regression models for ordered categorical data. *Journal of the Royal Statistical Society.Series B (Methodological), 42*(2), pp. 109-142.

McIntosh, W. M. A., Schulz, S., Dean, W., Scott, M. H., Barling, K. S., & Takei, I. (2009). Feedlot veterinarians' moral and instrumental beliefs regarding antimicrobial use in feedlot cattle. *Journal of Community & Applied Social Psychology, 19*(1), 51-67. doi:10.1002/casp.976

R Development Core Team (2009). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Raubertas, R., & Shook, G. (1982). Relationship Between Lactation Measures of Somatic-Cell Concentration and Milk-Yield. *Journal of Dairy Science, 65*(3), 419.

Reed, J. F. & Stark, D. B. (1995). Robust analysis of variance: a simulation study -- Robust analysis of variance. *Journal of Applied Statistics*, *22*(1), 87-104. doi:10.1080/757584400

Rocha, J. L., E. J. Eisen, F. Seiwerdt, L. D. V. Vleck and D. Pomp. (2004). A large-sample QTL study in mice: III. *Reproduction. Mamm. Genom*e, *15*, 878–886.

Russell, C. J., & Bobko, P. (1992). Moderated regression analysis and likert scales: Too coarse for comfort. *Journal of Applied Psychology, 77*(3), 336.

Snell, E. J.(1964). A Scaling Procedure for Ordered Categorical Data. *Biometrics*, 20(3), 592-607.

Taylor, T. K., Burns, G. L., Rusby, J. C. & Foster, E. M. (2006). Oppositional Defiant Disorder toward Adults and Oppositional Defiant Disordre toward Peers: Initial Evidence for Two SeparateConstructs. *Psychological Assessment, 18(4),* 439.

U.S. Energy Information Registration. (n.d.). *EIA-voluntary reporting of greenhouse gases program - original 1605(b) program* Retrieved 9/24/2010, 2010, from http://www.eia.doe.gov/oiaf/1605/FAQ_EmissionRed_MethodA.htm

Winship, C., & Mare, R. D. (1984). Regression models with ordered categorical variables. *American Sociological Review, 49*(4), pp. 512-525.

Xu W. H., Dai, Q., Xiang, Y. B., Zhao, G. M., Ruan, Z. X., & Cheng, J.R. (2007). Nutritional factors in relation to endometrial cancer: a report from a population-based case-control study in Shanghai, China. *International Journal of Cancer, 120*, 1776-1781.

Yi, N., Samprit, B., Pomp, D., & Yandell, B. S. (2007). Bayesian Mapping of Genomewide Interacting Trait Loci for Ordered categorical Traits. *Genetics, 176*(3), 1855.

Appendix A - Appendix A: Empirical Distributions of Slope

Figure A-1 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

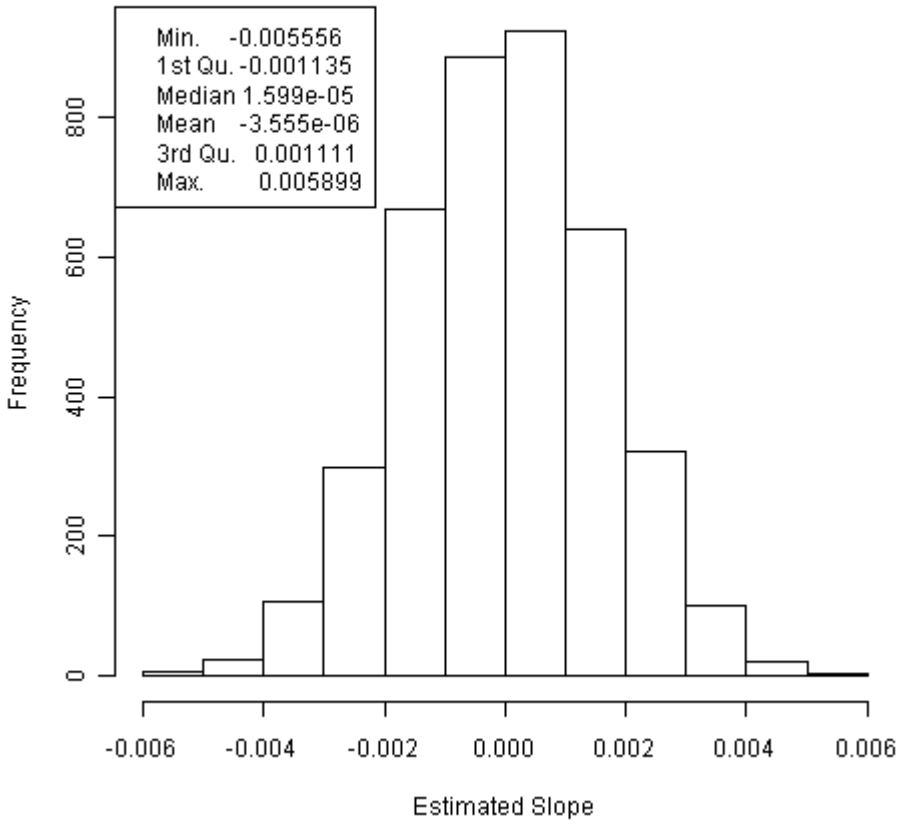


Figure A-2 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Bell-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

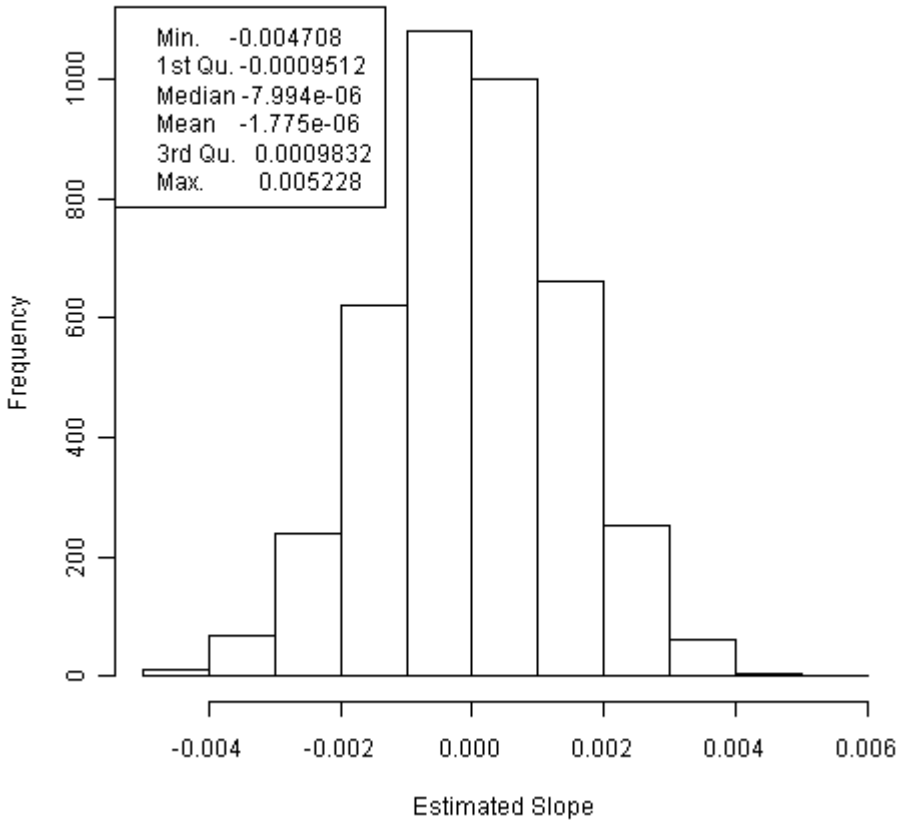


Figure A-3 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Triangularly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

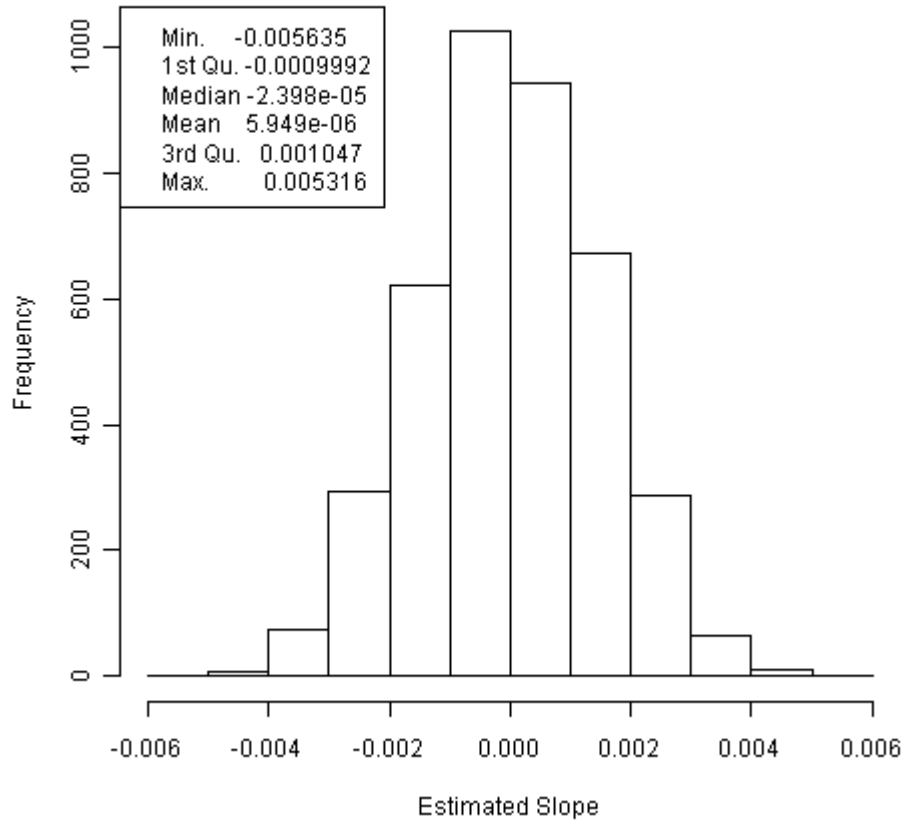


Figure A-4 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and an Exponentially-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

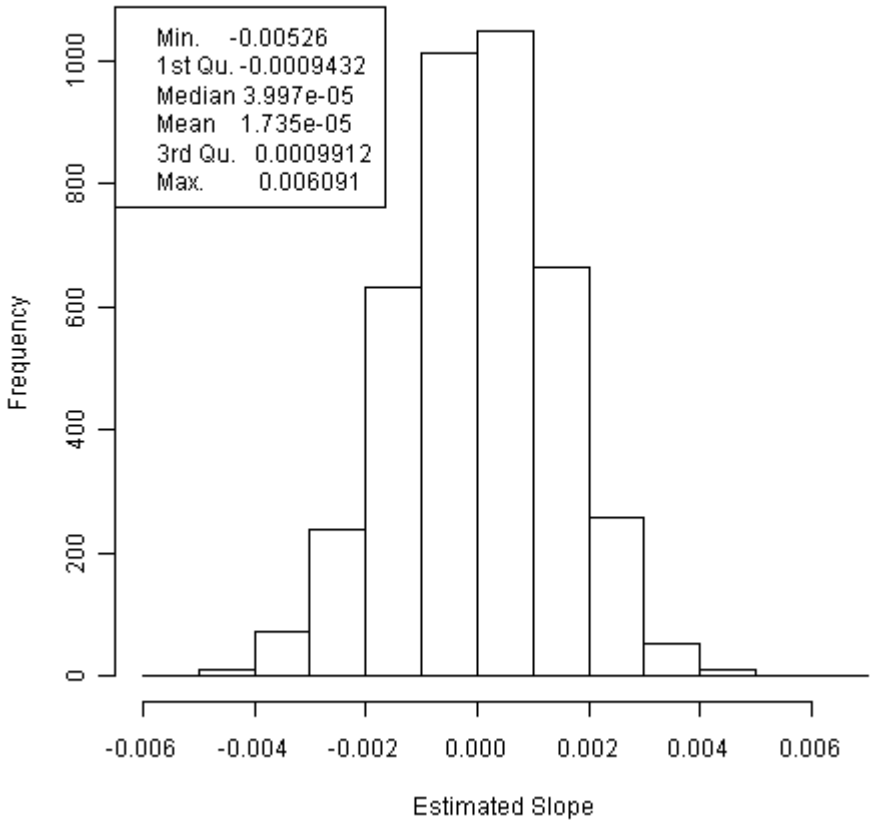


Figure A-5 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

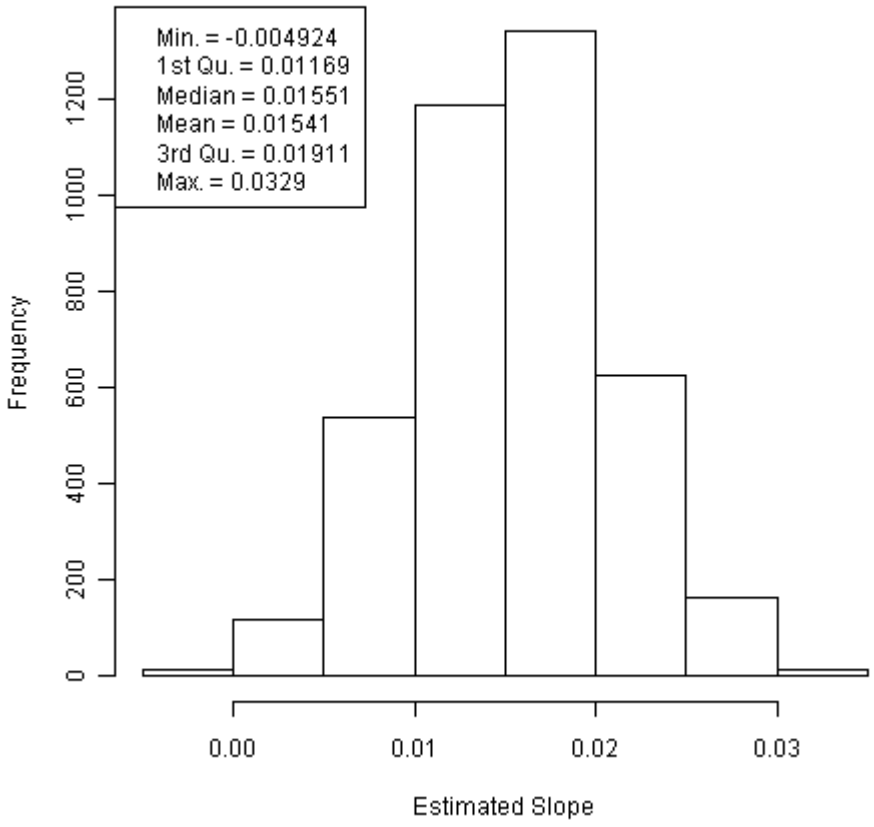


Figure A-6 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Bell-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

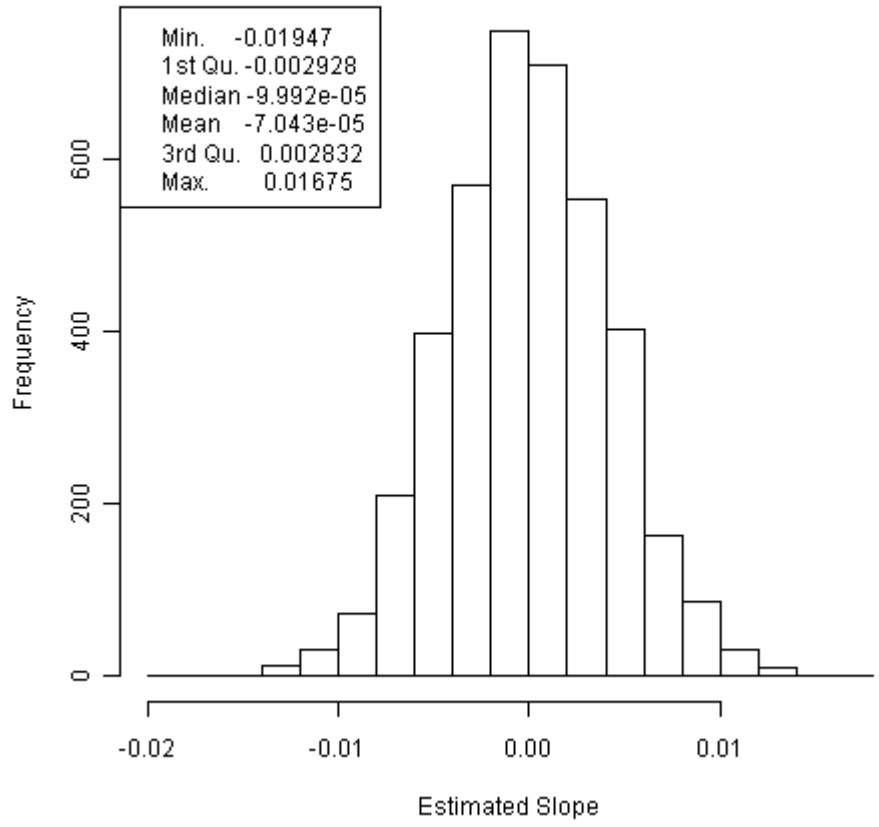


Figure A-7 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Triangularly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

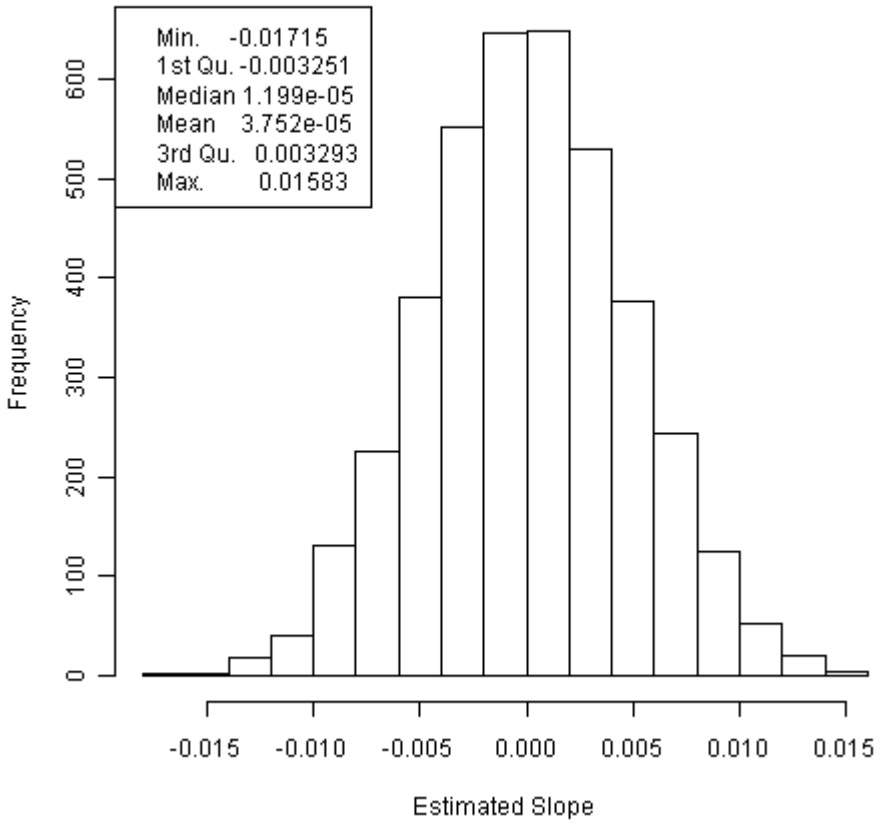


Figure A-8 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Exponentially-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 0$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

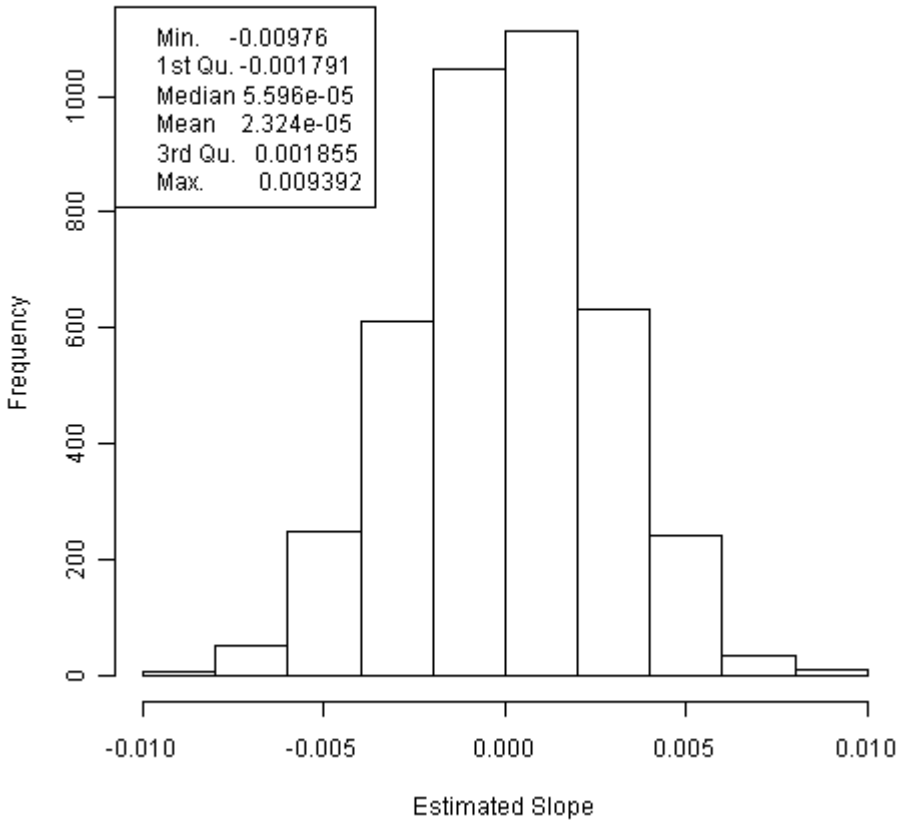


Figure A-9 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

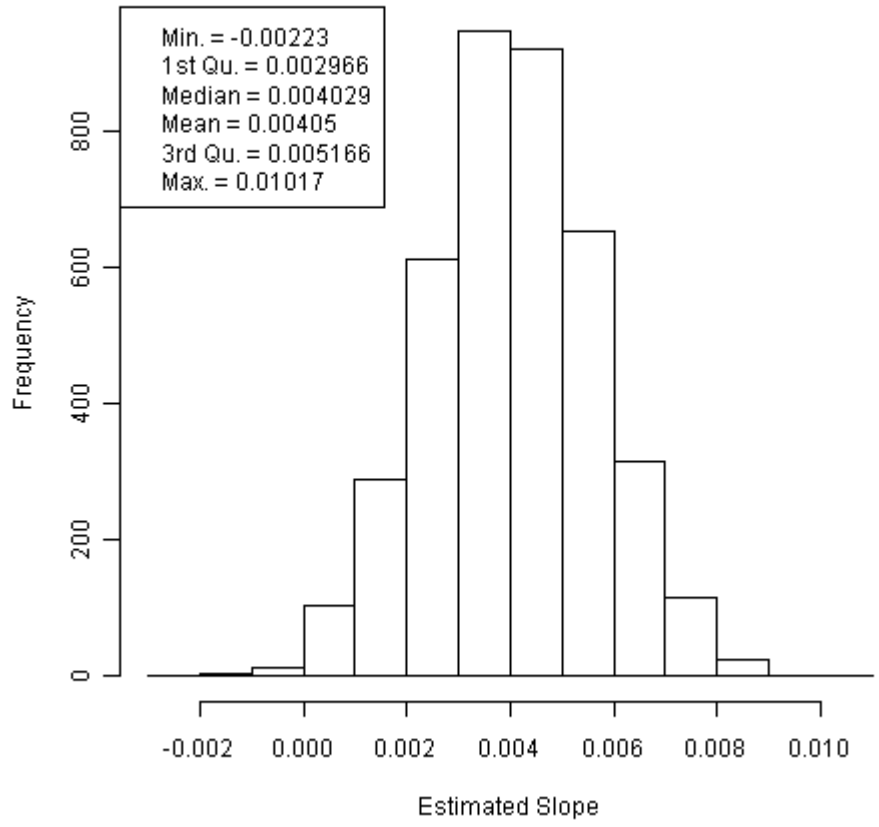


Figure A-10 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Bell-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

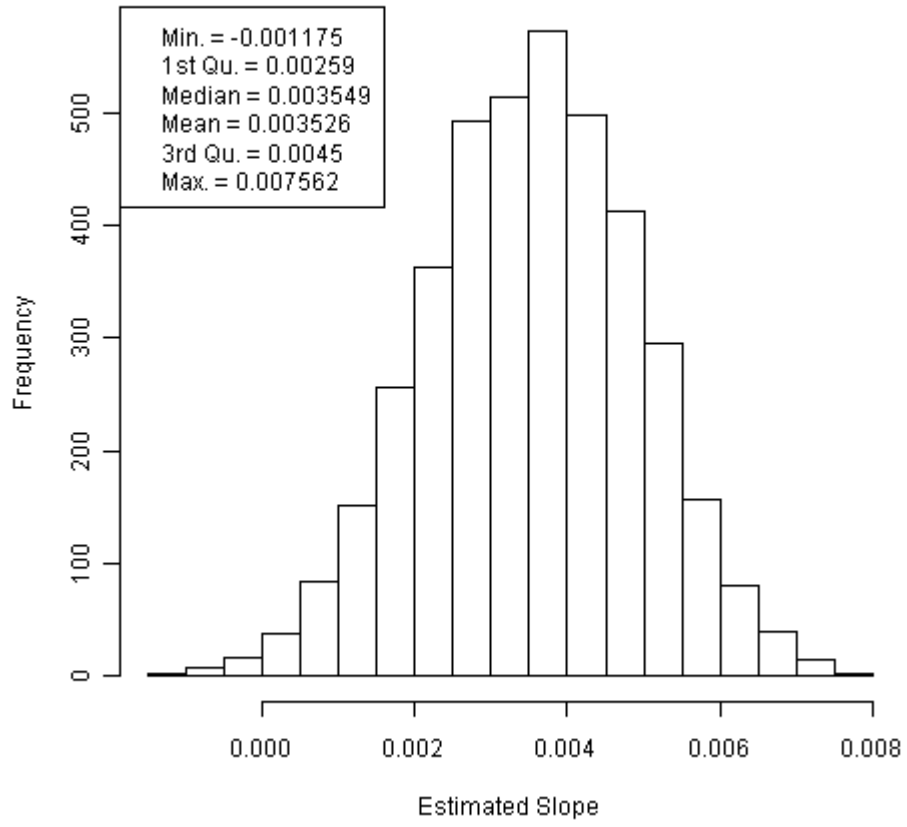


Figure A-11 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and a Triangularly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

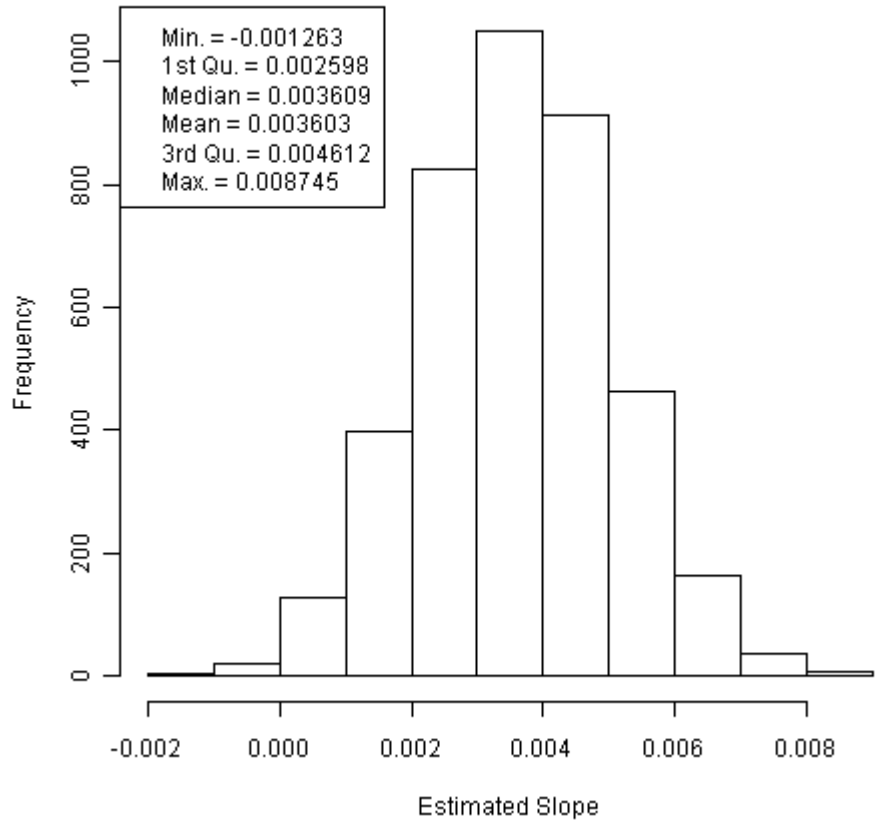


Figure A-12 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 3 Levels and an Exponentially-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

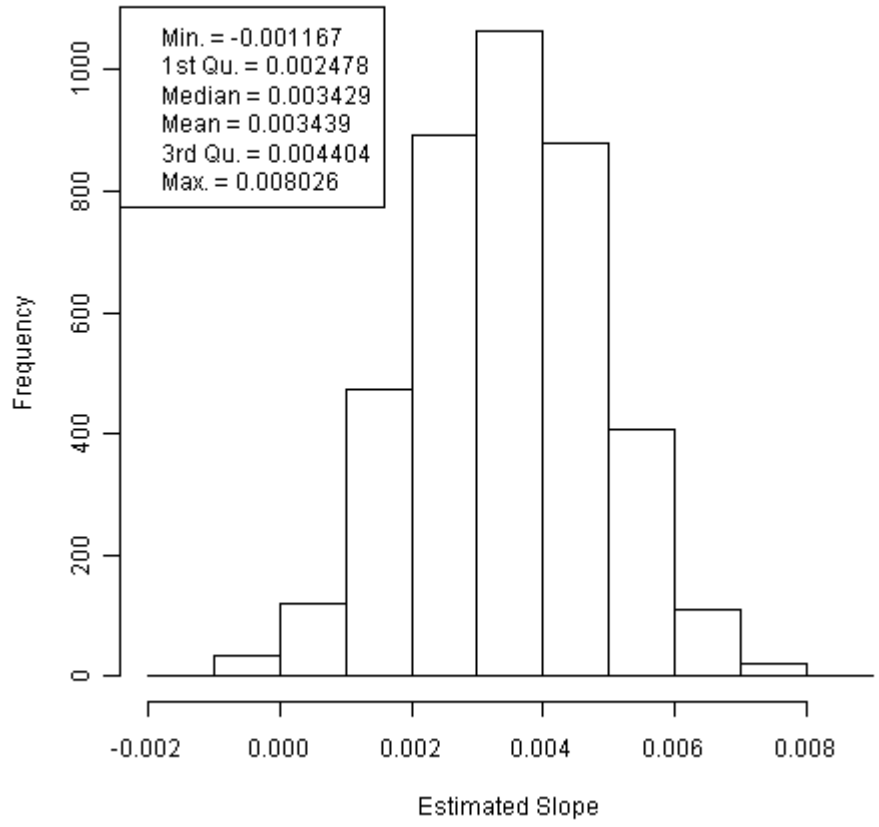


Figure A-13 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Uniformly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

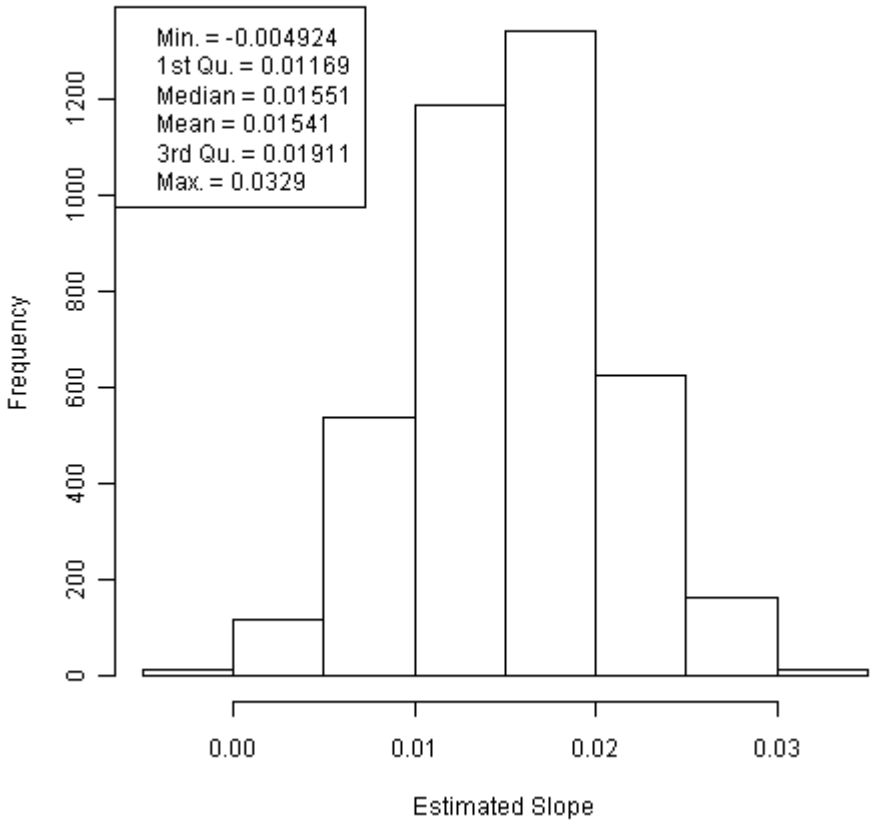


Figure A-14 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Bell-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

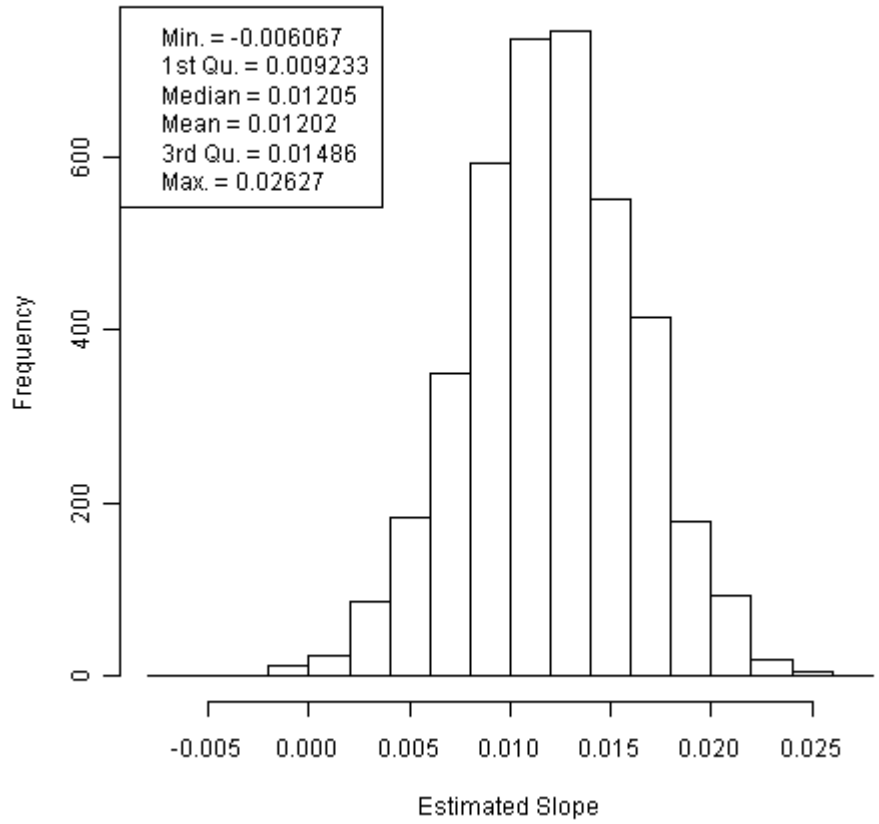


Figure A-15 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and a Triangularly-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.

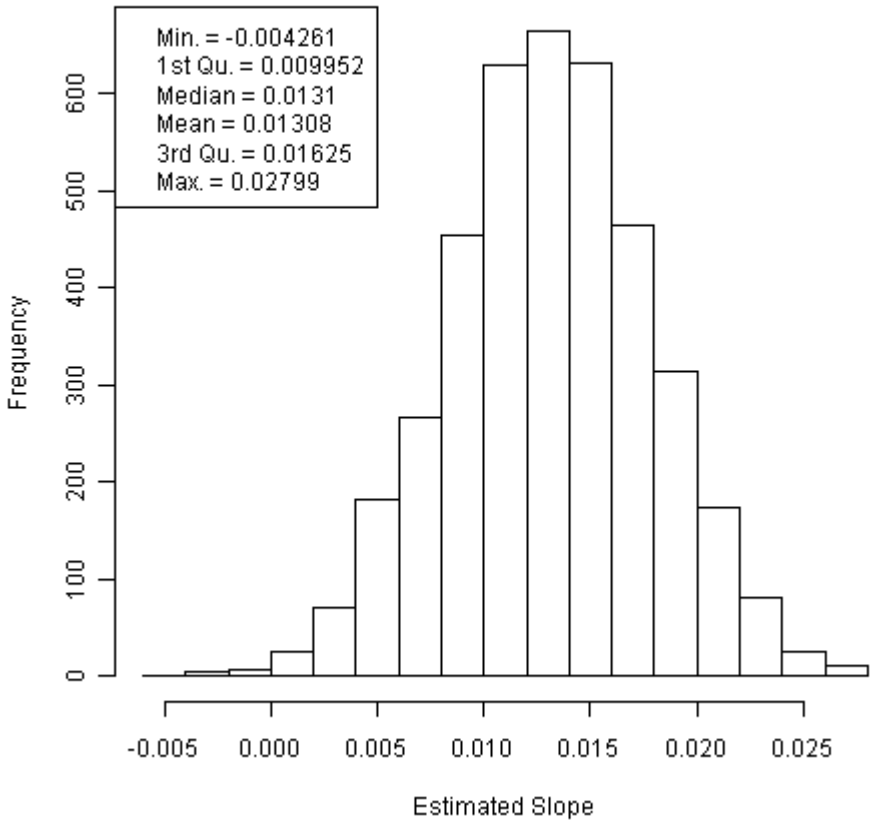
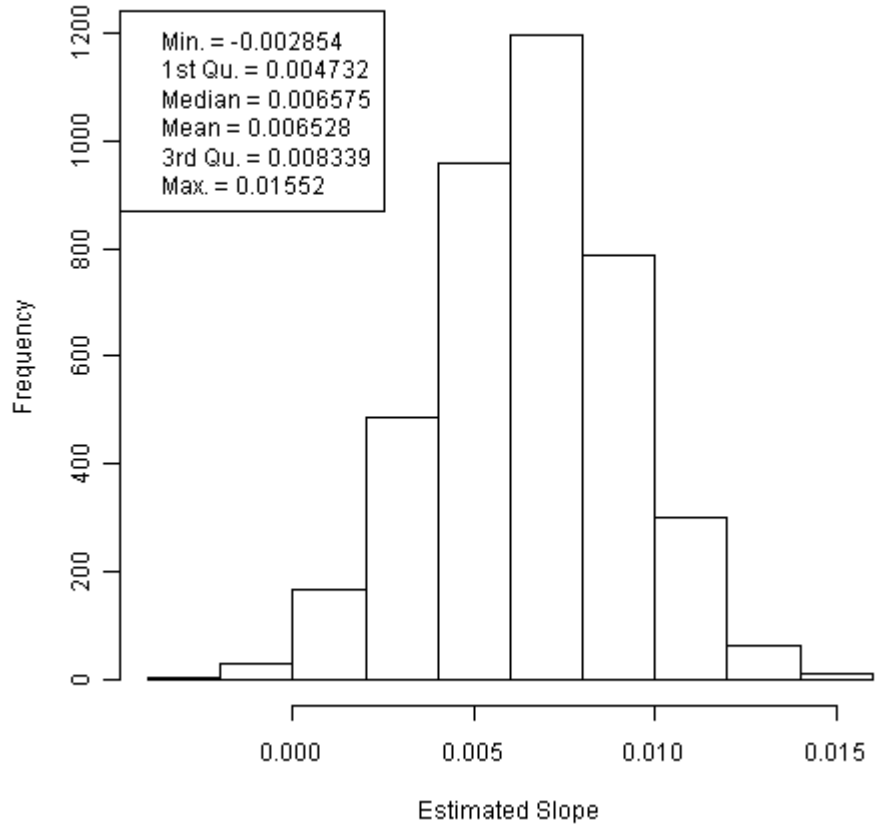


Figure A-16 Empirical sampling distribution for the slope parameter estimate obtained from fitting ordinary least squares linear regression to an ordered categorical response with 10 Levels and an Exponentially-Shaped frequency distribution. The histogram is based on 4000 Monte Carlo replications generated under the condition that $\beta^* = 1$. The figure represented the setting where the explanatory covariate consisted of 51 Levels ($x^{(51)}$) ranging from -50 to 50 in increments of 2.



Appendix B - Appendix B: Simulation Code for $\beta^* = 0$ Condition

```
##############################################################################
#Simulating the various cut and distribution scenarios
#I am going to create y variables based on x variables
#There will be no linear relationship between x and y (slope=0) plus some
#random noise.  This will allow me to assume an underlying dependent and
#independent variable which I may then cut anyway I please to make ordered categorical
#variables with any number of categories that maintain this underlying
#relationship
#
##############################################################################
#
#Part 1:  Create x variables
#Part 2:  Create error term
#Part 3:  Create underlying continuous latent variable y
#Part 4:  Run linear models and view R^2
#Part 5:  Calculated power and percent models significant
#Part 6:  Uniform
#Part 7:  Triangular
#Part 8:  Exponential
#Part 9:  Bell
#Part 10: Report power pictorally
#
#Parts 6-9 include the creation of the ordered categorical variable,
#ascertation of empirical power and production of various graphs.
#These sections are nearly identical, that is, looking at the section for
#Uniform will tell you exactly what happens in Triangular, Exponential and Bell,
#outside of the probabilities used to create the ordered categorical variables.
##############################################################################
##############################################################################

#Part 1:  Create x variables

#creating the predictor variable with 51 levels
xlatnames="xlat1111"   #making xlatnames a character variable
for (i in 1:4000) {xlatnames[i]=paste("xlat",i,sep="")}  #vector of names
x1=round(-49.5:49.5)  #making a vector of -50 to 50
xlat=rep(x1,12000)  #making a vector of -50 to 50 repeated 120000 times
dim(xlat)=c(300,4000)  #making a matrix of 4000 identical variables of length 300
colnames(xlat)=xlatnames  #assigning the x variables names

#creating the predictor matrix with 5 levels
x2=c(rep(-50,60),rep(-25,60),rep(0,60),rep(25,60),rep(50,60))
##could have used rep(seq(20,100,20),240000)###
xlat2=rep(x2,4000)
dim(xlat2)=c(300,4000)  #making a matrix of 4000 identical variables of length 300
colnames(xlat2)=xlatnames  #assigning the x variables names


##############################################################################
```

```
#Part 2:  Create the error terms (N(0,stdev51^2) and N(0,stdev5^2))

#51 Level Scenario
stdev51=177.9  #saving standard deviation for power calculation below
set.seed(54)  #set the randomization to a specific point

error51x1=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x1)=c(300,4000)

error51x2=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x2)=c(300,4000)

error51x3=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x3)=c(300,4000)

error51x4=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x4)=c(300,4000)

error51x5=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x5)=c(300,4000)

error51x6=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x6)=c(300,4000)

error51x7=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x7)=c(300,4000)

error51x8=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x8)=c(300,4000)

error51x9=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x9)=c(300,4000)

error51x10=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x10)=c(300,4000)

error51x11=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x11)=c(300,4000)

error51x12=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x12)=c(300,4000)

error51x13=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x13)=c(300,4000)

error51x14=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x14)=c(300,4000)

error51x15=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x15)=c(300,4000)

error51x16=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x16)=c(300,4000)
```

```
error51x17=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x17)=c(300,4000)

error51x18=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x18)=c(300,4000)

error51x19=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x19)=c(300,4000)

error51x20=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x20)=c(300,4000)

error51x21=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x21)=c(300,4000)

error51x22=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x22)=c(300,4000)

error51x23=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x23)=c(300,4000)

error51x24=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x24)=c(300,4000)



#5-Level Scenario
stdev5=217


error5x1=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x1)=c(300,4000)

error5x2=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x2)=c(300,4000)

error5x3=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x3)=c(300,4000)

error5x4=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x4)=c(300,4000)

error5x5=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x5)=c(300,4000)

error5x6=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x6)=c(300,4000)

error5x7=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x7)=c(300,4000)

error5x8=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x8)=c(300,4000)

error5x9=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
```

```
dim(error5x9)=c(300,4000)

error5x10=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x10)=c(300,4000)

error5x11=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x11)=c(300,4000)

error5x12=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x12)=c(300,4000)

error5x13=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x13)=c(300,4000)

error5x14=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x14)=c(300,4000)

error5x15=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x15)=c(300,4000)

error5x16=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x16)=c(300,4000)

error5x17=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x17)=c(300,4000)

error5x18=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x18)=c(300,4000)

error5x19=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x19)=c(300,4000)

error5x20=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x20)=c(300,4000)

error5x21=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x21)=c(300,4000)

error5x22=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x22)=c(300,4000)

error5x23=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x23)=c(300,4000)

error5x24=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x24)=c(300,4000)



##############################################################################

#Part 3:  Create underlying continuous latent variables based on x
```

```
slope=0
mytype="Type I Error"

#51-Level Scenario
ylatent51u2=slope*xlat+error51x1
ylatent51u3=slope*xlat+error51x2
ylatent51u4=slope*xlat+error51x3
ylatent51u5=slope*xlat+error51x4
ylatent51u7=slope*xlat+error51x5
ylatent51u10=slope*xlat+error51x6

ylatent51l2=slope*xlat+error51x7
ylatent51l3=slope*xlat+error51x8
ylatent51l4=slope*xlat+error51x9
ylatent51l5=slope*xlat+error51x10
ylatent51l7=slope*xlat+error51x11
ylatent51l10=slope*xlat+error51x12

ylatent51e2=slope*xlat+error51x13
ylatent51e3=slope*xlat+error51x14
ylatent51e4=slope*xlat+error51x15
ylatent51e5=slope*xlat+error51x16
ylatent51e7=slope*xlat+error51x17
ylatent51e10=slope*xlat+error51x18

ylatent51b2=slope*xlat+error51x19
ylatent51b3=slope*xlat+error51x20
ylatent51b4=slope*xlat+error51x21
ylatent51b5=slope*xlat+error51x22
ylatent51b7=slope*xlat+error51x23
ylatent51b10=slope*xlat+error51x24



#5-Level Scenario

ylatent5u22=slope*xlat2+error5x1
ylatent5u23=slope*xlat2+error5x2
ylatent5u24=slope*xlat2+error5x3
ylatent5u25=slope*xlat2+error5x4
ylatent5u27=slope*xlat2+error5x5
ylatent5u210=slope*xlat2+error5x6

ylatent5l22=slope*xlat2+error5x7
ylatent5l23=slope*xlat2+error5x8
ylatent5l24=slope*xlat2+error5x9
ylatent5l25=slope*xlat2+error5x10
ylatent5l27=slope*xlat2+error5x11
ylatent5l210=slope*xlat2+error5x12

ylatent5e22=slope*xlat2+error5x13
ylatent5e23=slope*xlat2+error5x14
ylatent5e24=slope*xlat2+error5x15
ylatent5e25=slope*xlat2+error5x16
ylatent5e27=slope*xlat2+error5x17
```

```
ylatent5e210=slope*xlat2+error5x18

ylatent5b22=slope*xlat2+error5x19
ylatent5b23=slope*xlat2+error5x20
ylatent5b24=slope*xlat2+error5x21
ylatent5b25=slope*xlat2+error5x22
ylatent5b27=slope*xlat2+error5x23
ylatent5b210=slope*xlat2+error5x24


################################################################################


#51-Level Scenario

rsq=vector("numeric",4000) #vector to hold r-squared values
fm1=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm1[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm2=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm2[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm3=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm3[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
```

```
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm4=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm4[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm5=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm6=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm6[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm7=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm7[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
```

```
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm8=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm8[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm9=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm9[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm10=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm10[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm11=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm11[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }
```

```
mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm12=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm12[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm13=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm13[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm14=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm14[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm15=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm15[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }
```

```
mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm16=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm16[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm17=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm17[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm18=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm18[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm19=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm19[i]=1}
rsq[i]=summary(m)$r.squared
```

```
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm20=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm20[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm21=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm21[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm22=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm22[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm23=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm23[i]=1}
```

```
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm24=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm24[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)



#5-Level Scenario

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x1=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x1[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x2=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x2[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x3=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
```

```
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x3[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x4=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x4[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x5=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x5[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x6=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x6[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
```

```
fm5x7=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x7[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x8=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x8[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x9=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x9[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x10=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x10[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)
```

```
rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x11=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x11[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x12=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x12[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x13=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x13[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x14=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x14[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)
```

```
rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x15=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x15[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x16=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x16[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x17=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x17[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x18=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x18[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
```

```
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x19=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x19[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x20=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x20[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x21=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x21[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x22=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x22[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
```

```
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x23=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x23[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x24=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x24[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)


##############################################################################

#Part 5:  Type I Error

#empirical type I error
per1=sum(fm1)/40  #divide by 40 b/c times 100 in numerator
per5x1=sum(fm5x1)/40

per2=sum(fm2)/40  #divide by 40 b/c times 100 in numerator
per5x2=sum(fm5x2)/40

per3=sum(fm3)/40  #divide by 40 b/c times 100 in numerator
per5x3=sum(fm5x3)/40

per4=sum(fm4)/40  #divide by 40 b/c times 100 in numerator
per5x4=sum(fm5x4)/40

per5=sum(fm5)/40  #divide by 40 b/c times 100 in numerator
per5x5=sum(fm5x5)/40

per6=sum(fm6)/40  #divide by 40 b/c times 100 in numerator
per5x6=sum(fm5x6)/40
```

```
per7=sum(fm7)/40  #divide by 40 b/c times 100 in numerator
per5x7=sum(fm5x7)/40

per8=sum(fm8)/40  #divide by 40 b/c times 100 in numerator
per5x8=sum(fm5x8)/40

per9=sum(fm9)/40  #divide by 40 b/c times 100 in numerator
per5x9=sum(fm5x9)/40

per10=sum(fm10)/40  #divide by 40 b/c times 100 in numerator
per5x10=sum(fm5x10)/40

per11=sum(fm11)/40  #divide by 40 b/c times 100 in numerator
per5x11=sum(fm5x11)/40

per12=sum(fm12)/40  #divide by 40 b/c times 100 in numerator
per5x12=sum(fm5x12)/40

per13=sum(fm13)/40  #divide by 40 b/c times 100 in numerator
per5x13=sum(fm5x13)/40

per14=sum(fm14)/40  #divide by 40 b/c times 100 in numerator
per5x14=sum(fm5x14)/40

per15=sum(fm15)/40  #divide by 40 b/c times 100 in numerator
per5x15=sum(fm5x15)/40

per16=sum(fm16)/40  #divide by 40 b/c times 100 in numerator
per5x16=sum(fm5x16)/40

per17=sum(fm17)/40  #divide by 40 b/c times 100 in numerator
per5x17=sum(fm5x17)/40

per18=sum(fm18)/40  #divide by 40 b/c times 100 in numerator
per5x18=sum(fm5x18)/40

per19=sum(fm19)/40  #divide by 40 b/c times 100 in numerator
per5x19=sum(fm5x19)/40

per20=sum(fm20)/40  #divide by 40 b/c times 100 in numerator
per5x20=sum(fm5x20)/40

per21=sum(fm21)/40  #divide by 40 b/c times 100 in numerator
per5x21=sum(fm5x21)/40

per22=sum(fm22)/40  #divide by 40 b/c times 100 in numerator
per5x22=sum(fm5x22)/40

per23=sum(fm23)/40  #divide by 40 b/c times 100 in numerator
per5x23=sum(fm5x23)/40

per24=sum(fm24)/40  #divide by 40 b/c times 100 in numerator
per5x24=sum(fm5x24)/40

per=c(per1,per2,per3,per4,per5,per6,per7,per8,per9,per10,per11,per12,per13,
```

```
per14,per15,per16,per17,per18,per19,per20,per21,per22,per23,per24)
hist(per)
mean(per)

per5x=c(per5x1,per5x2,per5x3,per5x4,per5x5,per5x6,per5x7,per5x8,per5x9,per5x10,
per5x11,per5x12,per5x13,per5x14,per5x15,per5x16,per5x17,per5x18,per5x19,per5x20,
per5x21,per5x22,per5x23,per5x24)
hist(per5x)
mean(per5x)



################################################################################
################################################################################




#Part 5:  Uniform

#set up probabilities for each section (matrix format with labels)
up=as.matrix(c(.5,0,0,0,0,0,0,0,0,
            (1/3),(1/3),0,0,0,0,0,0,0,
            .25,.25,.25,0,0,0,0,0,0,
           (1/5),(1/5),(1/5),(1/5),0,0,0,0,0,
           (1/7),(1/7),(1/7),(1/7),(1/7),(1/7),0,0,0,
           (1/10),(1/10),(1/10),(1/10),(1/10),(1/10),(1/10),(1/10),(1/10)))
dim(up)=c(9,6)
colnames(up)=c("Unif2", "Unif3", "Unif4", "Unif5", "Unif7", "Unif10")

#create empty matrices for ordered categorical uniform variables
utwo=matrix(0,300,4000)
uthree=matrix(0,300,4000)
ufour=matrix(0,300,4000)
ufive=matrix(0,300,4000)
useven=matrix(0,300,4000)
uten=matrix(0,300,4000)

utwo2=matrix(0,300,4000)
uthree2=matrix(0,300,4000)
ufour2=matrix(0,300,4000)
ufive2=matrix(0,300,4000)
useven2=matrix(0,300,4000)
uten2=matrix(0,300,4000)



#with the 51 level x (uniform)

for (k in 1:6){
        if (k==1){
                for (j in 1:4000){
                        for (i in 1:300){
```

```
                                        if(ylatent51u2[i,j]<=qnorm(up[1,k],0,sd(ylatent51u2[,j])+1)){utwo[i,j]=1}
else{utwo[i,j]=2}}}}


        if (k==2){
                for (j in 1:4000){
                        for (i in 1:300){
                                        if(ylatent51u3[i,j]<=qnorm(up[1,k],0,sd(ylatent51u3[,j])+1)){uthree[i,j]=1}
else{

        if(ylatent51u3[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u3[,j])+1)){uthree[i,j]=2}
else{uthree[i,j]=3}}}}}


        if (k==3){
                for (j in 1:4000){
                        for (i in 1:300){
                                        if(ylatent51u4[i,j]<=qnorm(up[1,k],0,sd(ylatent51u4[,j])+1)){ufour[i,j]=1} else{

        if(ylatent51u4[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u4[,j])+1)){ufour[i,j]=2} else{

        if(ylatent51u4[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent51u4[,j])+1)){ufour[i,j]=3}
else{ufour[i,j]=4}}}}}}


        if (k==4){
                for (j in 1:4000){
                        for (i in 1:300){
                                        if(ylatent51u5[i,j]<=qnorm(up[1,k],0,sd(ylatent51u5[,j])+1)){ufive[i,j]=1} else{

        if(ylatent51u5[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u5[,j])+1)){ufive[i,j]=2} else{

        if(ylatent51u5[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent51u5[,j])+1)){ufive[i,j]=3} else{

        if(ylatent51u5[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent51u5[,j])+1)){ufive[i,j]=4}
else{ufive[i,j]=5}}}}}}}



        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                        if(ylatent51u7[i,j]<=qnorm(up[1,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]=1}
else{

        if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]=2} else{

        if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]=3} else{

        if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]=4}
else{

        if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]
=5}else{
```

```
	if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k],0,sd(ylatent51u7[,j])+1)){us
even[i,j]=6}else{useven[i,j]=7}}}}}}}}}

	if (k==6){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51u10[i,j]<=qnorm(up[1,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]=1}
else{

	if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]=2} else{

	if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]=3} else{

	if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]=4}
else{

	if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]
=5}else{

	if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k],0,sd(ylatent51u10[,j])+1))
{uten[i,j]=6}else{

	if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k],0,sd(ylatent51u10
[,j])+1)){uten[i,j]=7}else{

	if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k]+up[8,k],0,sd(ylate
nt51u10[,j])+1)){uten[i,j]=8}else{

	if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k]+up[8,k]+up[9,k],
0,sd(ylatent51u10[,j])+1)){uten[i,j]=9}else{uten[i,j]=10}}}}}}}}}}}}}

}

write.csv(utwo, file="utwo0.csv")
write.csv(uthree, file="uthree0.csv")
write.csv(ufour, file="ufour0.csv")
write.csv(ufive, file="ufive0.csv")
write.csv(useven, file="useven0.csv")
write.csv(uten, file="uten0.csv")



#with the 5 level x (uniform)

for (k in 1:6){
	if (k==1){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5u22[i,j]<=qnorm(up[1,k],0,sd(ylatent5u22[,j])+1)){utwo2[i,j]=1}
else{utwo2[i,j]=2}}}}
```

```
	if (k==2){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5u23[i,j]<=qnorm(up[1,k],0,sd(ylatent5u23[,j])+1)){uthree2[i,j]=1}
else{

	if(ylatent5u23[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u23[,j])+1)){uthree2[i,j]=2}
else{uthree2[i,j]=3}}}}}



	if (k==3){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5u24[i,j]<=qnorm(up[1,k],0,sd(ylatent5u24[,j])+1)){ufour2[i,j]=1}
else{

	if(ylatent5u24[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u24[,j])+1)){ufour2[i,j]=2} else{

	if(ylatent5u24[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent5u24[,j])+1)){ufour2[i,j]=3}
else{ufour2[i,j]=4}}}}}}



	if (k==4){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5u25[i,j]<=qnorm(up[1,k],0,sd(ylatent5u25[,j])+1)){ufive2[i,j]=1}
else{

	if(ylatent5u25[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u25[,j])+1)){ufive2[i,j]=2} else{

	if(ylatent5u25[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent5u25[,j])+1)){ufive2[i,j]=3} else{

	if(ylatent5u25[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent5u25[,j])+1)){ufive2[i,j]=4}
else{ufive2[i,j]=5}}}}}}}



	if (k==5){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5u27[i,j]<=qnorm(up[1,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j]=1}
else{

	if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j]=2} else{

	if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j]=3} else{

	if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j]=4}
else{
```

```
	if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j
]=5}else{

	if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k],0,sd(ylatent5u27[,j])+1)){us
even2[i,j]=6}else{useven2[i,j]=7}}}}}}}}}


	if (k==6){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5u210[i,j]<=qnorm(up[1,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j]=1}
else{

	if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j]=2} else{

	if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j]=3} else{

	if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j]=4}
else{

	if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j
]=5}else{

	if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k],0,sd(ylatent5u210[,j])+1))
{uten2[i,j]=6}else{

	if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k],0,sd(ylatent5u210
[,j])+1)){uten2[i,j]=7}else{

	if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k]+up[8,k],0,sd(ylate
nt5u210[,j])+1)){uten2[i,j]=8}else{

	if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k]+up[8,k]+up[9,k],
0,sd(ylatent5u210[,j])+1)){uten2[i,j]=9}else{uten2[i,j]=10}}}}}}}}}}


}

write.csv(utwo2, file="utwo02.csv")
write.csv(uthree2, file="uthree02.csv")
write.csv(ufour2, file="ufour02.csv")
write.csv(ufive2, file="ufive02.csv")
write.csv(useven2, file="useven02.csv")
write.csv(uten2, file="uten02.csv")



#Get percent significant for each level
#vectors of zeros to hold 0,1 values, 1 if model had significant slope
#um2 -- uniform model 2 level dependent variable cont is the 100 level x, cat is the 5 level
um2cont=vector("numeric",4000)
um3cont=vector("numeric",4000)
um4cont=vector("numeric",4000)
um5cont=vector("numeric",4000)
```

```
um7cont=vector("numeric",4000)
um10cont=vector("numeric",4000)

um2cat=um2cont
um3cat=um2cont
um4cat=um2cont
um5cat=um2cont
um7cat=um2cont
um10cat=um2cont


#create empty vectors to store estimates of slope
uslopeone=vector("numeric",4000)
uslopeone2=vector("numeric",4000)
uslopetwo=vector("numeric",4000)
uslopetwo2=vector("numeric",4000)
uslopethree=vector("numeric",4000)
uslopethree2=vector("numeric",4000)
uslopefour=vector("numeric",4000)
uslopefour2=vector("numeric",4000)
uslopefive=vector("numeric",4000)
uslopefive2=vector("numeric",4000)
uslopeseven=vector("numeric",4000)
uslopeseven2=vector("numeric",4000)
uslopeten=vector("numeric",4000)
uslopeten2=vector("numeric",4000)


for (i in 1:4000) {

#running the linear models
#none of the cat ones are relevant yet and will require a variable change to be relevant
m2cont=lm(utwo[,i]~xlat[,1])
m2cat=lm(utwo2[,i]~xlat2[,1])
m3cont=lm(uthree[,i]~xlat[,1])
m3cat=lm(uthree2[,i]~xlat2[,1])
m4cont=lm(ufour[,i]~xlat[,1])
m4cat=lm(ufour2[,i]~xlat2[,1])
m5cont=lm(ufive[,i]~xlat[,1])
m5cat=lm(ufive2[,i]~xlat2[,1])
m7cont=lm(useven[,i]~xlat[,1])
m7cat=lm(useven2[,i]~xlat2[,1])
m10cont=lm(uten[,i]~xlat[,1])
m10cat=lm(uten2[,i]~xlat2[,1])

#grabbing the slopes
uslopetwo[i]=m2cont$coefficients[2]
uslopetwo2[i]=m2cat$coefficients[2]
uslopethree[i]=m3cont$coefficients[2]
uslopethree2[i]=m3cat$coefficients[2]
uslopefour[i]=m4cont$coefficients[2]
uslopefour2[i]=m4cat$coefficients[2]
uslopefive[i]=m5cont$coefficients[2]
uslopefive2[i]=m5cat$coefficients[2]
uslopeseven[i]=m7cont$coefficients[2]
```

```
uslopeseven2[i]=m7cat$coefficients[2]
uslopeten[i]=m10cont$coefficients[2]
uslopeten2[i]=m10cat$coefficients[2]



#counting how many of them are significant
if (summary(m2cont)$fstatistic[1] > qf(.95,summary(m2cont)$fstatistic[2],summary(m2cont)$fstatistic[3]))
{um2cont[i]=1}
if (summary(m2cat)$fstatistic[1] > qf(.95,summary(m2cat)$fstatistic[2],summary(m2cat)$fstatistic[3]))
{um2cat[i]=1}

if (summary(m3cont)$fstatistic[1] > qf(.95,summary(m3cont)$fstatistic[2],summary(m3cont)$fstatistic[3]))
{um3cont[i]=1}
if (summary(m3cat)$fstatistic[1] > qf(.95,summary(m3cat)$fstatistic[2],summary(m3cat)$fstatistic[3]))
{um3cat[i]=1}

if (summary(m4cont)$fstatistic[1] > qf(.95,summary(m4cont)$fstatistic[2],summary(m4cont)$fstatistic[3]))
{um4cont[i]=1}
if (summary(m4cat)$fstatistic[1] > qf(.95,summary(m4cat)$fstatistic[2],summary(m4cat)$fstatistic[3]))
{um4cat[i]=1}

if (summary(m5cont)$fstatistic[1] > qf(.95,summary(m5cont)$fstatistic[2],summary(m5cont)$fstatistic[3]))
{um5cont[i]=1}
if (summary(m5cat)$fstatistic[1] > qf(.95,summary(m5cat)$fstatistic[2],summary(m5cat)$fstatistic[3]))
{um5cat[i]=1}

if (summary(m7cont)$fstatistic[1] > qf(.95,summary(m7cont)$fstatistic[2],summary(m7cont)$fstatistic[3]))
{um7cont[i]=1}
if (summary(m7cat)$fstatistic[1] > qf(.95,summary(m7cat)$fstatistic[2],summary(m7cat)$fstatistic[3]))
{um7cat[i]=1}

if (summary(m10cont)$fstatistic[1] > qf(.95,summary(m10cont)$fstatistic[2],summary(m10cont)$fstatistic[3]))
{um10cont[i]=1}
if (summary(m10cat)$fstatistic[1] > qf(.95,summary(m10cat)$fstatistic[2],summary(m10cat)$fstatistic[3]))
{um10cat[i]=1}
}

#getting the percentages
uper2cont=sum(um2cont)/40
uper2cat=sum(um2cat)/40

uper3cont=sum(um3cont)/40
uper3cat=sum(um3cat)/40

uper4cont=sum(um4cont)/40
uper4cat=sum(um4cat)/40

uper5cont=sum(um5cont)/40
uper5cat=sum(um5cat)/40

uper7cont=sum(um7cont)/40
uper7cat=sum(um7cat)/40
```

```
uper10cont=sum(um10cont)/40
uper10cat=sum(um10cat)/40

write.csv(c(uper2cont,uper3cont,uper4cont,uper5cont,uper7cont,uper10cont,uper2cat,uper3cat,
uper4cat,uper5cat,uper7cat,uper10cat),file="uper0.csv")



#getting histograms of the estimated standard errors

#define my basic histogram function
slopehist=function(shapelevel,title){
hist(shapelevel,main="",xlab="Estimated Slope")
legend("topleft",legend=c(paste(attributes(summary(shapelevel))$names[1],summary(shapelevel)[1],sep="      "),
paste(attributes(summary(shapelevel))$names[2],summary(shapelevel)[2]),
paste(attributes(summary(shapelevel))$names[3],summary(shapelevel)[3]),
paste(attributes(summary(shapelevel))$names[4],summary(shapelevel)[4],sep="     "),
paste(attributes(summary(shapelevel))$names[5],summary(shapelevel)[5],sep="    "),
paste(attributes(summary(shapelevel))$names[6],summary(shapelevel)[6],sep="          "))))}

slopeuboth=cbind(uslopetwo,uslopethree,uslopefour,uslopefive,uslopeseven,uslopeten,uslopetwo2,uslopethree2,uslo
pefour2,uslopefive2,uslopeseven2,uslopeten2)
slpnames=c("Uniform 50, 2 category","Uniform 50, 3 category","Uniform 50, 4 category","Uniform 50, 5
category","Uniform 50, 7 category","Uniform 50, 10 category",
"Uniform 5, 2 category","Uniform 5, 3 category","Uniform 5, 4 category","Uniform 5, 5 category","Uniform 5, 7
category","Uniform 5, 10 category")

png("uhistslopeszero%02d.png")  #saving sampling distributions of slopes to an external file
for (i in 1:12){slopehist(shapelevel=slopeuboth[,i],title=slpnames[i])}
dev.off()

quantile(uslopetwo,c(.025,.975))


################################################################################
################################################################################




#Part 6:  Triangular

#set triangular probabilities for each section (matrix format with labels)
triangular=as.matrix(c((1/3),0,0,0,0,0,0,0,0,(1/6),(2/6),0,0,0,0,0,0,0,(1/10),(2/10),(3/10),0,0,0,0,0,0,(1/15),(2/15),(3/1
5),(4/15),0,0,0,0,0,(1/28),(2/28),(3/28),(4/28),(5/28),
(6/28),0,0,0,(1/55),(2/55),(3/55),(4/55),(5/55),(6/55),(7/55),(8/55),(9/55)))
dim(triangular)=c(9,6)
colnames(triangular)=c("Triangular2", "Triangular3", "Triangular4", "Triangular5", "Triangular7", "Triangular10")

#create empty matrices for ordered categorical “triangular” variables
ltwo=matrix(0,300,4000)
lthree=matrix(0,300,4000)
lfour=matrix(0,300,4000)
lfive=matrix(0,300,4000)
```

```
lseven=matrix(0,300,4000)
lten=matrix(0,300,4000)

ltwo2=matrix(0,300,4000)
lthree2=matrix(0,300,4000)
lfour2=matrix(0,300,4000)
lfive2=matrix(0,300,4000)
lseven2=matrix(0,300,4000)
lten2=matrix(0,300,4000)


#with the 51 level x (triangular)

for (k in 1:6){
	if (k==1){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5112[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5112[,j])+1)){ltwo[i,j]=1}
else{ltwo[i,j]=2}}}}



	if (k==2){
		for (j in 1:4000){
			for (i in 1:300){

	if(ylatent5113[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5113[,j])+1)){lthree[i,j]=1} else{

	if(ylatent5113[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent5113[,j])+1)){lthree[i,j]=2}
else{lthree[i,j]=3}}}}}


	if (k==3){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5114[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5114[,j])+1)){lfour[i,j]=1}
else{

	if(ylatent5114[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent5114[,j])+1)){lfour[i,j]=2} else{

	if(ylatent5114[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent5114[,j])+1)){lfour[i,j
]=3} else{lfour[i,j]=4}}}}}}


	if (k==4){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5115[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5115[,j])+1)){lfive[i,j]=1}
else{

	if(ylatent5115[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent5115[,j])+1)){lfive[i,j]=2} else{

	if(ylatent5115[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent5115[,j])+1)){lfive[i,j
]=3} else{
```

```
if(ylatent51l5[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k],0,sd(ylatent51l5[,
j])+1)){lfive[i,j]=4} else{lfive[i,j]=5}}}}}}}


if (k==5){
        for (j in 1:4000){
                for (i in 1:300){

if(ylatent51l7[i,j]<=qnorm(triangular[1,k],0,sd(ylatent51l7[,j])+1)){lseven[i,j]=1} else{

if(ylatent51l7[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent51l7[,j])+1)){lseven[i,j]=2} else{

if(ylatent51l7[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent51l7[,j])+1)){lseven[i
,j]=3} else{

if(ylatent51l7[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k],0,sd(ylatent51l7[,
j])+1)){lseven[i,j]=4} else{

if(ylatent51l7[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k],0
,sd(ylatent51l7[,j])+1)){lseven[i,j]=5}else{

if(ylatent51l7[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]+t
riangular[6,k],0,sd(ylatent51l7[,j])+1)){lseven[i,j]=6}else{lseven[i,j]=7}}}}}}}}}


if (k==6){
        for (j in 1:4000){
                for (i in 1:300){

if(ylatent51l10[i,j]<=qnorm(triangular[1,k],0,sd(ylatent51l10[,j])+1)){lten[i,j]=1} else{

if(ylatent51l10[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent51l10[,j])+1)){lten[i,j]=2} else{

if(ylatent51l10[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent51l10[,j])+1)){lten[i
,j]=3} else{

if(ylatent51l10[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k],0,sd(ylatent51l1
0[,j])+1)){lten[i,j]=4} else{

if(ylatent51l10[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k],
0,sd(ylatent51l10[,j])+1)){lten[i,j]=5}else{

if(ylatent51l10[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k],0,sd(ylatent51l10[,j])+1)){lten[i,j]=6}else{

if(ylatent51l10[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k]+triangular[7,k],0,sd(ylatent51l10[,j])+1)){lten[i,j]=7}else{

if(ylatent51l10[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k]+triangular[7,k]+triangular[8,k],0,sd(ylatent51l10[,j])+1)){lten[i,j]=8}else{

if(ylatent51l10[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
```

```
+triangular[6,k]+triangular[7,k]+triangular[8,k]+triangular[9,k],0,sd(ylatent51110[,j])+1)){lten[i,j]=9}else{lten[i,j]=
10}}}}}}}}}}}}

}

write.csv(ltwo, file="ltwo0.csv")
write.csv(lthree, file="lthree0.csv")
write.csv(lfour, file="lfour0.csv")
write.csv(lfive, file="lfive0.csv")
write.csv(lseven, file="lseven0.csv")
write.csv(lten, file="lten0.csv")



#with the 5 level x (triangular)

for (k in 1:6){
        if (k==1){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5l22[i,j]<=qnorm(lin[1,k],0,sd(ylatent5l22[,j])+1)){ltwo2[i,j]=1}
else{ltwo2[i,j]=2}}}}



        if (k==2){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5l23[i,j]<=qnorm(lin[1,k],0,sd(ylatent5l23[,j])+1)){lthree2[i,j]=1}
else{

        if(ylatent5l23[i,j]<=qnorm(lin[1,k]+lin[2,k],0,sd(ylatent5l23[,j])+1)){lthree2[i,j]=2}
else{lthree2[i,j]=3}}}}}



        if (k==3){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5l24[i,j]<=qnorm(lin[1,k],0,sd(ylatent5l24[,j])+1)){lfour2[i,j]=1} else{

        if(ylatent5l24[i,j]<=qnorm(lin[1,k]+lin[2,k],0,sd(ylatent5l24[,j])+1)){lfour2[i,j]=2} else{

        if(ylatent5l24[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k],0,sd(ylatent5l24[,j])+1)){lfour2[i,j]=3}
else{lfour2[i,j]=4}}}}}}



        if (k==4){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5l25[i,j]<=qnorm(lin[1,k],0,sd(ylatent5l25[,j])+1)){lfive2[i,j]=1} else{

        if(ylatent5l25[i,j]<=qnorm(lin[1,k]+lin[2,k],0,sd(ylatent5l25[,j])+1)){lfive2[i,j]=2} else{
```

```
        if(ylatent5l25[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k],0,sd(ylatent5l25[,j])+1)){lfive2[i,j]=3} else{

        if(ylatent5l25[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k],0,sd(ylatent5l25[,j])+1)){lfive2[i,j]=4}
else{lfive2[i,j]=5}}}}}}}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5l27[i,j]<=qnorm(lin[1,k],0,sd(ylatent5l27[,j])+1)){lseven2[i,j]=1}
else{

        if(ylatent5l27[i,j]<=qnorm(lin[1,k]+lin[2,k],0,sd(ylatent5l27[,j])+1)){lseven2[i,j]=2} else{

        if(ylatent5l27[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k],0,sd(ylatent5l27[,j])+1)){lseven2[i,j]=3} else{

        if(ylatent5l27[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k],0,sd(ylatent5l27[,j])+1)){lseven2[i,j]=4}
else{

        if(ylatent5l27[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k]+lin[5,k],0,sd(ylatent5l27[,j])+1)){lseven2[i,j]
=5}else{

        if(ylatent5l27[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k]+lin[5,k]+lin[6,k],0,sd(ylatent5l27[,j])+1)){ls
even2[i,j]=6}else{lseven2[i,j]=7}}}}}}}}}


        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5l210[i,j]<=qnorm(lin[1,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=1}
else{

        if(ylatent5l210[i,j]<=qnorm(lin[1,k]+lin[2,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=2} else{

        if(ylatent5l210[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=3} else{

        if(ylatent5l210[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=4}
else{

        if(ylatent5l210[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k]+lin[5,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]
=5}else{

        if(ylatent5l210[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k]+lin[5,k]+lin[6,k],0,sd(ylatent5l210[,j])+1)){
lten2[i,j]=6}else{

        if(ylatent5l210[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k]+lin[5,k]+lin[6,k]+lin[7,k],0,sd(ylatent5l210
[,j])+1)){lten2[i,j]=7}else{

        if(ylatent5l210[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k]+lin[5,k]+lin[6,k]+lin[7,k]+lin[8,k],0,sd(ylat
ent5l210[,j])+1)){lten2[i,j]=8}else{

        if(ylatent5l210[i,j]<=qnorm(lin[1,k]+lin[2,k]+lin[3,k]+lin[4,k]+lin[5,k]+lin[6,k]+lin[7,k]+lin[8,k]+lin[9,k],
0,sd(ylatent5l210[,j])+1)){lten2[i,j]=9}else{lten2[i,j]=10}}}}}}}}}}}
```

```
}

write.csv(ltwo2, file="ltwo02.csv")
write.csv(lthree2, file="lthree02.csv")
write.csv(lfour2, file="lfour02.csv")
write.csv(lfive2, file="lfive02.csv")
write.csv(lseven2, file="lseven02.csv")
write.csv(lten2, file="lten02.csv")


#Get percent significant for each level


#vectors of zeros to hold 0,1 valles, 1 if model had significant slope
#lm2 -- linear model 2 level dependent variable cont is the 100 level x, cat is the 5 level
lm2cont=vector("numeric",4000)
lm3cont=vector("numeric",4000)
lm4cont=vector("numeric",4000)
lm5cont=vector("numeric",4000)
lm7cont=vector("numeric",4000)
lm10cont=vector("numeric",4000)

lm2cat=lm2cont
lm3cat=lm2cont
lm4cat=lm2cont
lm5cat=lm2cont
lm7cat=lm2cont
lm10cat=lm2cont


#create empty vectors to store estimates of slope
lslopeone=vector("numeric",4000)
lslopeone2=vector("numeric",4000)
lslopetwo=vector("numeric",4000)
lslopetwo2=vector("numeric",4000)
lslopethree=vector("numeric",4000)
lslopethree2=vector("numeric",4000)
lslopefour=vector("numeric",4000)
lslopefour2=vector("numeric",4000)
lslopefive=vector("numeric",4000)
lslopefive2=vector("numeric",4000)
lslopeseven=vector("numeric",4000)
lslopeseven2=vector("numeric",4000)
lslopeten=vector("numeric",4000)
lslopeten2=vector("numeric",4000)


for (i in 1:4000) {

#running the linear models
#none of the cat ones are relevant yet and will require a variable change to be relevant
m2cont=lm(ltwo[,i]~xlat[,1])
m2cat=lm(ltwo2[,i]~xlat2[,1])
m3cont=lm(lthree[,i]~xlat[,1])
m3cat=lm(lthree2[,i]~xlat2[,1])
```

```
m4cont=lm(lfour[,i]~xlat[,1])
m4cat=lm(lfour2[,i]~xlat2[,1])
m5cont=lm(lfive[,i]~xlat[,1])
m5cat=lm(lfive2[,i]~xlat2[,1])
m7cont=lm(lseven[,i]~xlat[,1])
m7cat=lm(lseven2[,i]~xlat2[,1])
m10cont=lm(lten[,i]~xlat[,1])
m10cat=lm(lten2[,i]~xlat2[,1])


#grabbing the slopes
lslopetwo[i]=m2cont$coefficients[2]
lslopetwo2[i]=m2cat$coefficients[2]
lslopethree[i]=m3cont$coefficients[2]
lslopethree2[i]=m3cat$coefficients[2]
lslopefour[i]=m4cont$coefficients[2]
lslopefour2[i]=m4cat$coefficients[2]
lslopefive[i]=m5cont$coefficients[2]
lslopefive2[i]=m5cat$coefficients[2]
lslopeseven[i]=m7cont$coefficients[2]
lslopeseven2[i]=m7cat$coefficients[2]
lslopeten[i]=m10cont$coefficients[2]
lslopeten2[i]=m10cat$coefficients[2]


#counting how many of them are significant
if (summary(m2cont)$fstatistic[1] > qf(.95,summary(m2cont)$fstatistic[2],summary(m2cont)$fstatistic[3]))
{lm2cont[i]=1}
if (summary(m2cat)$fstatistic[1] > qf(.95,summary(m2cat)$fstatistic[2],summary(m2cat)$fstatistic[3]))
{lm2cat[i]=1}

if (summary(m3cont)$fstatistic[1] > qf(.95,summary(m3cont)$fstatistic[2],summary(m3cont)$fstatistic[3]))
{lm3cont[i]=1}
if (summary(m3cat)$fstatistic[1] > qf(.95,summary(m3cat)$fstatistic[2],summary(m3cat)$fstatistic[3]))
{lm3cat[i]=1}

if (summary(m4cont)$fstatistic[1] > qf(.95,summary(m4cont)$fstatistic[2],summary(m4cont)$fstatistic[3]))
{lm4cont[i]=1}
if (summary(m4cat)$fstatistic[1] > qf(.95,summary(m4cat)$fstatistic[2],summary(m4cat)$fstatistic[3]))
{lm4cat[i]=1}

if (summary(m5cont)$fstatistic[1] > qf(.95,summary(m5cont)$fstatistic[2],summary(m5cont)$fstatistic[3]))
{lm5cont[i]=1}
if (summary(m5cat)$fstatistic[1] > qf(.95,summary(m5cat)$fstatistic[2],summary(m5cat)$fstatistic[3]))
{lm5cat[i]=1}

if (summary(m7cont)$fstatistic[1] > qf(.95,summary(m7cont)$fstatistic[2],summary(m7cont)$fstatistic[3]))
{lm7cont[i]=1}
if (summary(m7cat)$fstatistic[1] > qf(.95,summary(m7cat)$fstatistic[2],summary(m7cat)$fstatistic[3]))
{lm7cat[i]=1}

if (summary(m10cont)$fstatistic[1] > qf(.95,summary(m10cont)$fstatistic[2],summary(m10cont)$fstatistic[3]))
{lm10cont[i]=1}
if (summary(m10cat)$fstatistic[1] > qf(.95,summary(m10cat)$fstatistic[2],summary(m10cat)$fstatistic[3]))
{lm10cat[i]=1}
```

```
}

#getting the percentages
lper2cont=sum(lm2cont)/40
lper2cat=sum(lm2cat)/40

lper3cont=sum(lm3cont)/40
lper3cat=sum(lm3cat)/40

lper4cont=sum(lm4cont)/40
lper4cat=sum(lm4cat)/40

lper5cont=sum(lm5cont)/40
lper5cat=sum(lm5cat)/40

lper7cont=sum(lm7cont)/40
lper7cat=sum(lm7cat)/40

lper10cont=sum(lm10cont)/40
lper10cat=sum(lm10cat)/40

write.csv(c(lper2cont,lper3cont,lper4cont,lper5cont,lper7cont,lper10cont,lper2cat,lper3cat,lper4cat,lper5cat,
lper7cat,lper10cat),file="lper0.csv")

#generating sampling distributions of the slope
slopeboth=cbind(lslopetwo,lslopethree,lslopefour,lslopefive,lslopeseven,lslopeten,lslopetwo2,lslopethree2,lslopefou
r2,lslopefive2,lslopeseven2,lslopeten2)
slpnames=c("Linear 50, 2 category","Linear 50, 3 category","Linear 50, 4 category","Linear 50, 5 category","Linear
50, 7 category","Linear 50, 10 category",
"Linear 5, 2 category","Linear 5, 3 category","Linear 5, 4 category","Linear 5, 5 category","Linear 5, 7
category","Linear 5, 10 category")

png("lhistslopeszero%02d.png")  #saving sampling distributions of slopes to an external file
for (i in 1:12){slopehist(shapelevel=slopeboth[,i],title=slpnames[i])}
dev.off()




################################################################################
################################################################################




#Part 7:  Exp

#set exp probabilities for each section (matrix format with labels)
exp=as.matrix(c((1/3),0,0,0,0,0,0,0,0,(1/7),(2/7),0,0,0,0,0,0,0,(1/15),(2/15),(4/15),0,0,0,0,0,0,(1/31),(2/31),(4/31),(8/
31),0,0,0,0,0,
(1/127),(2/127),(4/127),(8/127),(16/127),(32/127),0,0,0,(1/1023),(2/1023),(4/1023),(8/1023),(16/1023),(32/1023),(6
4/1023),(128/1023),(256/1023)))
dim(exp)=c(9,6)
```

```
colnames(exp)=c("Exp2", "Exp3", "Exp4", "Exp5", "Exp7", "Exp10")

#create empty matrices for ordered categorical “exp” variables
etwo=matrix(0,300,4000)
ethree=matrix(0,300,4000)
efour=matrix(0,300,4000)
efive=matrix(0,300,4000)
eseven=matrix(0,300,4000)
eten=matrix(0,300,4000)

etwo2=matrix(0,300,4000)
ethree2=matrix(0,300,4000)
efour2=matrix(0,300,4000)
efive2=matrix(0,300,4000)
eseven2=matrix(0,300,4000)
eten2=matrix(0,300,4000)

#with the 51 level x (exponential)

for (k in 1:6){
        if (k==1){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51e2[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e2[,j])+1)){etwo[i,j]=1}
else{etwo[i,j]=2}}}}


        if (k==2){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51e3[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e3[,j])+1)){ethree[i,j]=1}
else{

        if(ylatent51e3[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e3[,j])+1)){ethree[i,j]=2}
else{ethree[i,j]=3}}}}}


        if (k==3){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51e4[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e4[,j])+1)){efour[i,j]=1}
else{

        if(ylatent51e4[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e4[,j])+1)){efour[i,j]=2} else{

        if(ylatent51e4[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent51e4[,j])+1)){efour[i,j]=3}
else{efour[i,j]=4}}}}}}


        if (k==4){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51e5[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e5[,j])+1)){efive[i,j]=1}
else{
```

```
        if(ylatent51e5[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e5[,j])+1)){efive[i,j]=2} else{

        if(ylatent51e5[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent51e5[,j])+1)){efive[i,j]=3} else{

        if(ylatent51e5[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent51e5[,j])+1)){efive[i,j]=4}
else{efive[i,j]=5}}}}}}}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51e7[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e7[,j])+1)){eseven[i,j]=1}
else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e7[,j])+1)){eseven[i,j]=2} else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent51e7[,j])+1)){eseven[i,j]=3} else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent51e7[,j])+1)){eseven[i,j]=4}
else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k],0,sd(ylatent51e7[,j])+1)){eseve
n[i,j]=5}else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k],0,sd(ylatent51e7[,j])+
1)){eseven[i,j]=6}else{eseven[i,j]=7}}}}}}}}}


        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51e10[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=1}
else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=2} else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=3} else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=4}
else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k],0,sd(ylatent51e10[,j])+1)){ete
n[i,j]=5}else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k],0,sd(ylatent51e10[,j
])+1)){eten[i,j]=6}else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k],0,sd(ylate
nt51e10[,j])+1)){eten[i,j]=7}else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k]+exp[8,k],
0,sd(ylatent51e10[,j])+1)){eten[i,j]=8}else{
```

```
	if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k]+exp[8,k]+
exp[9,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=9}else{eten[i,j]=10}}}}}}}}}}}}

}

write.csv(etwo, file="etwo0.csv")
write.csv(ethree, file="ethree0.csv")
write.csv(efour, file="efour0.csv")
write.csv(efive, file="efive0.csv")
write.csv(eseven, file="eseven0.csv")
write.csv(eten, file="eten0.csv")



#with the 5 level x (exponential)

for (k in 1:6){
	if (k==1){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5e22[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e22[,j])+1)){etwo2[i,j]=1}
else{etwo2[i,j]=2}}}}


	if (k==2){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5e23[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e23[,j])+1)){ethree2[i,j]=1}
else{

	if(ylatent5e23[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e23[,j])+1)){ethree2[i,j]=2}
else{ethree2[i,j]=3}}}}}



	if (k==3){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5e24[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e24[,j])+1)){efour2[i,j]=1}
else{

	if(ylatent5e24[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e24[,j])+1)){efour2[i,j]=2} else{

	if(ylatent5e24[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent5e24[,j])+1)){efour2[i,j]=3}
else{efour2[i,j]=4}}}}}


	if (k==4){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5e25[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e25[,j])+1)){efive2[i,j]=1}
else{

	if(ylatent5e25[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e25[,j])+1)){efive2[i,j]=2} else{
```

```
        if(ylatent5e25[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent5e25[,j])+1)){efive2[i,j]=3} else{

        if(ylatent5e25[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent5e25[,j])+1)){efive2[i,j]=4}
else{efive2[i,j]=5}}}}}}}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5e27[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e27[,j])+1)){eseven2[i,j]=1}
else{

        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e27[,j])+1)){eseven2[i,j]=2} else{

        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent5e27[,j])+1)){eseven2[i,j]=3} else{

        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent5e27[,j])+1)){eseven2[i,j]=4}
else{

        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k],0,sd(ylatent5e27[,j])+1)){eseve
n2[i,j]=5}else{

        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k],0,sd(ylatent5e27[,j])+
1)){eseven2[i,j]=6}else{eseven2[i,j]=7}}}}}}}}}


        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5e210[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=1}
else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=2} else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=3} else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=4}
else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k],0,sd(ylatent5e210[,j])+1)){ete
n2[i,j]=5}else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k],0,sd(ylatent5e210[,j
])+1)){eten2[i,j]=6}else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k],0,sd(ylate
nt5e210[,j])+1)){eten2[i,j]=7}else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k]+exp[8,k],
0,sd(ylatent5e210[,j])+1)){eten2[i,j]=8}else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k]+exp[8,k]+
exp[9,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=9}else{eten2[i,j]=10}}}}}}}}}}
```

```
}

write.csv(etwo2, file="etwo02.csv")
write.csv(ethree2, file="ethree02.csv")
write.csv(efour2, file="efour02.csv")
write.csv(efive2, file="efive02.csv")
write.csv(eseven2, file="eseven02.csv")
write.csv(eten2, file="eten02.csv")


#Get percent significant for each level

#####NOTE:  WILL NEED TO ADJUST THIS FOR THE CORRECT NEMBER OF CUTS, DISCESS WITH
BELLO, CONFLICTING NOTES

#vectors of zeros to hold 0,1 values, 1 if model had significant slope
#em2 -- eniform model 2 level dependent variable cont is the 100 level x, cat is the 5 level
em2cont=vector("numeric",4000)
em3cont=vector("numeric",4000)
em4cont=vector("numeric",4000)
em5cont=vector("numeric",4000)
em7cont=vector("numeric",4000)
em10cont=vector("numeric",4000)

em2cat=em2cont
em3cat=em2cont
em4cat=em2cont
em5cat=em2cont
em7cat=em2cont
em10cat=em2cont

#vectors of zero to hold estimates of slope
eslopeone=vector("numeric",4000)
eslopeone2=vector("numeric",4000)
eslopetwo=vector("numeric",4000)
eslopetwo2=vector("numeric",4000)
eslopethree=vector("numeric",4000)
eslopethree2=vector("numeric",4000)
eslopefour=vector("numeric",4000)
eslopefour2=vector("numeric",4000)
eslopefive=vector("numeric",4000)
eslopefive2=vector("numeric",4000)
eslopeseven=vector("numeric",4000)
eslopeseven2=vector("numeric",4000)
eslopeten=vector("numeric",4000)
eslopeten2=vector("numeric",4000)

for (i in 1:4000) {

#running the linear models
#none of the cat ones are relevant yet and will reqeire a variable change to be relevant
m2cont=lm(etwo[,i]~xlat[,1])
m2cat=lm(etwo2[,i]~xlat2[,1])
m3cont=lm(ethree[,i]~xlat[,1])
m3cat=lm(ethree2[,i]~xlat2[,1])
```

```
m4cont=lm(efour[,i]~xlat[,1])
m4cat=lm(efour2[,i]~xlat2[,1])
m5cont=lm(efive[,i]~xlat[,1])
m5cat=lm(efive2[,i]~xlat2[,1])
m7cont=lm(eseven[,i]~xlat[,1])
m7cat=lm(eseven2[,i]~xlat2[,1])
m10cont=lm(eten[,i]~xlat[,1])
m10cat=lm(eten2[,i]~xlat2[,1])

#grabbing the slopes
eslopetwo[i]=m2cont$coefficients[2]
eslopetwo2[i]=m2cat$coefficients[2]
eslopethree[i]=m3cont$coefficients[2]
eslopethree2[i]=m3cat$coefficients[2]
eslopefour[i]=m4cont$coefficients[2]
eslopefour2[i]=m4cat$coefficients[2]
eslopefive[i]=m5cont$coefficients[2]
eslopefive2[i]=m5cat$coefficients[2]
eslopeseven[i]=m7cont$coefficients[2]
eslopeseven2[i]=m7cat$coefficients[2]
eslopeten[i]=m10cont$coefficients[2]
eslopeten2[i]=m10cat$coefficients[2]


#counting how many of them are significant
if (summary(m2cont)$fstatistic[1] > qf(.95,summary(m2cont)$fstatistic[2],summary(m2cont)$fstatistic[3]))
{em2cont[i]=1}
if (summary(m2cat)$fstatistic[1] > qf(.95,summary(m2cat)$fstatistic[2],summary(m2cat)$fstatistic[3]))
{em2cat[i]=1}

if (summary(m3cont)$fstatistic[1] > qf(.95,summary(m3cont)$fstatistic[2],summary(m3cont)$fstatistic[3]))
{em3cont[i]=1}
if (summary(m3cat)$fstatistic[1] > qf(.95,summary(m3cat)$fstatistic[2],summary(m3cat)$fstatistic[3]))
{em3cat[i]=1}

if (summary(m4cont)$fstatistic[1] > qf(.95,summary(m4cont)$fstatistic[2],summary(m4cont)$fstatistic[3]))
{em4cont[i]=1}
if (summary(m4cat)$fstatistic[1] > qf(.95,summary(m4cat)$fstatistic[2],summary(m4cat)$fstatistic[3]))
{em4cat[i]=1}

if (summary(m5cont)$fstatistic[1] > qf(.95,summary(m5cont)$fstatistic[2],summary(m5cont)$fstatistic[3]))
{em5cont[i]=1}
if (summary(m5cat)$fstatistic[1] > qf(.95,summary(m5cat)$fstatistic[2],summary(m5cat)$fstatistic[3]))
{em5cat[i]=1}

if (summary(m7cont)$fstatistic[1] > qf(.95,summary(m7cont)$fstatistic[2],summary(m7cont)$fstatistic[3]))
{em7cont[i]=1}
if (summary(m7cat)$fstatistic[1] > qf(.95,summary(m7cat)$fstatistic[2],summary(m7cat)$fstatistic[3]))
{em7cat[i]=1}

if (summary(m10cont)$fstatistic[1] > qf(.95,summary(m10cont)$fstatistic[2],summary(m10cont)$fstatistic[3]))
{em10cont[i]=1}
if (summary(m10cat)$fstatistic[1] > qf(.95,summary(m10cat)$fstatistic[2],summary(m10cat)$fstatistic[3]))
{em10cat[i]=1}
}
```

```
#getting the percentages
eper2cont=sum(em2cont)/40
eper2cat=sum(em2cat)/40

eper3cont=sum(em3cont)/40
eper3cat=sum(em3cat)/40

eper4cont=sum(em4cont)/40
eper4cat=sum(em4cat)/40

eper5cont=sum(em5cont)/40
eper5cat=sum(em5cat)/40

eper7cont=sum(em7cont)/40
eper7cat=sum(em7cat)/40

eper10cont=sum(em10cont)/40
eper10cat=sum(em10cat)/40

write.csv(c(eper2cont,eper3cont,eper4cont,eper5cont,eper7cont,eper10cont,
eper2cat,eper3cat,eper4cat,eper5cat,eper7cat,eper10cat),file="eper0.csv")

#looking at the sampling distributions of slope
slopeboth=cbind(eslopetwo,eslopethree,eslopefour,eslopefive,eslopeseven,eslopeten,eslopetwo2,eslopethree2,eslope
four2,eslopefive2,eslopeseven2,eslopeten2)
slpnames=c("Exponential 50, 2 category","Exponential 50, 3 category","Exponential 50, 4 category","Exponential
50, 5 category","Exponential 50, 7 category","Exponential 50, 10 category",
"Exponential 5, 2 category","Exponential 5, 3 category","Exponential 5, 4 category","Exponential 5, 5
category","Exponential 5, 7 category","Exponential 5, 10 category")

png("ehistslopeszero%02d.png")  #saving sampling distributions of slopes to an external file
for (i in 1:12){slopehist(shapelevel=slopeboth[,i],title=slpnames[i])}
dev.off()




################################################################################
################################################################################



#Part 8:  Bell

#set bp probabilities for each section (matrix format with labels)
bp=as.matrix(c(.5,0,0,0,0,0,0,0,0,
          (1/4),(2/4),0,0,0,0,0,0,0,
          (1/6),(2/6),(2/6),0,0,0,0,0,0,
         (1/9),(2/9),(3/9),(2/9),0,0,0,0,0,
         (1/16),(2/16),(3/16),(4/16),(3/16),(2/16),0,0,0,
                  (1/30),(2/30),(3/30),(4/30),(5/30),(5/30),(4/30),(3/30),(2/30)))
dim(bp)=c(9,6)
```

```
colnames(bp)=c("Bell2", "Bell3", "Bell4", "Bell5", "Bell7", "Bell10")

#create empty matrices for ordered categorical bell shaped variables
btwo=matrix(0,300,4000)
bthree=matrix(0,300,4000)
bfour=matrix(0,300,4000)
bfive=matrix(0,300,4000)
bseven=matrix(0,300,4000)
bten=matrix(0,300,4000)

btwo2=matrix(0,300,4000)
bthree2=matrix(0,300,4000)
bfour2=matrix(0,300,4000)
bfive2=matrix(0,300,4000)
bseven2=matrix(0,300,4000)
bten2=matrix(0,300,4000)

#with the 51 level x (bell)

for (k in 1:6){
	if (k==1){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51b2[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b2[,j])+1)){btwo[i,j]=1}
else{btwo[i,j]=2}}}}


	if (k==2){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51b3[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b3[,j])+1)){bthree[i,j]=1}
else{

	if(ylatent51b3[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b3[,j])+1)){bthree[i,j]=2}
else{bthree[i,j]=3}}}}}


	if (k==3){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51b4[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b4[,j])+1)){bfour[i,j]=1} else{

	if(ylatent51b4[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b4[,j])+1)){bfour[i,j]=2} else{

	if(ylatent51b4[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent51b4[,j])+1)){bfour[i,j]=3}
else{bfour[i,j]=4}}}}}}


	if (k==4){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51b5[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b5[,j])+1)){bfive[i,j]=1} else{

	if(ylatent51b5[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b5[,j])+1)){bfive[i,j]=2} else{
```

```
        if(ylatent51b5[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent51b5[,j])+1)){bfive[i,j]=3} else{

        if(ylatent51b5[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent51b5[,j])+1)){bfive[i,j]=4}
else{bfive[i,j]=5}}}}}}}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51b7[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]=1}
else{

        if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]=2} else{

        if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]=3} else{

        if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]=4}
else{

        if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]
=5}else{

        if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k],0,sd(ylatent51b7[,j])+1)){bs
even[i,j]=6}else{bseven[i,j]=7}}}}}}}}}


        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51b10[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]=1}
else{

        if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]=2} else{

        if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]=3} else{

        if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]=4}
else{

        if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]
=5}else{

        if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k],0,sd(ylatent51b10[,j])+1))
{bten[i,j]=6}else{

        if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k],0,sd(ylatent51b10
[,j])+1)){bten[i,j]=7}else{

        if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k]+bp[8,k],0,sd(ylate
nt51b10[,j])+1)){bten[i,j]=8}else{

        if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k]+bp[8,k]+bp[9,k],
0,sd(ylatent51b10[,j])+1)){bten[i,j]=9}else{bten[i,j]=10}}}}}}}}}}}
```

```
}

write.csv(btwo, file="btwo0.csv")
write.csv(bthree, file="bthree0.csv")
write.csv(bfour, file="bfour0.csv")
write.csv(bfive, file="bfive0.csv")
write.csv(bseven, file="bseven0.csv")
write.csv(bten, file="bten0.csv")



#with the 5 level x (bell)

for (k in 1:6){
	if (k==1){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5b22[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b22[,j])+1)){btwo2[i,j]=1}
else{btwo2[i,j]=2}}}}


	if (k==2){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5b23[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b23[,j])+1)){bthree2[i,j]=1}
else{

	if(ylatent5b23[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b23[,j])+1)){bthree2[i,j]=2}
else{bthree2[i,j]=3}}}}}



	if (k==3){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5b24[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b24[,j])+1)){bfour2[i,j]=1}
else{

	if(ylatent5b24[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b24[,j])+1)){bfour2[i,j]=2} else{

	if(ylatent5b24[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent5b24[,j])+1)){bfour2[i,j]=3}
else{bfour2[i,j]=4}}}}}}



	if (k==4){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5b25[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b25[,j])+1)){bfive2[i,j]=1}
else{

	if(ylatent5b25[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b25[,j])+1)){bfive2[i,j]=2} else{

	if(ylatent5b25[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent5b25[,j])+1)){bfive2[i,j]=3} else{
```

```
        if(ylatent5b25[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent5b25[,j])+1)){bfive2[i,j]=4}
else{bfive2[i,j]=5}}}}}}}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5b27[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j]=1}
else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j]=2} else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j]=3} else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j]=4}
else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j
]=5}else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k],0,sd(ylatent5b27[,j])+1)){bs
even2[i,j]=6}else{bseven2[i,j]=7}}}}}}}}}


        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5b210[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j]=1}
else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j]=2} else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j]=3} else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j]=4}
else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j
]=5}else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k],0,sd(ylatent5b210[,j])+1))
{bten2[i,j]=6}else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k],0,sd(ylatent5b210
[,j])+1)){bten2[i,j]=7}else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k]+bp[8,k],0,sd(ylate
nt5b210[,j])+1)){bten2[i,j]=8}else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k]+bp[8,k]+bp[9,k],
0,sd(ylatent5b210[,j])+1)){bten2[i,j]=9}else{bten2[i,j]=10}}}}}}}}}}}

}
```

```
write.csv(btwo2, file="btwo2.csv")
write.csv(bthree2, file="bthree2.csv")
write.csv(bfour2, file="bfour2.csv")
write.csv(bfive2, file="bfive2.csv")
write.csv(bseven2, file="bseven2.csv")
write.csv(bten2, file="bten2.csv")


#Get percent significant for each level
#vectors of zeros to hold 0,1 values, 1 if model had significant slope
#um2 -- uniform model 2 level dependent variable cont is the 100 level x, cat is the 5 level
bm2cont=vector("numeric",4000)
bm3cont=vector("numeric",4000)
bm4cont=vector("numeric",4000)
bm5cont=vector("numeric",4000)
bm7cont=vector("numeric",4000)
bm10cont=vector("numeric",4000)

bm2cat=bm2cont
bm3cat=bm2cont
bm4cat=bm2cont
bm5cat=bm2cont
bm7cat=bm2cont
bm10cat=bm2cont

#vectors of zero to hold estimates of slope
bslopeone=vector("numeric",4000)
bslopeone2=vector("numeric",4000)
bslopetwo=vector("numeric",4000)
bslopetwo2=vector("numeric",4000)
bslopethree=vector("numeric",4000)
bslopethree2=vector("numeric",4000)
bslopefour=vector("numeric",4000)
bslopefour2=vector("numeric",4000)
bslopefive=vector("numeric",4000)
bslopefive2=vector("numeric",4000)
bslopeseven=vector("numeric",4000)
bslopeseven2=vector("numeric",4000)
bslopeten=vector("numeric",4000)
bslopeten2=vector("numeric",4000)


for (i in 1:4000) {

#running the linear models
#none of the cat ones are relevant yet and will require a variable change to be relevant
m2cont=lm(btwo[,i]~xlat[,1])
m2cat=lm(btwo2[,i]~xlat2[,1])
m3cont=lm(bthree[,i]~xlat[,1])
m3cat=lm(bthree2[,i]~xlat2[,1])
m4cont=lm(bfour[,i]~xlat[,1])
m4cat=lm(bfour2[,i]~xlat2[,1])
m5cont=lm(bfive[,i]~xlat[,1])
m5cat=lm(bfive2[,i]~xlat2[,1])
m7cont=lm(bseven[,i]~xlat[,1])
```

```
m7cat=lm(bseven2[,i]~xlat2[,1])
m10cont=lm(bten[,i]~xlat[,1])
m10cat=lm(bten2[,i]~xlat2[,1])

#grabbing the slopes
bslopetwo[i]=m2cont$coefficients[2]
bslopetwo2[i]=m2cat$coefficients[2]
bslopethree[i]=m3cont$coefficients[2]
bslopethree2[i]=m3cat$coefficients[2]
bslopefour[i]=m4cont$coefficients[2]
bslopefour2[i]=m4cat$coefficients[2]
bslopefive[i]=m5cont$coefficients[2]
bslopefive2[i]=m5cat$coefficients[2]
bslopeseven[i]=m7cont$coefficients[2]
bslopeseven2[i]=m7cat$coefficients[2]
bslopeten[i]=m10cont$coefficients[2]
bslopeten2[i]=m10cat$coefficients[2]


#counting how many of them are significant
if (summary(m2cont)$fstatistic[1] > qf(.95,summary(m2cont)$fstatistic[2],summary(m2cont)$fstatistic[3]))
{bm2cont[i]=1}
if (summary(m2cat)$fstatistic[1] > qf(.95,summary(m2cat)$fstatistic[2],summary(m2cat)$fstatistic[3]))
{bm2cat[i]=1}

if (summary(m3cont)$fstatistic[1] > qf(.95,summary(m3cont)$fstatistic[2],summary(m3cont)$fstatistic[3]))
{bm3cont[i]=1}
if (summary(m3cat)$fstatistic[1] > qf(.95,summary(m3cat)$fstatistic[2],summary(m3cat)$fstatistic[3]))
{bm3cat[i]=1}

if (summary(m4cont)$fstatistic[1] > qf(.95,summary(m4cont)$fstatistic[2],summary(m4cont)$fstatistic[3]))
{bm4cont[i]=1}
if (summary(m4cat)$fstatistic[1] > qf(.95,summary(m4cat)$fstatistic[2],summary(m4cat)$fstatistic[3]))
{bm4cat[i]=1}

if (summary(m5cont)$fstatistic[1] > qf(.95,summary(m5cont)$fstatistic[2],summary(m5cont)$fstatistic[3]))
{bm5cont[i]=1}
if (summary(m5cat)$fstatistic[1] > qf(.95,summary(m5cat)$fstatistic[2],summary(m5cat)$fstatistic[3]))
{bm5cat[i]=1}

if (summary(m7cont)$fstatistic[1] > qf(.95,summary(m7cont)$fstatistic[2],summary(m7cont)$fstatistic[3]))
{bm7cont[i]=1}
if (summary(m7cat)$fstatistic[1] > qf(.95,summary(m7cat)$fstatistic[2],summary(m7cat)$fstatistic[3]))
{bm7cat[i]=1}

if (summary(m10cont)$fstatistic[1] > qf(.95,summary(m10cont)$fstatistic[2],summary(m10cont)$fstatistic[3]))
{bm10cont[i]=1}
if (summary(m10cat)$fstatistic[1] > qf(.95,summary(m10cat)$fstatistic[2],summary(m10cat)$fstatistic[3]))
{bm10cat[i]=1}
}

#getting the percentages
bper2cont=sum(bm2cont)/40
bper2cat=sum(bm2cat)/40
```

```
bper3cont=sum(bm3cont)/40
bper3cat=sum(bm3cat)/40

bper4cont=sum(bm4cont)/40
bper4cat=sum(bm4cat)/40

bper5cont=sum(bm5cont)/40
bper5cat=sum(bm5cat)/40

bper7cont=sum(bm7cont)/40
bper7cat=sum(bm7cat)/40

bper10cont=sum(bm10cont)/40
bper10cat=sum(bm10cat)/40

write.csv(c(bper2cont,bper3cont,bper4cont,bper5cont,bper7cont,bper10cont,bper2cat,bper3cat,
bper4cat,bper5cat,bper7cat,bper10cat),file="bper0.csv")

#looking at the sampling distributions of slope
slopeboth=cbind(bslopetwo,bslopethree,bslopefour,bslopefive,bslopeseven,bslopeten,bslopetwo2,bslopethree2,bslop
efour2,bslopefive2,bslopeseven2,bslopeten2)
slpnames=c("Bell 50, 2 category","Bell 50, 3 category","Bell 50, 4 category","Bell 50, 5 category","Bell 50, 7
category","Bell 50, 10 category",
"Bell 5, 2 category","Bell 5, 3 category","Bell 5, 4 category","Bell 5, 5 category","Bell 5, 7 category","Bell 5, 10
category")

png("bhistslopeszero%02d.png")  #saving sampling distributions of slopes to an external file
for (i in 1:12){slopehist(shapelevel=slopeboth[,i],title=slpnames[i])}
dev.off()



################################################################################
################################################################################


#Report Type I error pictorally

levels=c(2,3,4,5,7,10)  #creating a variable for x axis of plot

u5=c(uper2cat,uper3cat,uper4cat,uper5cat,uper7cat,uper10cat)/100 #making a vector of my αs
u50=c(uper2cont,uper3cont,uper4cont,uper5cont,uper7cont,uper10cont)/100
l5=c(lper2cat,lper3cat,lper4cat,lper5cat,lper7cat,lper10cat)/100 #making a vector of my αs
l50=c(lper2cont,lper3cont,lper4cont,lper5cont,lper7cont,lper10cont)/100
e5=c(eper2cat,eper3cat,eper4cat,eper5cat,eper7cat,eper10cat)/100 #making a vector of my αs
e50=c(eper2cont,eper3cont,eper4cont,eper5cont,eper7cont,eper10cont)/100
b5=c(bper2cat,bper3cat,bper4cat,bper5cat,bper7cat,bper10cat)/100 #making a vector of my αs
b50=c(bper2cont,bper3cont,bper4cont,bper5cont,bper7cont,bper10cont)/100

#for a given shape and xlat combo, plot the type I eror
#colors and types of line will need adjusting after discussion

byshape50=cbind(u50,l50,e50,b50)  #making a matrix of ys for matplot
byshape5=cbind(u5,l5,e5,b5)
```

```
type1storage=cbind(byshape50,byshape5)  #making a matrix to store as a csv
rownames(type1storage)=c("2-level","3-level","4-level","5-level","7-level","10-level")
write.csv(type1storage,file="typeIerror.csv")

png("byshapeszeroαaxis%02d.png")  #saving byshapes to an external file

#plot for the 51 level scenario
matplot(x=levels, y=byshape50, type="l", ylim=c(.02,.08),ylab=expression(α), xlab="Number of Ordered
Categories",lty=2:5, col=2:5,
lwd=2, main="", font=1, font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abline(h=.05,lty=1,col=1)
abline(h=.04325 , lty=9, col=1)
abline(h=.05675, lty=9, col=1)
legend("topright", legend=c(expression(paste("Nominal ",α)),expression(paste("Bound on
",α)),"Uniform","Belled","Triangular","Exponential"), lty=c(1,9,2,5,3,4), col=c(1,1,2,5,3,4))

#plot for the 5 level scenario
matplot(x=levels, y=byshape5, type="l", ylim=c(.02,.08),ylab=expression(α), xlab="Number of Ordered
Categories",lty=2:5, col=2:5,
lwd=2, main="", font=1, font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abline(h=.05,lty=1,col=1)
abline(h=.04325 , lty=9, col=1)
abline(h=.05675, lty=9, col=1)
legend("topright", legend=c(expression(paste("Nominal ",α)),expression(paste("Bound on
",α)),"Uniform","Belled","Triangular","Exponential"), lty=c(1,9,2,5,3,4), col=c(1,1,2,5,3,4))

dev.off()


#looking at each shapes type I error by level

png("bylevelpershapezero%02d.png")  #saving bylevels to an external file

matplot(x=levels, y=cbind(u50,u5), type="l", ylim=c(0,.1),ylab=mytype, xlab="Number of Ordered Categories",
lty=c(3,4),
col=c(3,4),lwd=2,main="",font=1,font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abline(h=.05,lty=1,col=1)
abline(h=.04325 , lty=9, col=1)
abline(h=.05675, lty=9, col=1)
legend("bottomright", legend=c(expression(paste("Nominal ",α)),expression(paste("Bound on
",α)),expression(paste("OLSLR ", x^(51))),
expression(paste("OLSLR ",x^(5)))),lty=c(1,9,3,4),col=c(1,1,3,4))

matplot(x=levels, y=cbind(l50,l5), type="l", ylim=c(0,.1),ylab=mytype, xlab="Number of Ordered Categories",
lty=c(3,4),
col=c(3,4),lwd=2,main="",font=1,font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abline(h=.05,lty=1,col=1)
abline(h=.04325 , lty=9, col=1)
abline(h=.05675, lty=9, col=1)
```

```
legend("bottomright", legend=c(expression(paste("Nominal ",α)),expression(paste("Bound on
",α)),expression(paste("OLSLR ", x^(51))),
expression(paste("OLSLR ",x^(5)))),lty=c(1,9,3,4),col=c(1,1,3,4))

matplot(x=levels, y=cbind(e50,e5), type="l", ylim=c(0,.1),ylab=mytype, xlab="Number of Ordered Categories",
lty=c(3,4),col=c(3,4),lwd=2,main="",font=1,font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abline(h=.05,lty=1,col=1)
abline(h=.04325 , lty=9, col=1)
abline(h=.05675, lty=9, col=1)
legend("bottomright", legend=c(expression(paste("Nominal ",α)),expression(paste("Bound on
",α)),expression(paste("OLSLR ", x^(51))),
expression(paste("OLSLR ",x^(5)))),lty=c(1,9,3,4),col=c(1,1,3,4))

matplot(x=levels, y=cbind(b50,b5), type="l", ylim=c(0,.1),ylab=mytype, xlab="Number of Ordered Categories",
lty=c(3,4),col=c(3,4),lwd=2,main="",font=1,font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abline(h=.05,lty=1,col=1)
abline(h=.04325 , lty=9, col=1)
abline(h=.05675, lty=9, col=1)
legend("bottomright", legend=c(expression(paste("Nominal ",α)),expression(paste("Bound on
",α)),expression(paste("OLSLR ", x^(51))),
expression(paste("OLSLR ",x^(5)))),lty=c(1,9,3,4),col=c(1,1,3,4))

dev.off()
```

Appendix C - Appendix C: Simulation Code for $\beta^* = 1$ Condition

```
##############################################################################
#Simulating the various cut and distribution scenarios
#I am going to create y variables based on x variables
#There will be a one to one relationship between x and y (slope=1) plus some
#random noise.  This will allow me to assume an underlying dependent and
#independent variable which I may then cut anyway I please to make ordered categorical
#variables with any number of categories that maintain this underlying
#relationship
#
#
##############################################################################
#
#
#Part 1:  Create x variables
#Part 2:  Create error term
#Part 3:  Create underlying continuous latent variable y
#Part 4:  Run triangularear models and view R2
#Part 5:  Calculated power and percent models significant
#Part 6:  Uniformorm
#Part 7:  Triangular
#Part 8:  Exponential
#Part 9:  Bell
#Part 10: Report power pictorally
#Part 11: Compute intervals
#
#
#Parts 6-9 include the creation of the ordered categorical variable, running of probit
#models, ascertation of empirical power and production of various graphs.
#These sections are nearly identical, that is, looking at the section for
#Uniform will tell you exactly what happens in Triangular, Exponential and Bell,
#outside of the probabilities used to create the ordered categorical variables.
##############################################################################
##############################################################################

#Part 1:  Create x variables

#creating the predictor variable with 51 levels
xlatnames="xlat1111"   #making xlatnames a character variable
for (i in 1:4000) {xlatnames[i]=paste("xlat",i,sep="")}  #vector of names
x1=round(-49.5:49.5)  #making a vector of -50 to 50
xlat=rep(x1,12000)  #making a vector of -50 to 50 repeated 120000 times
dim(xlat)=c(300,4000)  #making a matrix of 4000 identical variables of length 300
colnames(xlat)=xlatnames  #assigning the x variables names

#creating the predictor matrix with 5 levels
x2=c(rep(-50,60),rep(-25,60),rep(0,60),rep(25,60),rep(50,60))
##could have used rep(seq(20,100,20),240000)###
xlat2=rep(x2,4000)
dim(xlat2)=c(300,4000)  #making a matrix of 4000 identical variables of length 300
colnames(xlat2)=xlatnames  #assigning the x variables names
```

```
################################################################################

#Part 2:  Create the error terms (N(0,stdev51^2) and N(0,stdev5^2))

#51 Level Scenario
stdev51=177.9  #saving standard deviation for power calculation below
set.seed(54)  #set the randomization to a specific point

error51x1=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x1)=c(300,4000)

error51x2=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x2)=c(300,4000)

error51x3=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x3)=c(300,4000)

error51x4=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x4)=c(300,4000)

error51x5=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x5)=c(300,4000)

error51x6=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x6)=c(300,4000)

error51x7=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x7)=c(300,4000)

error51x8=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x8)=c(300,4000)

error51x9=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x9)=c(300,4000)

error51x10=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x10)=c(300,4000)

error51x11=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x11)=c(300,4000)

error51x12=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x12)=c(300,4000)

error51x13=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x13)=c(300,4000)

error51x14=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x14)=c(300,4000)

error51x15=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x15)=c(300,4000)
```

```
error51x16=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x16)=c(300,4000)

error51x17=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x17)=c(300,4000)

error51x18=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x18)=c(300,4000)

error51x19=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x19)=c(300,4000)

error51x20=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x20)=c(300,4000)

error51x21=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x21)=c(300,4000)

error51x22=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x22)=c(300,4000)

error51x23=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x23)=c(300,4000)

error51x24=as.matrix(rnorm(1200000, 0, stdev51))   #matrix of values from N(0,351.5625)
dim(error51x24)=c(300,4000)



#5-Level Scenario
stdev5=217


error5x1=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x1)=c(300,4000)

error5x2=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x2)=c(300,4000)

error5x3=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x3)=c(300,4000)

error5x4=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x4)=c(300,4000)

error5x5=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x5)=c(300,4000)

error5x6=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x6)=c(300,4000)

error5x7=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x7)=c(300,4000)

error5x8=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
```

```
dim(error5x8)=c(300,4000)

error5x9=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x9)=c(300,4000)

error5x10=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x10)=c(300,4000)

error5x11=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x11)=c(300,4000)

error5x12=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x12)=c(300,4000)

error5x13=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x13)=c(300,4000)

error5x14=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x14)=c(300,4000)

error5x15=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x15)=c(300,4000)

error5x16=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x16)=c(300,4000)

error5x17=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x17)=c(300,4000)

error5x18=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x18)=c(300,4000)

error5x19=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x19)=c(300,4000)

error5x20=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x20)=c(300,4000)

error5x21=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x21)=c(300,4000)

error5x22=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x22)=c(300,4000)

error5x23=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x23)=c(300,4000)

error5x24=as.matrix(rnorm(1200000, 0, stdev5))   #matrix of values from N(0,529)
dim(error5x24)=c(300,4000)



################################################################################
```

```
#Part 3:  Create underlying continuous latent variables based on x
slope=1
mytype="Power"

#51-Level Scenario
ylatent51u2=slope*xlat+error51x1
ylatent51u3=slope*xlat+error51x2
ylatent51u4=slope*xlat+error51x3
ylatent51u5=slope*xlat+error51x4
ylatent51u7=slope*xlat+error51x5
ylatent51u10=slope*xlat+error51x6

ylatent51l2=slope*xlat+error51x7
ylatent51l3=slope*xlat+error51x8
ylatent51l4=slope*xlat+error51x9
ylatent51l5=slope*xlat+error51x10
ylatent51l7=slope*xlat+error51x11
ylatent51l10=slope*xlat+error51x12

ylatent51e2=slope*xlat+error51x13
ylatent51e3=slope*xlat+error51x14
ylatent51e4=slope*xlat+error51x15
ylatent51e5=slope*xlat+error51x16
ylatent51e7=slope*xlat+error51x17
ylatent51e10=slope*xlat+error51x18

ylatent51b2=slope*xlat+error51x19
ylatent51b3=slope*xlat+error51x20
ylatent51b4=slope*xlat+error51x21
ylatent51b5=slope*xlat+error51x22
ylatent51b7=slope*xlat+error51x23
ylatent51b10=slope*xlat+error51x24



#5-Level Scenario

ylatent5u22=slope*xlat2+error5x1
ylatent5u23=slope*xlat2+error5x2
ylatent5u24=slope*xlat2+error5x3
ylatent5u25=slope*xlat2+error5x4
ylatent5u27=slope*xlat2+error5x5
ylatent5u210=slope*xlat2+error5x6

ylatent5l22=slope*xlat2+error5x7
ylatent5l23=slope*xlat2+error5x8
ylatent5l24=slope*xlat2+error5x9
ylatent5l25=slope*xlat2+error5x10
ylatent5l27=slope*xlat2+error5x11
ylatent5l210=slope*xlat2+error5x12

ylatent5e22=slope*xlat2+error5x13
ylatent5e23=slope*xlat2+error5x14
ylatent5e24=slope*xlat2+error5x15
```

```
ylatent5e25=slope*xlat2+error5x16
ylatent5e27=slope*xlat2+error5x17
ylatent5e210=slope*xlat2+error5x18

ylatent5b22=slope*xlat2+error5x19
ylatent5b23=slope*xlat2+error5x20
ylatent5b24=slope*xlat2+error5x21
ylatent5b25=slope*xlat2+error5x22
ylatent5b27=slope*xlat2+error5x23
ylatent5b210=slope*xlat2+error5x24




################################################################################


#Part 4:  Looking at r-squared values to control noise (avg r-squared of latent variable approx .8)

#51-Level Scenario

rsq=vector("numeric",4000) #vector to hold r-squared values
fm1=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm1[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm2=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm2[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm3=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u4[,i]~xlat[,1])
```

```
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm3[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm4=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm4[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm5=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm6=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm6[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm7=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
```

```
m=lm(ylatent51l2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm7[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm8=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm8[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm9=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm9[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm10=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm10[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm11=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
```

```
for (i in 1:4000) {
m=lm(ylatent51l7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm11[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm12=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm12[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm13=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm13[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm14=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm14[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm15=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
```

```
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm15[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm16=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm16[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm17=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm17[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm18=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm18[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
```

```
fm19=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm19[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm20=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm20[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm21=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm21[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm22=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm22[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)
```

```
rsq=vector("numeric",4000) #vector to hold r-squared values
fm23=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm23[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)

rsq=vector("numeric",4000) #vector to hold r-squared values
fm24=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm24[i]=1}
rsq[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq)
sd(sl)
hist(sl)



#5-Level Scenario

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x1=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x1[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x2=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x2[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }
```

```
mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x3=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x3[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x4=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x4[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x5=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x5[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x6=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51u10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x6[i]=1}
rsq2[i]=summary(m)$r.squared
```

```
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x7=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x7[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x8=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x8[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x9=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x9[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x10=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x10[i]=1}
```

```
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x11=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x11[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x12=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51l10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x12[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x13=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x13[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x14=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
```

```
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x14[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x15=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x15[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x16=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x16[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x17=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x17[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x18=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51e10[,i]~xlat[,1])
```

```
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x18[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x19=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b2[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x19[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x20=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b3[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x20[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x21=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b4[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x21[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x22=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
```

```
m=lm(ylatent51b5[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x22[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x23=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b7[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x23[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)

rsq2=vector("numeric",4000) #vector to hold r-squared values
fm5x24=vector("numeric",4000)  #vectors to hold 0, 1 values   #1 is a correct decision, 0 is incorrect
sl=vector("numeric",4000)
for (i in 1:4000) {
m=lm(ylatent51b10[,i]~xlat[,1])
#fcalc>f(95% (lower tail), num df, den df)
if (summary(m)$fstatistic[1] > qf(.95,summary(m)$fstatistic[2],summary(m)$fstatistic[3])) {fm5x24[i]=1}
rsq2[i]=summary(m)$r.squared
sl[i]=m$coefficients[2]  }

mean(rsq2)
sd(sl)
hist(sl)


################################################################################


#Part 5:  Calculating what power should be and what it currently is on the latent models

#Calculated "true" power
ss51=sqrt((stdev51^2)/sum((xlat[,1]-mean(xlat[,1]))^2))  #standard deviation of the slope 51 level
ss5=sqrt((stdev5^2)/sum((xlat2[,1]-mean(xlat2[,1]))^2))  #standard deviation of the slope 5 level x

non51=slope/ss51  #noncentrality for 51 level
non5=slope/ss5    #noncentrality for 5 level

power=(1-pt(qt(.975,df=298,lower.tail=TRUE),df=298,ncp=non51,lower.tail=TRUE)-
pt(qt(.025,df=298,lower.tail=TRUE),df=298,ncp=non51,lower.tail=TRUE))
power2=(1-pt(qt(.975,df=298,lower.tail=TRUE),df=298,ncp=non5,lower.tail=TRUE)-
pt(qt(.025,df=298,lower.tail=TRUE),df=298,ncp=non5,lower.tail=TRUE))
```

```
powerdisplay=paste(power*100,"%",sep="")
power2display=paste(power2*100,"%",sep="")
powerdisplay
power2display

#empirical power
per1=sum(fm1)/40  #divide by 40 b/c times 100 in numerator
per5x1=sum(fm5x1)/40

per2=sum(fm2)/40  #divide by 40 b/c times 100 in numerator
per5x2=sum(fm5x2)/40

per3=sum(fm3)/40  #divide by 40 b/c times 100 in numerator
per5x3=sum(fm5x3)/40

per4=sum(fm4)/40  #divide by 40 b/c times 100 in numerator
per5x4=sum(fm5x4)/40

per5=sum(fm5)/40  #divide by 40 b/c times 100 in numerator
per5x5=sum(fm5x5)/40

per6=sum(fm6)/40  #divide by 40 b/c times 100 in numerator
per5x6=sum(fm5x6)/40

per7=sum(fm7)/40  #divide by 40 b/c times 100 in numerator
per5x7=sum(fm5x7)/40

per8=sum(fm8)/40  #divide by 40 b/c times 100 in numerator
per5x8=sum(fm5x8)/40

per9=sum(fm9)/40  #divide by 40 b/c times 100 in numerator
per5x9=sum(fm5x9)/40

per10=sum(fm10)/40  #divide by 40 b/c times 100 in numerator
per5x10=sum(fm5x10)/40

per11=sum(fm11)/40  #divide by 40 b/c times 100 in numerator
per5x11=sum(fm5x11)/40

per12=sum(fm12)/40  #divide by 40 b/c times 100 in numerator
per5x12=sum(fm5x12)/40

per13=sum(fm13)/40  #divide by 40 b/c times 100 in numerator
per5x13=sum(fm5x13)/40

per14=sum(fm14)/40  #divide by 40 b/c times 100 in numerator
per5x14=sum(fm5x14)/40

per15=sum(fm15)/40  #divide by 40 b/c times 100 in numerator
per5x15=sum(fm5x15)/40

per16=sum(fm16)/40  #divide by 40 b/c times 100 in numerator
per5x16=sum(fm5x16)/40

per17=sum(fm17)/40  #divide by 40 b/c times 100 in numerator
```

```
per5x17=sum(fm5x17)/40

per18=sum(fm18)/40  #divide by 40 b/c times 100 in numerator
per5x18=sum(fm5x18)/40

per19=sum(fm19)/40  #divide by 40 b/c times 100 in numerator
per5x19=sum(fm5x19)/40

per20=sum(fm20)/40  #divide by 40 b/c times 100 in numerator
per5x20=sum(fm5x20)/40

per21=sum(fm21)/40  #divide by 40 b/c times 100 in numerator
per5x21=sum(fm5x21)/40

per22=sum(fm22)/40  #divide by 40 b/c times 100 in numerator
per5x22=sum(fm5x22)/40

per23=sum(fm23)/40  #divide by 40 b/c times 100 in numerator
per5x23=sum(fm5x23)/40

per24=sum(fm24)/40  #divide by 40 b/c times 100 in numerator
per5x24=sum(fm5x24)/40


################################################################################
################################################################################


#initializing library that contains probit functions
library(MASS)


#Part 5:  Uniform

#set up probabilities for each section (matrix format with labels)
up=as.matrix(c(.5,0,0,0,0,0,0,0,0,
            (1/3),(1/3),0,0,0,0,0,0,0,
            .25,.25,.25,0,0,0,0,0,0,
           (1/5),(1/5),(1/5),(1/5),0,0,0,0,0,
           (1/7),(1/7),(1/7),(1/7),(1/7),(1/7),(1/7),0,0,0,
           (1/10),(1/10),(1/10),(1/10),(1/10),(1/10),(1/10),(1/10),(1/10),(1/10)))
dim(up)=c(9,6)
colnames(up)=c("Uniform2", "Uniform3", "Uniform4", "Uniform5", "Uniform7", "Uniform10")

#create empty matrices for ordered categorical uniformorm variables
utwo=matrix(0,300,4000)
uthree=matrix(0,300,4000)
ufour=matrix(0,300,4000)
ufive=matrix(0,300,4000)
useven=matrix(0,300,4000)
uten=matrix(0,300,4000)

utwo2=matrix(0,300,4000)
uthree2=matrix(0,300,4000)
ufour2=matrix(0,300,4000)
```

```
ufive2=matrix(0,300,4000)
useven2=matrix(0,300,4000)
uten2=matrix(0,300,4000)

#create empty vectors to hold counts of significant slopes in the probit models(1=sig 0=insig)
uprob2=vector("numeric",4000)
uprob3=uprob2
uprob4=uprob2
uprob5=uprob2
uprob7=uprob2
uprob10=uprob2

uprob22=vector("numeric",4000)
uprob32=uprob22
uprob42=uprob22
uprob52=uprob22
uprob72=uprob22
uprob102=uprob22


#with the 51 level x (uniform)

for (k in 1:6){
        if (k==1){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51u2[i,j]<=qnorm(up[1,k],0,sd(ylatent51u2[,j])+1)){utwo[i,j]=1}
else{utwo[i,j]=2}}
                        #running a binary probit using glm
                        glm.model=summary(glm(as.factor(utwo[,j])~xlat[,1],family=binomial("probit")))
                        if (glm.model$coefficients[2,4]<.05) {uprob2[j]=1}} #counting the significant slopes
                        probutwo=sum(uprob2)/4000} #compting empirical power for the probit models


        if (k==2){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51u3[i,j]<=qnorm(up[1,k],0,sd(ylatent51u3[,j])+1)){uthree[i,j]=1}
else{

        if(ylatent51u3[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u3[,j])+1)){uthree[i,j]=2} else{uthree[i,j]=3}}}
                        #running an ordered probit regression from the MASS package
                        polr.model=summary(polr(as.factor(uthree[,j])~xlat[,1],method="probit",Hess=T))
                        #squaring my T from the output and counting the significant slopes

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob3[j]=1}}
                        probuthree=sum(uprob3)/4000}  #calculating empirical power in the ordered probit
models


        if (k==3){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51u4[i,j]<=qnorm(up[1,k],0,sd(ylatent51u4[,j])+1)){ufour[i,j]=1} else{
```

```
        if(ylatent51u4[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u4[,j])+1)){ufour[i,j]=2} else{

        if(ylatent51u4[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent51u4[,j])+1)){ufour[i,j]=3}
else{ufour[i,j]=4}}}}
                        polr.model=summary(polr(as.factor(ufour[,j])~xlat[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob4[j]=1}}
                        probufour=sum(uprob4)/4000}


        if (k==4){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51u5[i,j]<=qnorm(up[1,k],0,sd(ylatent51u5[,j])+1)){ufive[i,j]=1} else{

        if(ylatent51u5[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u5[,j])+1)){ufive[i,j]=2} else{

        if(ylatent51u5[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent51u5[,j])+1)){ufive[i,j]=3} else{

        if(ylatent51u5[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent51u5[,j])+1)){ufive[i,j]=4}
else{ufive[i,j]=5}}}}}
                        polr.model=summary(polr(as.factor(ufive[,j])~xlat[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob5[j]=1}}
                        probufive=sum(uprob5)/4000}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51u7[i,j]<=qnorm(up[1,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]=1}
else{

        if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]=2} else{

        if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]=3} else{

        if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]=4}
else{

        if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k],0,sd(ylatent51u7[,j])+1)){useven[i,j]
=5}else{

        if(ylatent51u7[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k],0,sd(ylatent51u7[,j])+1)){us
even[i,j]=6}else{useven[i,j]=7}}}}}}}
                        polr.model=summary(polr(as.factor(useven[,j])~xlat[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob7[j]=1}}
                        probuseven=sum(uprob7)/4000}


        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
```

```
                                        if(ylatent51u10[i,j]<=qnorm(up[1,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]=1}
else{

        if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]=2} else{

        if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]=3} else{

        if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]=4}
else{

        if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k],0,sd(ylatent51u10[,j])+1)){uten[i,j]
=5}else{

        if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k],0,sd(ylatent51u10[,j])+1))
{uten[i,j]=6}else{

        if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k],0,sd(ylatent51u10
[,j])+1)){uten[i,j]=7}else{

        if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k]+up[8,k],0,sd(ylate
nt51u10[,j])+1)){uten[i,j]=8}else{

        if(ylatent51u10[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k]+up[8,k]+up[9,k],
0,sd(ylatent51u10[,j])+1)){uten[i,j]=9}else{uten[i,j]=10}}}}}}}}}}
                        polr.model=summary(polr(as.factor(uten[,j])~xlat[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob10[j]=1}}
                        probuten=sum(uprob10)/4000}

}

write.csv(utwo, file="utwo.csv")
write.csv(uthree, file="uthree.csv")
write.csv(ufour, file="ufour.csv")
write.csv(ufive, file="ufive.csv")
write.csv(useven, file="useven.csv")
write.csv(uten, file="uten.csv")

#with the 5 level x (uniformorm)

for (k in 1:6){
        if (k==1){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5u22[i,j]<=qnorm(up[1,k],0,sd(ylatent5u22[,j])+1)){utwo2[i,j]=1}
else{utwo2[i,j]=2}}
                        glm.model=summary(glm(as.factor(utwo2[,j])~xlat2[,1],family=binomial("probit")))
                        if (glm.model$coefficients[2,4]<.05) {uprob22[j]=1}}
                        probutwo2=sum(uprob22)/4000}


        if (k==2){
                for (j in 1:4000){
                        for (i in 1:300){
```

```
                                        if(ylatent5u23[i,j]<=qnorm(up[1,k],0,sd(ylatent5u23[,j])+1)){uthree2[i,j]=1}
else{

        if(ylatent5u23[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u23[,j])+1)){uthree2[i,j]=2}
else{uthree2[i,j]=3}}}
                                polr.model=summary(polr(as.factor(uthree2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob32[j]=1}}
                                probuthree2=sum(uprob32)/4000}



        if (k==3){
                for (j in 1:4000){
                        for (i in 1:300){
                                        if(ylatent5u24[i,j]<=qnorm(up[1,k],0,sd(ylatent5u24[,j])+1)){ufour2[i,j]=1}
else{

        if(ylatent5u24[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u24[,j])+1)){ufour2[i,j]=2} else{

        if(ylatent5u24[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent5u24[,j])+1)){ufour2[i,j]=3}
else{ufour2[i,j]=4}}}}
                                polr.model=summary(polr(as.factor(ufour2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob42[j]=1}}
                                probufour2=sum(uprob42)/4000}



        if (k==4){
                for (j in 1:4000){
                        for (i in 1:300){
                                        if(ylatent5u25[i,j]<=qnorm(up[1,k],0,sd(ylatent5u25[,j])+1)){ufive2[i,j]=1}
else{

        if(ylatent5u25[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u25[,j])+1)){ufive2[i,j]=2} else{

        if(ylatent5u25[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent5u25[,j])+1)){ufive2[i,j]=3} else{

        if(ylatent5u25[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent5u25[,j])+1)){ufive2[i,j]=4}
else{ufive2[i,j]=5}}}}}
                                polr.model=summary(polr(as.factor(ufive2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob52[j]=1}}
                                probufive2=sum(uprob52)/4000}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                        if(ylatent5u27[i,j]<=qnorm(up[1,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j]=1}
else{

        if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j]=2} else{
```

```
        if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j]=3} else{

        if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j]=4}
else{

        if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k],0,sd(ylatent5u27[,j])+1)){useven2[i,j
]=5}else{

        if(ylatent5u27[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k],0,sd(ylatent5u27[,j])+1)){us
even2[i,j]=6}else{useven2[i,j]=7}}}}}}}
                        polr.model=summary(polr(as.factor(useven2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob72[j]=1}}
                        probuseven2=sum(uprob72)/4000}

        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5u210[i,j]<=qnorm(up[1,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j]=1}
else{

        if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j]=2} else{

        if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j]=3} else{

        if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j]=4}
else{

        if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k],0,sd(ylatent5u210[,j])+1)){uten2[i,j
]=5}else{

        if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k],0,sd(ylatent5u210[,j])+1))
{uten2[i,j]=6}else{

        if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k],0,sd(ylatent5u210
[,j])+1)){uten2[i,j]=7}else{

        if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k]+up[8,k],0,sd(ylate
nt5u210[,j])+1)){uten2[i,j]=8}else{

        if(ylatent5u210[i,j]<=qnorm(up[1,k]+up[2,k]+up[3,k]+up[4,k]+up[5,k]+up[6,k]+up[7,k]+up[8,k]+up[9,k],
0,sd(ylatent5u210[,j])+1)){uten2[i,j]=9}else{uten2[i,j]=10}}}}}}}}}}
                        polr.model=summary(polr(as.factor(uten2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){uprob102[j]=1}}
                        probuten2=sum(uprob102)/4000}

}

write.csv(utwo2, file="utwo2.csv")
write.csv(uthree2, file="uthree2.csv")
write.csv(ufour2, file="ufour2.csv")
write.csv(ufive2, file="ufive2.csv")
write.csv(useven2, file="useven2.csv")
```

```
write.csv(uten2, file="uten2.csv")


#Get percent significant for each level

#vectors of zeros to hold 0,1 values, 1 if model had significant slope
#um2 -- uniform model 2 level dependent variable cont is the 100 level x, cat is the 5 level
um2cont=vector("numeric",4000)
um3cont=vector("numeric",4000)
um4cont=vector("numeric",4000)
um5cont=vector("numeric",4000)
um7cont=vector("numeric",4000)
um10cont=vector("numeric",4000)

um2cat=um2cont
um3cat=um2cont
um4cat=um2cont
um5cat=um2cont
um7cat=um2cont
um10cat=um2cont


#create empty vectors to store estimates of slope
uslopeone=vector("numeric",4000)
uslopeone2=vector("numeric",4000)
uslopetwo=vector("numeric",4000)
uslopetwo2=vector("numeric",4000)
uslopethree=vector("numeric",4000)
uslopethree2=vector("numeric",4000)
uslopefour=vector("numeric",4000)
uslopefour2=vector("numeric",4000)
uslopefive=vector("numeric",4000)
uslopefive2=vector("numeric",4000)
uslopeseven=vector("numeric",4000)
uslopeseven2=vector("numeric",4000)
uslopeten=vector("numeric",4000)
uslopeten2=vector("numeric",4000)


for (i in 1:4000) {

#running the triangularear models
#none of the cat ones are relevant yet and will require a variable change to be relevant
m2cont=lm(utwo[,i]~xlat[,1])
m2cat=lm(utwo2[,i]~xlat2[,1])
m3cont=lm(uthree[,i]~xlat[,1])
m3cat=lm(uthree2[,i]~xlat2[,1])
m4cont=lm(ufour[,i]~xlat[,1])
m4cat=lm(ufour2[,i]~xlat2[,1])
m5cont=lm(ufive[,i]~xlat[,1])
m5cat=lm(ufive2[,i]~xlat2[,1])
m7cont=lm(useven[,i]~xlat[,1])
m7cat=lm(useven2[,i]~xlat2[,1])
m10cont=lm(uten[,i]~xlat[,1])
m10cat=lm(uten2[,i]~xlat2[,1])
```

```
#grabbing the slopes
uslopetwo[i]=m2cont$coefficients[2]
uslopetwo2[i]=m2cat$coefficients[2]
uslopethree[i]=m3cont$coefficients[2]
uslopethree2[i]=m3cat$coefficients[2]
uslopefour[i]=m4cont$coefficients[2]
uslopefour2[i]=m4cat$coefficients[2]
uslopefive[i]=m5cont$coefficients[2]
uslopefive2[i]=m5cat$coefficients[2]
uslopeseven[i]=m7cont$coefficients[2]
uslopeseven2[i]=m7cat$coefficients[2]
uslopeten[i]=m10cont$coefficients[2]
uslopeten2[i]=m10cat$coefficients[2]



#counting how many of them are significant
if (summary(m2cont)$fstatistic[1] > qf(.95,summary(m2cont)$fstatistic[2],summary(m2cont)$fstatistic[3]))
{um2cont[i]=1}
if (summary(m2cat)$fstatistic[1] > qf(.95,summary(m2cat)$fstatistic[2],summary(m2cat)$fstatistic[3]))
{um2cat[i]=1}

if (summary(m3cont)$fstatistic[1] > qf(.95,summary(m3cont)$fstatistic[2],summary(m3cont)$fstatistic[3]))
{um3cont[i]=1}
if (summary(m3cat)$fstatistic[1] > qf(.95,summary(m3cat)$fstatistic[2],summary(m3cat)$fstatistic[3]))
{um3cat[i]=1}

if (summary(m4cont)$fstatistic[1] > qf(.95,summary(m4cont)$fstatistic[2],summary(m4cont)$fstatistic[3]))
{um4cont[i]=1}
if (summary(m4cat)$fstatistic[1] > qf(.95,summary(m4cat)$fstatistic[2],summary(m4cat)$fstatistic[3]))
{um4cat[i]=1}

if (summary(m5cont)$fstatistic[1] > qf(.95,summary(m5cont)$fstatistic[2],summary(m5cont)$fstatistic[3]))
{um5cont[i]=1}
if (summary(m5cat)$fstatistic[1] > qf(.95,summary(m5cat)$fstatistic[2],summary(m5cat)$fstatistic[3]))
{um5cat[i]=1}

if (summary(m7cont)$fstatistic[1] > qf(.95,summary(m7cont)$fstatistic[2],summary(m7cont)$fstatistic[3]))
{um7cont[i]=1}
if (summary(m7cat)$fstatistic[1] > qf(.95,summary(m7cat)$fstatistic[2],summary(m7cat)$fstatistic[3]))
{um7cat[i]=1}

if (summary(m10cont)$fstatistic[1] > qf(.95,summary(m10cont)$fstatistic[2],summary(m10cont)$fstatistic[3]))
{um10cont[i]=1}
if (summary(m10cat)$fstatistic[1] > qf(.95,summary(m10cat)$fstatistic[2],summary(m10cat)$fstatistic[3]))
{um10cat[i]=1}
}

#getting the percentages
uper2cont=sum(um2cont)/40
uper2cat=sum(um2cat)/40

uper3cont=sum(um3cont)/40
```

```
uper3cat=sum(um3cat)/40

uper4cont=sum(um4cont)/40
uper4cat=sum(um4cat)/40

uper5cont=sum(um5cont)/40
uper5cat=sum(um5cat)/40

uper7cont=sum(um7cont)/40
uper7cat=sum(um7cat)/40

uper10cont=sum(um10cont)/40
uper10cat=sum(um10cat)/40

write.csv(c(uper2cont,uper3cont,uper4cont,uper5cont,uper7cont,uper10cont,uper2cat,uper3cat,
uper4cat,uper5cat,uper7cat,uper10cat),file="uper.csv")

#getting histograms of the estimated standard errors

#define my basic histogram function
slopehist=function(shapelevel,title){
hist(shapelevel,main="",xlab=paste("Latent slope =",slope))
legend("topleft",legend=c(paste(attributes(summary(shapelevel))$names[1],summary(shapelevel)[1],sep=" = "),
paste(attributes(summary(shapelevel))$names[2],summary(shapelevel)[2],sep=" = "),
paste(attributes(summary(shapelevel))$names[3],summary(shapelevel)[3],sep=" = "),
paste(attributes(summary(shapelevel))$names[4],summary(shapelevel)[4],sep=" = "),
paste(attributes(summary(shapelevel))$names[5],summary(shapelevel)[5],sep=" = "),
paste(attributes(summary(shapelevel))$names[6],summary(shapelevel)[6],sep=" = ")))}

slopeuboth=cbind(uslopetwo,uslopethree,uslopefour,uslopefive,uslopeseven,uslopeten,uslopetwo2,uslopethree2,uslo
pefour2,uslopefive2,uslopeseven2,uslopeten2)
slpnames=c("Uniformorm 51, 2 category","Uniformorm 51, 3 category","Uniformorm 51, 4 category","Uniformorm
51, 5 category","Uniformorm 51, 7 category","Uniformorm 51, 10 category",
"Uniformorm 5, 2 category","Uniformorm 5, 3 category","Uniformorm 5, 4 category","Uniformorm 5, 5
category","Uniformorm 5, 7 category","Uniformorm 5, 10 category")

png("uhistslopescont%02d.png")  #saving samptriangularg distributions of slopes to an external file
for (i in 1:12){slopehist(shapelevel=slopeuboth[,i],title=slpnames[i])}
dev.off()

#make matrices of my empirical power on the probit model and save them to an external file
uprobit=cbind(probutwo,probuthree,probufour,probufive,probuseven,probuten,
                              probutwo2,probuthree2,probufour2,probufive2,probuseven2,probuten2)
dim(uprobit)=c(6,2)
rownames(uprobit)=c("2-Level","3-Level","4-Level","5-Level","7-Level","10-Level")
colnames(uprobit)=c("51 Level Power","5 Level Power")
write.csv(uprobit,file="uprobit.csv")


##############################################################################
##############################################################################
```

```
#Part 6:  Triangular

#set triangular probabilities for each section (matrix format with labels)
triangular=as.matrix(c((1/3),0,0,0,0,0,0,0,0,(1/6),(2/6),0,0,0,0,0,0,0,0,(1/10),(2/10),(3/10),0,0,0,0,0,0,0,(1/15),(2/15),(3/1
5),(4/15),0,0,0,0,0,0,(1/28),(2/28),(3/28),(4/28),(5/28),
(6/28),0,0,0,(1/55),(2/55),(3/55),(4/55),(5/55),(6/55),(7/55),(8/55),(9/55)))
dim(triangular)=c(9,6)
colnames(triangular)=c("Triangular2", "Triangular3", "Triangular4", "Triangular5", "Triangular7", "Triangular10")

#create empty matrices for ordered categorical “triangularear” variables
ltwo=matrix(0,300,4000)
lthree=matrix(0,300,4000)
lfour=matrix(0,300,4000)
lfive=matrix(0,300,4000)
lseven=matrix(0,300,4000)
lten=matrix(0,300,4000)

ltwo2=matrix(0,300,4000)
lthree2=matrix(0,300,4000)
lfour2=matrix(0,300,4000)
lfive2=matrix(0,300,4000)
lseven2=matrix(0,300,4000)
lten2=matrix(0,300,4000)


lprob2=vector("numeric",4000)
lprob3=lprob2
lprob4=lprob2
lprob5=lprob2
lprob7=lprob2
lprob10=lprob2

lprob22=vector("numeric",4000)
lprob32=lprob2
lprob42=lprob2
lprob52=lprob2
lprob72=lprob2
lprob102=lprob2




#with the 51 level x (triangular)

for (k in 1:6){
        if (k==1){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51l2[i,j]<=qnorm(triangular[1,k],0,sd(ylatent51l2[,j])+1)){ltwo[i,j]=1}
else{ltwo[i,j]=2}}
                        #running a binary probit using glm
                        glm.model=summary(glm(as.factor(ltwo[,j])~xlat[,1],family=binomial("probit")))
                        if (glm.model$coefficients[2,4]<.05) {lprob2[j]=1}} #counting the significant slopes
                        probltwo=sum(lprob2)/4000} #compting empirical power for the probit models
```

```
	if (k==2){
		for (j in 1:4000){
			for (i in 1:300){

	if(ylatent51l3[i,j]<=qnorm(triangular[1,k],0,sd(ylatent51l3[,j])+1)){lthree[i,j]=1} else{

	if(ylatent51l3[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent51l3[,j])+1)){lthree[i,j]=2}
else{lthree[i,j]=3}}}
			#running an ordered probit regression from the MASS package
			polr.model=summary(polr(as.factor(lthree[,j])~xlat[,1],method="probit",Hess=T))
			#squaring my T from the output and counting the significant slopes

	if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob3[j]=1}}
			problthree=sum(lprob3)/4000}  #calculating empirical power in the ordered probit
models


	if (k==3){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51l4[i,j]<=qnorm(triangular[1,k],0,sd(ylatent51l4[,j])+1)){lfour[i,j]=1}
else{

	if(ylatent51l4[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent51l4[,j])+1)){lfour[i,j]=2} else{

	if(ylatent51l4[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent51l4[,j])+1)){lfour[i,j
]=3} else{lfour[i,j]=4}}}}
			polr.model=summary(polr(as.factor(lfour[,j])~xlat[,1],method="probit",Hess=T))

	if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob4[j]=1}}
			problfour=sum(lprob4)/4000}


	if (k==4){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51l5[i,j]<=qnorm(triangular[1,k],0,sd(ylatent51l5[,j])+1)){lfive[i,j]=1}
else{

	if(ylatent51l5[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent51l5[,j])+1)){lfive[i,j]=2} else{

	if(ylatent51l5[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent51l5[,j])+1)){lfive[i,j
]=3} else{

	if(ylatent51l5[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k],0,sd(ylatent51l5[,
j])+1)){lfive[i,j]=4} else{lfive[i,j]=5}}}}}
			polr.model=summary(polr(as.factor(lfive[,j])~xlat[,1],method="probit",Hess=T))

	if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob5[j]=1}}
			problfive=sum(lprob5)/4000}


	if (k==5){
```

```
		for (j in 1:4000){
			for (i in 1:300){

	if(ylatent5117[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5117[,j])+1)){lseven[i,j]=1} else{

	if(ylatent5117[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent5117[,j])+1)){lseven[i,j]=2} else{

	if(ylatent5117[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent5117[,j])+1)){lseven[i
,j]=3} else{

	if(ylatent5117[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k],0,sd(ylatent5117[,
j])+1)){lseven[i,j]=4} else{

	if(ylatent5117[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k],0
,sd(ylatent5117[,j])+1)){lseven[i,j]=5}else{

	if(ylatent5117[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]+t
riangular[6,k],0,sd(ylatent5117[,j])+1)){lseven[i,j]=6}else{lseven[i,j]=7}}}}}}}
			polr.model=summary(polr(as.factor(lseven[,j])~xlat[,1],method="probit",Hess=T))

	if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob7[j]=1}}
			problseven=sum(lprob7)/4000}


	if (k==6){
		for (j in 1:4000){
			for (i in 1:300){

	if(ylatent51110[i,j]<=qnorm(triangular[1,k],0,sd(ylatent51110[,j])+1)){lten[i,j]=1} else{

	if(ylatent51110[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent51110[,j])+1)){lten[i,j]=2} else{

	if(ylatent51110[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent51110[,j])+1)){lten[i
,j]=3} else{

	if(ylatent51110[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k],0,sd(ylatent5111
0[,j])+1)){lten[i,j]=4} else{

	if(ylatent51110[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k],
0,sd(ylatent51110[,j])+1)){lten[i,j]=5}else{

	if(ylatent51110[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k],0,sd(ylatent51110[,j])+1)){lten[i,j]=6}else{

	if(ylatent51110[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k]+triangular[7,k],0,sd(ylatent51110[,j])+1)){lten[i,j]=7}else{

	if(ylatent51110[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k]+triangular[7,k]+triangular[8,k],0,sd(ylatent51110[,j])+1)){lten[i,j]=8}else{

	if(ylatent51110[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k]+triangular[7,k]+triangular[8,k]+triangular[9,k],0,sd(ylatent51110[,j])+1)){lten[i,j]=9}else{lten[i,j]=
10}}}}}}}}}}
			polr.model=summary(polr(as.factor(lten[,j])~xlat[,1],method="probit",Hess=T))
```

```
        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob10[j]=1}}
                        problten=sum(lprob10)/4000}

}

write.csv(ltwo, file="ltwo.csv")
write.csv(lthree, file="lthree.csv")
write.csv(lfour, file="lfour.csv")
write.csv(lfive, file="lfive.csv")
write.csv(lseven, file="lseven.csv")
write.csv(lten, file="lten.csv")



#with the 5 level x (triangular)

for (k in 1:6){
        if (k==1){
                for (j in 1:4000){
                        for (i in 1:300){

        if(ylatent5l22[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5l22[,j])+1)){ltwo2[i,j]=1} else{ltwo2[i,j]=2}}
                        glm.model=summary(glm(as.factor(ltwo2[,j])~xlat2[,1],family=binomial("probit")))
                        if (glm.model$coefficients[2,4]<.05) {lprob22[j]=1}}
                        probltwo2=sum(lprob22)/4000}


        if (k==2){
                for (j in 1:4000){
                        for (i in 1:300){

        if(ylatent5l23[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5l23[,j])+1)){lthree2[i,j]=1} else{

        if(ylatent5l23[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent5l23[,j])+1)){lthree2[i,j]=2}
else{lthree2[i,j]=3}}}
                        polr.model=summary(polr(as.factor(lthree2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob32[j]=1}}
                        probltwo2=sum(lprob32)/4000}



        if (k==3){
                for (j in 1:4000){
                        for (i in 1:300){

        if(ylatent5l24[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5l24[,j])+1)){lfour2[i,j]=1} else{

        if(ylatent5l24[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent5l24[,j])+1)){lfour2[i,j]=2} else{

        if(ylatent5l24[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent5l24[,j])+1)){lfour2[i
,j]=3} else{lfour2[i,j]=4}}}}
                        polr.model=summary(polr(as.factor(lfour2[,j])~xlat2[,1],method="probit",Hess=T))
```

```
        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob42[j]=1}}
                        problfour2=sum(lprob42)/4000}



        if (k==4){
                for (j in 1:4000){
                        for (i in 1:300){

        if(ylatent5l25[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5l25[,j])+1)){lfive2[i,j]=1} else{

        if(ylatent5l25[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent5l25[,j])+1)){lfive2[i,j]=2} else{

        if(ylatent5l25[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent5l25[,j])+1)){lfive2[i,
j]=3} else{

        if(ylatent5l25[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k],0,sd(ylatent5l25[,
j])+1)){lfive2[i,j]=4} else{lfive2[i,j]=5}}}}}
                        polr.model=summary(polr(as.factor(lfive2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob52[j]=1}}
                        problfive2=sum(lprob52)/4000}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){

        if(ylatent5l27[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5l27[,j])+1)){lseven2[i,j]=1} else{

        if(ylatent5l27[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent5l27[,j])+1)){lseven2[i,j]=2} else{

        if(ylatent5l27[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent5l27[,j])+1)){lseven2
[i,j]=3} else{

        if(ylatent5l27[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k],0,sd(ylatent5l27[,
j])+1)){lseven2[i,j]=4} else{

        if(ylatent5l27[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k],0
,sd(ylatent5l27[,j])+1)){lseven2[i,j]=5}else{

        if(ylatent5l27[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]+t
riangular[6,k],0,sd(ylatent5l27[,j])+1)){lseven2[i,j]=6}else{lseven2[i,j]=7}}}}}}}
                        polr.model=summary(polr(as.factor(lseven2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob72[j]=1}}
                        problseven2=sum(lprob72)/4000}

        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){

        if(ylatent5l210[i,j]<=qnorm(triangular[1,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=1} else{
```

```
        if(ylatent5l210[i,j]<=qnorm(triangular[1,k]+triangular[2,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=2} else{

        if(ylatent5l210[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k],0,sd(ylatent5l210[,j])+1)){lten2
[i,j]=3} else{

        if(ylatent5l210[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k],0,sd(ylatent5l21
0[,j])+1)){lten2[i,j]=4} else{

        if(ylatent5l210[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k],
0,sd(ylatent5l210[,j])+1)){lten2[i,j]=5}else{

        if(ylatent5l210[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=6}else{

        if(ylatent5l210[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k]+triangular[7,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=7}else{

        if(ylatent5l210[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k]+triangular[7,k]+triangular[8,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=8}else{

        if(ylatent5l210[i,j]<=qnorm(triangular[1,k]+triangular[2,k]+triangular[3,k]+triangular[4,k]+triangular[5,k]
+triangular[6,k]+triangular[7,k]+triangular[8,k]+triangular[9,k],0,sd(ylatent5l210[,j])+1)){lten2[i,j]=9}else{lten2[i,j
]=10}}}}}}}}}}
                        polr.model=summary(polr(as.factor(lten2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){lprob102[j]=1}}
                        problten2=sum(lprob102)/4000}
}

write.csv(ltwo2, file="ltwo2.csv")
write.csv(lthree2, file="lthree2.csv")
write.csv(lfour2, file="lfour2.csv")
write.csv(lfive2, file="lfive2.csv")
write.csv(lseven2, file="lseven2.csv")
write.csv(lten2, file="lten2.csv")


#Get percent significant for each level

#vectors of zeros to hold 0,1 valles, 1 if model had significant slope
#lm2 -- triangularear model 2 level dependent variable cont is the 100 level x, cat is the 5 level
lm2cont=vector("numeric",4000)
lm3cont=vector("numeric",4000)
lm4cont=vector("numeric",4000)
lm5cont=vector("numeric",4000)
lm7cont=vector("numeric",4000)
lm10cont=vector("numeric",4000)

lm2cat=lm2cont
lm3cat=lm2cont
lm4cat=lm2cont
lm5cat=lm2cont
lm7cat=lm2cont
lm10cat=lm2cont
```

```
#create empty vectors to store estimates of slope
lslopeone=vector("numeric",4000)
lslopeone2=vector("numeric",4000)
lslopetwo=vector("numeric",4000)
lslopetwo2=vector("numeric",4000)
lslopethree=vector("numeric",4000)
lslopethree2=vector("numeric",4000)
lslopefour=vector("numeric",4000)
lslopefour2=vector("numeric",4000)
lslopefive=vector("numeric",4000)
lslopefive2=vector("numeric",4000)
lslopeseven=vector("numeric",4000)
lslopeseven2=vector("numeric",4000)
lslopeten=vector("numeric",4000)
lslopeten2=vector("numeric",4000)


for (i in 1:4000) {

#running the triangular models
m2cont=lm(ltwo[,i]~xlat[,1])
m2cat=lm(ltwo2[,i]~xlat2[,1])
m3cont=lm(lthree[,i]~xlat[,1])
m3cat=lm(lthree2[,i]~xlat2[,1])
m4cont=lm(lfour[,i]~xlat[,1])
m4cat=lm(lfour2[,i]~xlat2[,1])
m5cont=lm(lfive[,i]~xlat[,1])
m5cat=lm(lfive2[,i]~xlat2[,1])
m7cont=lm(lseven[,i]~xlat[,1])
m7cat=lm(lseven2[,i]~xlat2[,1])
m10cont=lm(lten[,i]~xlat[,1])
m10cat=lm(lten2[,i]~xlat2[,1])


#grabbing the slopes
lslopetwo[i]=m2cont$coefficients[2]
lslopetwo2[i]=m2cat$coefficients[2]
lslopethree[i]=m3cont$coefficients[2]
lslopethree2[i]=m3cat$coefficients[2]
lslopefour[i]=m4cont$coefficients[2]
lslopefour2[i]=m4cat$coefficients[2]
lslopefive[i]=m5cont$coefficients[2]
lslopefive2[i]=m5cat$coefficients[2]
lslopeseven[i]=m7cont$coefficients[2]
lslopeseven2[i]=m7cat$coefficients[2]
lslopeten[i]=m10cont$coefficients[2]
lslopeten2[i]=m10cat$coefficients[2]


#counting how many of them are significant
if (summary(m2cont)$fstatistic[1] > qf(.95,summary(m2cont)$fstatistic[2],summary(m2cont)$fstatistic[3]))
{lm2cont[i]=1}
```

```
if (summary(m2cat)$fstatistic[1] > qf(.95,summary(m2cat)$fstatistic[2],summary(m2cat)$fstatistic[3]))
{lm2cat[i]=1}

if (summary(m3cont)$fstatistic[1] > qf(.95,summary(m3cont)$fstatistic[2],summary(m3cont)$fstatistic[3]))
{lm3cont[i]=1}
if (summary(m3cat)$fstatistic[1] > qf(.95,summary(m3cat)$fstatistic[2],summary(m3cat)$fstatistic[3]))
{lm3cat[i]=1}

if (summary(m4cont)$fstatistic[1] > qf(.95,summary(m4cont)$fstatistic[2],summary(m4cont)$fstatistic[3]))
{lm4cont[i]=1}
if (summary(m4cat)$fstatistic[1] > qf(.95,summary(m4cat)$fstatistic[2],summary(m4cat)$fstatistic[3]))
{lm4cat[i]=1}

if (summary(m5cont)$fstatistic[1] > qf(.95,summary(m5cont)$fstatistic[2],summary(m5cont)$fstatistic[3]))
{lm5cont[i]=1}
if (summary(m5cat)$fstatistic[1] > qf(.95,summary(m5cat)$fstatistic[2],summary(m5cat)$fstatistic[3]))
{lm5cat[i]=1}

if (summary(m7cont)$fstatistic[1] > qf(.95,summary(m7cont)$fstatistic[2],summary(m7cont)$fstatistic[3]))
{lm7cont[i]=1}
if (summary(m7cat)$fstatistic[1] > qf(.95,summary(m7cat)$fstatistic[2],summary(m7cat)$fstatistic[3]))
{lm7cat[i]=1}

if (summary(m10cont)$fstatistic[1] > qf(.95,summary(m10cont)$fstatistic[2],summary(m10cont)$fstatistic[3]))
{lm10cont[i]=1}
if (summary(m10cat)$fstatistic[1] > qf(.95,summary(m10cat)$fstatistic[2],summary(m10cat)$fstatistic[3]))
{lm10cat[i]=1}
}

#getting the percentages
lper2cont=sum(lm2cont)/40
lper2cat=sum(lm2cat)/40

lper3cont=sum(lm3cont)/40
lper3cat=sum(lm3cat)/40

lper4cont=sum(lm4cont)/40
lper4cat=sum(lm4cat)/40

lper5cont=sum(lm5cont)/40
lper5cat=sum(lm5cat)/40

lper7cont=sum(lm7cont)/40
lper7cat=sum(lm7cat)/40

lper10cont=sum(lm10cont)/40
lper10cat=sum(lm10cat)/40

write.csv(c(lper2cont,lper3cont,lper4cont,lper5cont,lper7cont,lper10cont,lper2cat,lper3cat,lper4cat,lper5cat,
lper7cat,lper10cat),file="lper.csv")

#generating sampling distributions of the slope
slopeboth=cbind(lslopetwo,lslopethree,lslopefour,lslopefive,lslopeseven,lslopeten,lslopetwo2,lslopethree2,lslopefou
r2,lslopefive2,lslopeseven2,lslopeten2)
```

```
slpnames=c("Triangularear 51, 2 category","Triangularear 51, 3 category","Triangular 51, 4
category","Triangularear 51, 5 category","Triangularear 51, 7 category","Triangular 51, 10 category",
"Triangularear 5, 2 category","Triangularear 5, 3 category","Triangularear 5, 4 category","Triangularear 5, 5
category","Triangularear 5, 7 category","Triangularear 5, 10 category")

png("lhistslopescont%02d.png")  #saving samptriangularg distributions of slopes to an external file
for (i in 1:12){slopehist(shapelevel=slopeboth[,i],title=slpnames[i])}
dev.off()

#make matrices of my empirical power on the probit model and save them to an external file
lprobit=c(probltwo,problthree,problfour,problfive,problseven,problten,
          probltwo2,problthree2,problfour2,problfive2,problseven2,problten2)
dim(lprobit)=c(6,2)
rownames(lprobit)=c("2-Level","3-Level","4-Level","5-Level","7-Level","10-Level")
colnames(lprobit)=c("51 Level Power","5 Level Power")
write.csv(lprobit,file="lprobit.csv")




################################################################################
################################################################################




#Part 7:  Exponential

#set exponential  probabilities for each section (matrix format with labels)
exp=as.matrix(c((1/3),0,0,0,0,0,0,0,0,(1/7),(2/7),0,0,0,0,0,0,0,(1/15),(2/15),(4/15),0,0,0,0,0,0,(1/31),(2/31),(4/31),(8/
31),0,0,0,0,0,
(1/127),(2/127),(4/127),(8/127),(16/127),(32/127),0,0,0,(1/1023),(2/1023),(4/1023),(8/1023),(16/1023),(32/1023),(6
4/1023),(128/1023),(256/1023)))
dim(exp)=c(9,6)
colnames(exp)=c("Exp2", "Exp3", "Exp4", "Exp5", "Exp7", "Exp10")

#create empty matrices for ordered categorical “exp” variables
etwo=matrix(0,300,4000)
ethree=matrix(0,300,4000)
efour=matrix(0,300,4000)
efive=matrix(0,300,4000)
eseven=matrix(0,300,4000)
eten=matrix(0,300,4000)

etwo2=matrix(0,300,4000)
ethree2=matrix(0,300,4000)
efour2=matrix(0,300,4000)
efive2=matrix(0,300,4000)
eseven2=matrix(0,300,4000)
eten2=matrix(0,300,4000)

#recall, these are the vectors to hold the counts of significant slopes on the probit models
eprob2=vector("numeric",4000)
eprob3=eprob2
```

```
eprob4=eprob2
eprob5=eprob2
eprob7=eprob2
eprob10=eprob2

eprob22=vector("numeric",4000)
eprob32=eprob2
eprob42=eprob2
eprob52=eprob2
eprob72=eprob2
eprob102=eprob2


#with the 51 level x (exponential)

for (k in 1:6){
	if (k==1){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51e2[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e2[,j])+1)){etwo[i,j]=1}
else{etwo[i,j]=2}}
			#running a binary probit using glm
			glm.model=summary(glm(as.factor(etwo[,j])~xlat[,1],family=binomial("probit")))
			if (glm.model$coefficients[2,4]<.05) {eprob2[j]=1}} #counting the significant slopes
			probetwo=sum(eprob2)/4000} #compting empirical power for the probit models


	if (k==2){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51e3[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e3[,j])+1)){ethree[i,j]=1}
else{

	if(ylatent51e3[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e3[,j])+1)){ethree[i,j]=2}
else{ethree[i,j]=3}}}
			#running an ordered probit regression from the MASS package
			polr.model=summary(polr(as.factor(ethree[,j])~xlat[,1],method="probit",Hess=T))
			#squaring my T from the output and counting the significant slopes

	if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob3[j]=1}}
			probethree=sum(eprob3)/4000}  #calculating empirical power in the ordered probit
models


	if (k==3){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51e4[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e4[,j])+1)){efour[i,j]=1}
else{

	if(ylatent51e4[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e4[,j])+1)){efour[i,j]=2} else{

	if(ylatent51e4[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent51e4[,j])+1)){efour[i,j]=3}
else{efour[i,j]=4}}}}
			polr.model=summary(polr(as.factor(efour[,j])~xlat[,1],method="probit",Hess=T))
```

```
        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob4[j]=1}}
                        probefour=sum(eprob4)/4000}


        if (k==4){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51e5[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e5[,j])+1)){efive[i,j]=1}
else{

        if(ylatent51e5[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e5[,j])+1)){efive[i,j]=2} else{

        if(ylatent51e5[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent51e5[,j])+1)){efive[i,j]=3} else{

        if(ylatent51e5[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent51e5[,j])+1)){efive[i,j]=4}
else{efive[i,j]=5}}}}}
                        polr.model=summary(polr(as.factor(efive[,j])~xlat[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob5[j]=1}}
                        probefive=sum(eprob5)/4000}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51e7[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e7[,j])+1)){eseven[i,j]=1}
else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e7[,j])+1)){eseven[i,j]=2} else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent51e7[,j])+1)){eseven[i,j]=3} else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent51e7[,j])+1)){eseven[i,j]=4}
else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k],0,sd(ylatent51e7[,j])+1)){eseve
n[i,j]=5}else{

        if(ylatent51e7[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k],0,sd(ylatent51e7[,j])+
1)){eseven[i,j]=6}else{eseven[i,j]=7}}}}}}}
                        polr.model=summary(polr(as.factor(eseven[,j])~xlat[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob7[j]=1}}
                        probeseven=sum(eprob7)/4000}


        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51e10[i,j]<=qnorm(exp[1,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=1}
else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=2} else{
```

```
        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=3} else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=4}
else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k],0,sd(ylatent51e10[,j])+1)){ete
n[i,j]=5}else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k],0,sd(ylatent51e10[,j
])+1)){eten[i,j]=6}else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k],0,sd(ylate
nt51e10[,j])+1)){eten[i,j]=7}else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k]+exp[8,k],
0,sd(ylatent51e10[,j])+1)){eten[i,j]=8}else{

        if(ylatent51e10[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k]+exp[8,k]+
exp[9,k],0,sd(ylatent51e10[,j])+1)){eten[i,j]=9}else{eten[i,j]=10}}}}}}}}}}
                        polr.model=summary(polr(as.factor(eten[,j])~xlat[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob10[j]=1}}
                        probeten=sum(eprob10)/4000}

}

write.csv(etwo, file="etwo.csv")
write.csv(ethree, file="ethree.csv")
write.csv(efour, file="efour.csv")
write.csv(efive, file="efive.csv")
write.csv(eseven, file="eseven.csv")
write.csv(eten, file="eten.csv")


#with the 5 level x (exponential)

for (k in 1:6){
        if (k==1){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5e22[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e22[,j])+1)){etwo2[i,j]=1}
else{etwo2[i,j]=2}}
                        glm.model=summary(glm(as.factor(etwo2[,j])~xlat2[,1],family=binomial("probit")))
                        if (glm.model$coefficients[2,4]<.05) {eprob22[j]=1}}
                        probetwo2=sum(eprob22)/4000}


        if (k==2){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5e23[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e23[,j])+1)){ethree2[i,j]=1}
else{
```

```
        if(ylatent5e23[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e23[,j])+1)){ethree2[i,j]=2}
else{ethree2[i,j]=3}}}
                        polr.model=summary(polr(as.factor(ethree2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob32[j]=1}}
                        probethree2=sum(eprob32)/4000}



        if (k==3){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5e24[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e24[,j])+1)){efour2[i,j]=1}
else{

        if(ylatent5e24[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e24[,j])+1)){efour2[i,j]=2} else{

        if(ylatent5e24[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent5e24[,j])+1)){efour2[i,j]=3}
else{efour2[i,j]=4}}}}
                        polr.model=summary(polr(as.factor(efour2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob42[j]=1}}
                        probefour2=sum(eprob42)/4000}



        if (k==4){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5e25[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e25[,j])+1)){efive2[i,j]=1}
else{

        if(ylatent5e25[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e25[,j])+1)){efive2[i,j]=2} else{

        if(ylatent5e25[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent5e25[,j])+1)){efive2[i,j]=3} else{

        if(ylatent5e25[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent5e25[,j])+1)){efive2[i,j]=4}
else{efive2[i,j]=5}}}}}
                        polr.model=summary(polr(as.factor(efive2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob52[j]=1}}
                        probefive2=sum(eprob52)/4000}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5e27[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e27[,j])+1)){eseven2[i,j]=1}
else{

        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e27[,j])+1)){eseven2[i,j]=2} else{

        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent5e27[,j])+1)){eseven2[i,j]=3} else{
```

```
        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent5e27[,j])+1)){eseven2[i,j]=4}
else{

        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k],0,sd(ylatent5e27[,j])+1)){eseve
n2[i,j]=5}else{

        if(ylatent5e27[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k],0,sd(ylatent5e27[,j])+
1)){eseven2[i,j]=6}else{eseven2[i,j]=7}}}}}}}
                        polr.model=summary(polr(as.factor(eseven2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob72[j]=1}}
                        probeseven2=sum(eprob72)/4000}

        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5e210[i,j]<=qnorm(exp[1,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=1}
else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=2} else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=3} else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=4}
else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k],0,sd(ylatent5e210[,j])+1)){ete
n2[i,j]=5}else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k],0,sd(ylatent5e210[,j
])+1)){eten2[i,j]=6}else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k],0,sd(ylate
nt5e210[,j])+1)){eten2[i,j]=7}else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k]+exp[8,k],
0,sd(ylatent5e210[,j])+1)){eten2[i,j]=8}else{

        if(ylatent5e210[i,j]<=qnorm(exp[1,k]+exp[2,k]+exp[3,k]+exp[4,k]+exp[5,k]+exp[6,k]+exp[7,k]+exp[8,k]+
exp[9,k],0,sd(ylatent5e210[,j])+1)){eten2[i,j]=9}else{eten2[i,j]=10}}}}}}}}}}
                        polr.model=summary(polr(as.factor(eten2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){eprob102[j]=1}}
                        probeten2=sum(eprob102)/4000}
}

write.csv(etwo2, file="etwo2.csv")
write.csv(ethree2, file="ethree2.csv")
write.csv(efour2, file="efour2.csv")
write.csv(efive2, file="efive2.csv")
write.csv(eseven2, file="eseven2.csv")
write.csv(eten2, file="eten2.csv")
```

```
#Get percent significant for each level

#vectors of zeros to hold 0,1 values, 1 if model had significant slope
#em2 -- uniform model 2 level dependent variable cont is the 100 level x, cat is the 5 level
em2cont=vector("numeric",4000)
em3cont=vector("numeric",4000)
em4cont=vector("numeric",4000)
em5cont=vector("numeric",4000)
em7cont=vector("numeric",4000)
em10cont=vector("numeric",4000)

em2cat=em2cont
em3cat=em2cont
em4cat=em2cont
em5cat=em2cont
em7cat=em2cont
em10cat=em2cont

#vectors of zero to hold estimates of slope
eslopeone=vector("numeric",4000)
eslopeone2=vector("numeric",4000)
eslopetwo=vector("numeric",4000)
eslopetwo2=vector("numeric",4000)
eslopethree=vector("numeric",4000)
eslopethree2=vector("numeric",4000)
eslopefour=vector("numeric",4000)
eslopefour2=vector("numeric",4000)
eslopefive=vector("numeric",4000)
eslopefive2=vector("numeric",4000)
eslopeseven=vector("numeric",4000)
eslopeseven2=vector("numeric",4000)
eslopeten=vector("numeric",4000)
eslopeten2=vector("numeric",4000)

for (i in 1:4000) {

#running the linear models
#none of the cat ones are relevant yet and will reqeire a variable change to be relevant
m2cont=lm(etwo[,i]~xlat[,1])
m2cat=lm(etwo2[,i]~xlat2[,1])
m3cont=lm(ethree[,i]~xlat[,1])
m3cat=lm(ethree2[,i]~xlat2[,1])
m4cont=lm(efour[,i]~xlat[,1])
m4cat=lm(efour2[,i]~xlat2[,1])
m5cont=lm(efive[,i]~xlat[,1])
m5cat=lm(efive2[,i]~xlat2[,1])
m7cont=lm(eseven[,i]~xlat[,1])
m7cat=lm(eseven2[,i]~xlat2[,1])
m10cont=lm(eten[,i]~xlat[,1])
m10cat=lm(eten2[,i]~xlat2[,1])

#grabbing the slopes
eslopetwo[i]=m2cont$coefficients[2]
eslopetwo2[i]=m2cat$coefficients[2]
eslopethree[i]=m3cont$coefficients[2]
```

```
eslopethree2[i]=m3cat$coefficients[2]
eslopefour[i]=m4cont$coefficients[2]
eslopefour2[i]=m4cat$coefficients[2]
eslopefive[i]=m5cont$coefficients[2]
eslopefive2[i]=m5cat$coefficients[2]
eslopeseven[i]=m7cont$coefficients[2]
eslopeseven2[i]=m7cat$coefficients[2]
eslopeten[i]=m10cont$coefficients[2]
eslopeten2[i]=m10cat$coefficients[2]


#counting how many of them are significant
if (summary(m2cont)$fstatistic[1] > qf(.95,summary(m2cont)$fstatistic[2],summary(m2cont)$fstatistic[3]))
{em2cont[i]=1}
if (summary(m2cat)$fstatistic[1] > qf(.95,summary(m2cat)$fstatistic[2],summary(m2cat)$fstatistic[3]))
{em2cat[i]=1}

if (summary(m3cont)$fstatistic[1] > qf(.95,summary(m3cont)$fstatistic[2],summary(m3cont)$fstatistic[3]))
{em3cont[i]=1}
if (summary(m3cat)$fstatistic[1] > qf(.95,summary(m3cat)$fstatistic[2],summary(m3cat)$fstatistic[3]))
{em3cat[i]=1}

if (summary(m4cont)$fstatistic[1] > qf(.95,summary(m4cont)$fstatistic[2],summary(m4cont)$fstatistic[3]))
{em4cont[i]=1}
if (summary(m4cat)$fstatistic[1] > qf(.95,summary(m4cat)$fstatistic[2],summary(m4cat)$fstatistic[3]))
{em4cat[i]=1}

if (summary(m5cont)$fstatistic[1] > qf(.95,summary(m5cont)$fstatistic[2],summary(m5cont)$fstatistic[3]))
{em5cont[i]=1}
if (summary(m5cat)$fstatistic[1] > qf(.95,summary(m5cat)$fstatistic[2],summary(m5cat)$fstatistic[3]))
{em5cat[i]=1}

if (summary(m7cont)$fstatistic[1] > qf(.95,summary(m7cont)$fstatistic[2],summary(m7cont)$fstatistic[3]))
{em7cont[i]=1}
if (summary(m7cat)$fstatistic[1] > qf(.95,summary(m7cat)$fstatistic[2],summary(m7cat)$fstatistic[3]))
{em7cat[i]=1}

if (summary(m10cont)$fstatistic[1] > qf(.95,summary(m10cont)$fstatistic[2],summary(m10cont)$fstatistic[3]))
{em10cont[i]=1}
if (summary(m10cat)$fstatistic[1] > qf(.95,summary(m10cat)$fstatistic[2],summary(m10cat)$fstatistic[3]))
{em10cat[i]=1}
}

#getting the percentages
eper2cont=sum(em2cont)/40
eper2cat=sum(em2cat)/40

eper3cont=sum(em3cont)/40
eper3cat=sum(em3cat)/40

eper4cont=sum(em4cont)/40
eper4cat=sum(em4cat)/40

eper5cont=sum(em5cont)/40
eper5cat=sum(em5cat)/40
```

```
eper7cont=sum(em7cont)/40
eper7cat=sum(em7cat)/40

eper10cont=sum(em10cont)/40
eper10cat=sum(em10cat)/40

write.csv(c(eper2cont,eper3cont,eper4cont,eper5cont,eper7cont,eper10cont,
eper2cat,eper3cat,eper4cat,eper5cat,eper7cat,eper10cat),file="eper.csv")

#looking at the samptriangularg distributions of slope
slopeboth=cbind(eslopetwo,eslopethree,eslopefour,eslopefive,eslopeseven,eslopeten,eslopetwo2,eslopethree2,eslope
four2,eslopefive2,eslopeseven2,eslopeten2)
slpnames=c("Exponential 51, 2 category","Exponential 51, 3 category","Exponential 51, 4 category","Exponential
51, 5 category","Exponential 51, 7 category","Exponential 51, 10 category",
"Exponential 5, 2 category","Exponential 5, 3 category","Exponential 5, 4 category","Exponential 5, 5
category","Exponential 5, 7 category","Exponential 5, 10 category")

png("ehistslopescont%02d.png")  #saving samptriangularg distributions of slopes to an external file
for (i in 1:12){slopehist(shapelevel=slopeboth[,i],title=slpnames[i])}
dev.off()


#make matrices of my empirical power on the probit model and save them to an external file
eprobit=c(probetwo,probethree,probefour,probefive,probeseven,probeten,
                                probetwo2,probethree2,probefour2,probefive2,probeseven2,probeten2)
dim(eprobit)=c(6,2)
rownames(eprobit)=c("2-Level","3-Level","4-Level","5-Level","7-Level","10-Level")
colnames(eprobit)=c("51 Level Power","5 Level Power")
write.csv(eprobit,file="eprobit.csv")

#output an example histogram
png("exponential7.png")
hist(eseven[,10],main="Histogram of a 7-Level Exponential Ordered categorical Response",xlab="Category")
dev.off()




################################################################################
################################################################################



#Part 8:  Bell

#set bp probabilities for each section (matrix format with labels)
bp=as.matrix(c(.5,0,0,0,0,0,0,0,0,
          (1/4),(2/4),0,0,0,0,0,0,0,
          (1/6),(2/6),(2/6),0,0,0,0,0,0,
         (1/9),(2/9),(3/9),(2/9),0,0,0,0,0,
         (1/16),(2/16),(3/16),(4/16),(3/16),(2/16),0,0,0,
                  (1/30),(2/30),(3/30),(4/30),(5/30),(5/30),(4/30),(3/30),(2/30)))
dim(bp)=c(9,6)
```

```
colnames(bp)=c("Bell2", "Bell3", "Bell4", "Bell5", "Bell7", "Bell10")

#create empty matrices for ordered categorical bell shaped variables
btwo=matrix(0,300,4000)
bthree=matrix(0,300,4000)
bfour=matrix(0,300,4000)
bfive=matrix(0,300,4000)
bseven=matrix(0,300,4000)
bten=matrix(0,300,4000)

btwo2=matrix(0,300,4000)
bthree2=matrix(0,300,4000)
bfour2=matrix(0,300,4000)
bfive2=matrix(0,300,4000)
bseven2=matrix(0,300,4000)
bten2=matrix(0,300,4000)

bprob2=vector("numeric",4000)
bprob3=bprob2
bprob4=bprob2
bprob5=bprob2
bprob7=bprob2
bprob10=bprob2

bprob22=vector("numeric",4000)
bprob32=bprob22
bprob42=bprob22
bprob52=bprob22
bprob72=bprob22
bprob102=bprob22


#with the 51 level x (bell)

for (k in 1:6){
        if (k==1){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51b2[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b2[,j])+1)){btwo[i,j]=1}
else{btwo[i,j]=2}}
                        #running a binary probit using glm
                        glm.model=summary(glm(as.factor(btwo[,j])~xlat[,1],family=binomial("probit")))
                        if (glm.model$coefficients[2,4]<.05) {bprob2[j]=1}} #counting the significant slopes
                        probbtwo=sum(bprob2)/4000} #compting empirical power for the probit models


        if (k==2){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51b3[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b3[,j])+1)){bthree[i,j]=1}
else{

        if(ylatent51b3[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b3[,j])+1)){bthree[i,j]=2} else{bthree[i,j]=3}}}
                        #running an ordered probit regression from the MASS package
                        polr.model=summary(polr(as.factor(bthree[,j])~xlat[,1],method="probit",Hess=T))
```

```
                        #squaring my T from the output and counting the significant slopes

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob3[j]=1}}
                        probbthree=sum(bprob3)/4000}  #calculating empirical power in the ordered probit
models


        if (k==3){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51b4[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b4[,j])+1)){bfour[i,j]=1} else{

        if(ylatent51b4[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b4[,j])+1)){bfour[i,j]=2} else{

        if(ylatent51b4[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent51b4[,j])+1)){bfour[i,j]=3}
else{bfour[i,j]=4}}}}
                        polr.model=summary(polr(as.factor(bfour[,j])~xlat[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob4[j]=1}}
                        probbfour=sum(bprob4)/4000}


        if (k==4){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51b5[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b5[,j])+1)){bfive[i,j]=1} else{

        if(ylatent51b5[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b5[,j])+1)){bfive[i,j]=2} else{

        if(ylatent51b5[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent51b5[,j])+1)){bfive[i,j]=3} else{

        if(ylatent51b5[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent51b5[,j])+1)){bfive[i,j]=4}
else{bfive[i,j]=5}}}}}
                        polr.model=summary(polr(as.factor(bfive[,j])~xlat[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob5[j]=1}}
                        probbfive=sum(bprob5)/4000}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent51b7[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]=1}
else{

        if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]=2} else{

        if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]=3} else{

        if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]=4}
else{

        if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k],0,sd(ylatent51b7[,j])+1)){bseven[i,j]
=5}else{
```

```
	if(ylatent51b7[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k],0,sd(ylatent51b7[,j])+1)){bs
even[i,j]=6}else{bseven[i,j]=7}}}}}}}
				polr.model=summary(polr(as.factor(bseven[,j])~xlat[,1],method="probit",Hess=T))

	if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob7[j]=1}}
				probbseven=sum(bprob7)/4000}


	if (k==6){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent51b10[i,j]<=qnorm(bp[1,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]=1}
else{

	if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]=2} else{

	if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]=3} else{

	if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]=4}
else{

	if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k],0,sd(ylatent51b10[,j])+1)){bten[i,j]
=5}else{

	if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k],0,sd(ylatent51b10[,j])+1))
{bten[i,j]=6}else{

	if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k],0,sd(ylatent51b10
[,j])+1)){bten[i,j]=7}else{

	if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k]+bp[8,k],0,sd(ylate
nt51b10[,j])+1)){bten[i,j]=8}else{

	if(ylatent51b10[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k]+bp[8,k]+bp[9,k],
0,sd(ylatent51b10[,j])+1)){bten[i,j]=9}else{bten[i,j]=10}}}}}}}}}}
				polr.model=summary(polr(as.factor(bten[,j])~xlat[,1],method="probit",Hess=T))

	if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob10[j]=1}}
				probbten=sum(bprob10)/4000}

}

write.csv(btwo, file="btwo.csv")
write.csv(bthree, file="bthree.csv")
write.csv(bfour, file="bfour.csv")
write.csv(bfive, file="bfive.csv")
write.csv(bseven, file="bseven.csv")
write.csv(bten, file="bten.csv")



#with the 5 level x (bell)

for (k in 1:6){
```

```
	if (k==1){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5b22[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b22[,j])+1)){btwo2[i,j]=1}
else{btwo2[i,j]=2}}
			glm.model=summary(glm(as.factor(btwo2[,j])~xlat2[,1],family=binomial("probit")))
			if (glm.model$coefficients[2,4]<.05) {bprob22[j]=1}}
			probbtwo2=sum(bprob22)/4000}


	if (k==2){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5b23[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b23[,j])+1)){bthree2[i,j]=1}
else{

	if(ylatent5b23[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b23[,j])+1)){bthree2[i,j]=2}
else{bthree2[i,j]=3}}}
			polr.model=summary(polr(as.factor(bthree2[,j])~xlat2[,1],method="probit",Hess=T))

	if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob32[j]=1}}
			probbthree2=sum(bprob32)/4000}



	if (k==3){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5b24[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b24[,j])+1)){bfour2[i,j]=1}
else{

	if(ylatent5b24[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b24[,j])+1)){bfour2[i,j]=2} else{

	if(ylatent5b24[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent5b24[,j])+1)){bfour2[i,j]=3}
else{bfour2[i,j]=4}}}}
			polr.model=summary(polr(as.factor(bfour2[,j])~xlat2[,1],method="probit",Hess=T))

	if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob42[j]=1}}
			probbfour2=sum(bprob42)/4000}



	if (k==4){
		for (j in 1:4000){
			for (i in 1:300){
				if(ylatent5b25[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b25[,j])+1)){bfive2[i,j]=1}
else{

	if(ylatent5b25[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b25[,j])+1)){bfive2[i,j]=2} else{

	if(ylatent5b25[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent5b25[,j])+1)){bfive2[i,j]=3} else{

	if(ylatent5b25[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent5b25[,j])+1)){bfive2[i,j]=4}
else{bfive2[i,j]=5}}}}}
			polr.model=summary(polr(as.factor(bfive2[,j])~xlat2[,1],method="probit",Hess=T))
```

```
        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob52[j]=1}}
                        probbfive2=sum(bprob52)/4000}


        if (k==5){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5b27[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j]=1}
else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j]=2} else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j]=3} else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j]=4}
else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k],0,sd(ylatent5b27[,j])+1)){bseven2[i,j
]=5}else{

        if(ylatent5b27[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k],0,sd(ylatent5b27[,j])+1)){bs
even2[i,j]=6}else{bseven2[i,j]=7}}}}}}}
                        polr.model=summary(polr(as.factor(bseven2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob72[j]=1}}
                        probbseven2=sum(bprob72)/4000}

        if (k==6){
                for (j in 1:4000){
                        for (i in 1:300){
                                if(ylatent5b210[i,j]<=qnorm(bp[1,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j]=1}
else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j]=2} else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j]=3} else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j]=4}
else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k],0,sd(ylatent5b210[,j])+1)){bten2[i,j
]=5}else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k],0,sd(ylatent5b210[,j])+1))
{bten2[i,j]=6}else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k],0,sd(ylatent5b210
[,j])+1)){bten2[i,j]=7}else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k]+bp[8,k],0,sd(ylate
nt5b210[,j])+1)){bten2[i,j]=8}else{

        if(ylatent5b210[i,j]<=qnorm(bp[1,k]+bp[2,k]+bp[3,k]+bp[4,k]+bp[5,k]+bp[6,k]+bp[7,k]+bp[8,k]+bp[9,k],
0,sd(ylatent5b210[,j])+1)){bten2[i,j]=9}else{bten2[i,j]=10}}}}}}}}}}
```

```
                    polr.model=summary(polr(as.factor(bten2[,j])~xlat2[,1],method="probit",Hess=T))

        if(pf((polr.model$coefficients[1,3]^2),1,polr.model$df.residual,lower.tail=FALSE)<.05){bprob102[j]=1}}
                    probbten2=sum(bprob102)/4000}
}

write.csv(btwo2, file="btwo2.csv")
write.csv(bthree2, file="bthree2.csv")
write.csv(bfour2, file="bfour2.csv")
write.csv(bfive2, file="bfive2.csv")
write.csv(bseven2, file="bseven2.csv")
write.csv(bten2, file="bten2.csv")


#Get percent significant for each level

#vectors of zeros to hold 0,1 values, 1 if model had significant slope
#um2 -- uniformorm model 2 level dependent variable cont is the 100 level x, cat is the 5 level
bm2cont=vector("numeric",4000)
bm3cont=vector("numeric",4000)
bm4cont=vector("numeric",4000)
bm5cont=vector("numeric",4000)
bm7cont=vector("numeric",4000)
bm10cont=vector("numeric",4000)

bm2cat=bm2cont
bm3cat=bm2cont
bm4cat=bm2cont
bm5cat=bm2cont
bm7cat=bm2cont
bm10cat=bm2cont

#vectors of zero to hold estimates of slope
bslopeone=vector("numeric",4000)
bslopeone2=vector("numeric",4000)
bslopetwo=vector("numeric",4000)
bslopetwo2=vector("numeric",4000)
bslopethree=vector("numeric",4000)
bslopethree2=vector("numeric",4000)
bslopefour=vector("numeric",4000)
bslopefour2=vector("numeric",4000)
bslopefive=vector("numeric",4000)
bslopefive2=vector("numeric",4000)
bslopeseven=vector("numeric",4000)
bslopeseven2=vector("numeric",4000)
bslopeten=vector("numeric",4000)
bslopeten2=vector("numeric",4000)


for (i in 1:4000) {

#running the triangular models
m2cont=lm(btwo[,i]~xlat[,1])
m2cat=lm(btwo2[,i]~xlat2[,1])
m3cont=lm(bthree[,i]~xlat[,1])
```

```
m3cat=lm(bthree2[,i]~xlat2[,1])
m4cont=lm(bfour[,i]~xlat[,1])
m4cat=lm(bfour2[,i]~xlat2[,1])
m5cont=lm(bfive[,i]~xlat[,1])
m5cat=lm(bfive2[,i]~xlat2[,1])
m7cont=lm(bseven[,i]~xlat[,1])
m7cat=lm(bseven2[,i]~xlat2[,1])
m10cont=lm(bten[,i]~xlat[,1])
m10cat=lm(bten2[,i]~xlat2[,1])

#grabbing the slopes
bslopetwo[i]=m2cont$coefficients[2]
bslopetwo2[i]=m2cat$coefficients[2]
bslopethree[i]=m3cont$coefficients[2]
bslopethree2[i]=m3cat$coefficients[2]
bslopefour[i]=m4cont$coefficients[2]
bslopefour2[i]=m4cat$coefficients[2]
bslopefive[i]=m5cont$coefficients[2]
bslopefive2[i]=m5cat$coefficients[2]
bslopeseven[i]=m7cont$coefficients[2]
bslopeseven2[i]=m7cat$coefficients[2]
bslopeten[i]=m10cont$coefficients[2]
bslopeten2[i]=m10cat$coefficients[2]


#counting how many of them are significant
if (summary(m2cont)$fstatistic[1] > qf(.95,summary(m2cont)$fstatistic[2],summary(m2cont)$fstatistic[3]))
{bm2cont[i]=1}
if (summary(m2cat)$fstatistic[1] > qf(.95,summary(m2cat)$fstatistic[2],summary(m2cat)$fstatistic[3]))
{bm2cat[i]=1}

if (summary(m3cont)$fstatistic[1] > qf(.95,summary(m3cont)$fstatistic[2],summary(m3cont)$fstatistic[3]))
{bm3cont[i]=1}
if (summary(m3cat)$fstatistic[1] > qf(.95,summary(m3cat)$fstatistic[2],summary(m3cat)$fstatistic[3]))
{bm3cat[i]=1}

if (summary(m4cont)$fstatistic[1] > qf(.95,summary(m4cont)$fstatistic[2],summary(m4cont)$fstatistic[3]))
{bm4cont[i]=1}
if (summary(m4cat)$fstatistic[1] > qf(.95,summary(m4cat)$fstatistic[2],summary(m4cat)$fstatistic[3]))
{bm4cat[i]=1}

if (summary(m5cont)$fstatistic[1] > qf(.95,summary(m5cont)$fstatistic[2],summary(m5cont)$fstatistic[3]))
{bm5cont[i]=1}
if (summary(m5cat)$fstatistic[1] > qf(.95,summary(m5cat)$fstatistic[2],summary(m5cat)$fstatistic[3]))
{bm5cat[i]=1}

if (summary(m7cont)$fstatistic[1] > qf(.95,summary(m7cont)$fstatistic[2],summary(m7cont)$fstatistic[3]))
{bm7cont[i]=1}
if (summary(m7cat)$fstatistic[1] > qf(.95,summary(m7cat)$fstatistic[2],summary(m7cat)$fstatistic[3]))
{bm7cat[i]=1}

if (summary(m10cont)$fstatistic[1] > qf(.95,summary(m10cont)$fstatistic[2],summary(m10cont)$fstatistic[3]))
{bm10cont[i]=1}
if (summary(m10cat)$fstatistic[1] > qf(.95,summary(m10cat)$fstatistic[2],summary(m10cat)$fstatistic[3]))
{bm10cat[i]=1}
```

```
}

#getting the percentages
bper2cont=sum(bm2cont)/40
bper2cat=sum(bm2cat)/40

bper3cont=sum(bm3cont)/40
bper3cat=sum(bm3cat)/40

bper4cont=sum(bm4cont)/40
bper4cat=sum(bm4cat)/40

bper5cont=sum(bm5cont)/40
bper5cat=sum(bm5cat)/40

bper7cont=sum(bm7cont)/40
bper7cat=sum(bm7cat)/40

bper10cont=sum(bm10cont)/40
bper10cat=sum(bm10cat)/40

write.csv(c(bper2cont,bper3cont,bper4cont,bper5cont,bper7cont,bper10cont,bper2cat,bper3cat,
bper4cat,bper5cat,bper7cat,bper10cat),file="bper.csv")

#looking at the samptriangularg distributions of slope
slopeboth=cbind(bslopetwo,bslopethree,bslopefour,bslopefive,bslopeseven,bslopeten,bslopetwo2,bslopethree2,bslop
efour2,bslopefive2,bslopeseven2,bslopeten2)
slpnames=c("Bell 51, 2 category","Bell 51, 3 category","Bell 51, 4 category","Bell 51, 5 category","Bell 51, 7
category","Bell 51, 10 category",
"Bell 5, 2 category","Bell 5, 3 category","Bell 5, 4 category","Bell 5, 5 category","Bell 5, 7 category","Bell 5, 10
category")

png("bhistslopescont%02d.png")  #saving samptriangularg distributions of slopes to an external file
for (i in 1:12){slopehist(shapelevel=slopeboth[,i],title=slpnames[i])}
dev.off()


#make matrices of my empirical power on the probit model and save them to an external file
bprobit=c(probbtwo,probbthree,probbfour,probbfive,probbseven,probbten,
                        probbtwo2,probbthree2,probbfour2,probbfive2,probbseven2,probbten2)
dim(bprobit)=c(6,2)
rownames(bprobit)=c("2-Level","3-Level","4-Level","5-Level","7-Level","10-Level")
colnames(bprobit)=c("51 Level Power","5 Level Power")
write.csv(bprobit,file="bprobit.csv")


#output an example histogram
png("bell10.png")
hist(bten[,10],main="Histogram of a 10-Level Bell Ordered categorical Response",xlab="Category")
dev.off()



################################################################################
################################################################################
```

```
#Part 10 Report power pictorally

levels=c(2,3,4,5,7,10)  #creating a variable for x axis of plot

u50=c(uper2cont,uper3cont,uper4cont,uper5cont,uper7cont,uper10cont)/100 #51 level then 5 level
u5=c(uper2cat,uper3cat,uper4cat,uper5cat,uper7cat,uper10cat)/100 #making a vector of my power from uniformorm
scenario
l50=c(lper2cont,lper3cont,lper4cont,lper5cont,lper7cont,lper10cont)/100
l5=c(lper2cat,lper3cat,lper4cat,lper5cat,lper7cat,lper10cat)/100 #making a vector of my power from triangularear
scenario
e50=c(eper2cont,eper3cont,eper4cont,eper5cont,eper7cont,eper10cont)/100
e5=c(eper2cat,eper3cat,eper4cat,eper5cat,eper7cat,eper10cat)/100 #making a vector of my power from exponential
scenario
b50=c(bper2cont,bper3cont,bper4cont,bper5cont,bper7cont,bper10cont)/100
b5=c(bper2cat,bper3cat,bper4cat,bper5cat,bper7cat,bper10cat)/100 #making a vector of my power from bell
scenario

#now I am going to vectorize the actual power from the probit models
uactual50=uprobit[,1]
uactual5=uprobit[,2]
lactual50=lprobit[,1]
lactual5=lprobit[,2]
eactual50=eprobit[,1]
eactual5=eprobit[,2]
bactual50=bprobit[,1]
bactual5=bprobit[,2]


#for a given shape and xlat combo, plot the power

byshape50=cbind(u50,l50,e50,b50)  #making a matrix of ys for matplot
byshape5=cbind(u5,l5,e5,b5)
power=cbind(byshape50,byshape5)
row.names(power)=c("2-level","3-level","4-level","5-level","7-level","10-level")
write.csv(power,file="power_april.csv")

png("byshapes%02d.png")  #saving byshapes to an external file

#generating plots for poster
#plot for the 51 level scenario
matplot(x=levels, y=byshape50, type="l",ylim=c(.57,.83), ylab=mytype, xlab="Number of Categories",lty=1:4,
col=1:4,
lwd=2, main="", font=1, font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abtriangulare(h=.80,lty=5,col=5)
legend("bottomright", legend=c("Reference","Uniformorm","Belled","Triangularular","Exponential"),
lty=c(5,1,4,2,3), col=c(5,1,4,2,3))

#plot for the 5 level scenario
matplot(x=levels, y=byshape5, type="l",ylim=c(.57,.83), ylab=mytype, xlab="Number of Categories",lty=1:4,
col=1:4,
lwd=2, main="", font=1, font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
```

```
abtriangulare(h=.80,lty=5,col=5)
legend("bottomright", legend=c("Reference","Uniformorm","Belled","Triangularular","Exponential"),
lty=c(5,1,4,2,3), col=c(5,1,4,2,3))

dev.off()


#looking at each shapes power by level

png("bylevelpershape%02d.png")  #saving bylevels to an external file

matplot(x=levels, y=cbind(u50,u5,uactual50,uactual5), type="l", ylim=c(.57,.83), ylab=mytype, xlab="Number of
Categories",
lty=c(3,4,6,7),col=c(3,4,6,7),lwd=2,main="",font=1,font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abtriangulare(h=.80,lty=5,col=5)
legend("bottomright", legend=c("Reference",expression(paste("OLSLR",x^(51))),expression(paste("OLSLR",
x^(5))),
expression(paste("Probit ",x^(51))),expression(paste("Probit ",x^(5))),lty=c(5,3,4,6,7),col=c(5,3,4,6,7))

matplot(x=levels, y=cbind(l50,l5,lactual50,lactual5), type="l", ylim=c(.57,.83), ylab=mytype, xlab="Number of
Categories",
lty=c(3,4,6,7),col=c(3,4,6,7),lwd=2,main="",font=1,font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abtriangulare(h=.80,lty=5,col=5)
legend("bottomright", legend=c("Reference",expression(paste("OLSLR",x^(51))),expression(paste("OLSLR",
x^(5))),
expression(paste("Probit ",x^(51))),expression(paste("Probit ",x^(5))),lty=c(5,3,4,6,7),col=c(5,3,4,6,7))

matplot(x=levels, y=cbind(e50,e5,eactual50,eactual5), type="l", ylim=c(.57,.83), ylab=paste("Percent ",mytype),
xlab="Number of Categories",
lty=c(3,4,6,7),col=c(3,4,6,7),lwd=2,main="",font=1,font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abtriangulare(h=.80,lty=5,col=5)
legend("bottomright", legend=c("Reference",expression(paste("OLSLR",x^(51))),expression(paste("OLSLR",
x^(5))),
expression(paste("Probit ",x^(51))),expression(paste("Probit ",x^(5))),lty=c(5,3,4,6,7),col=c(5,3,4,6,7))

matplot(x=levels, y=cbind(b50,b5,bactual50,bactual5), type="l", ylim=c(.57,.83), ylab=paste("Percent ",mytype),
xlab="Number of Categories",
lty=c(3,4,6,7),col=c(3,4,6,7),lwd=2,main="",font=1,font.lab=1,xaxt="n")
axis(side=1, at=c(2,3,4,5,7,10), labels=c(2,3,4,5,7,10))
abtriangulare(h=.80,lty=5,col=5)
legend("bottomright", legend=c("Reference",expression(paste("OLSLR",x^(51))),expression(paste("OLSLR",
x^(5))),
expression(paste("Probit ",x^(51))),expression(paste("Probit ",x^(5))),lty=c(5,3,4,6,7),col=c(5,3,4,6,7))

dev.off()

#making a matrix of my powers on the OLSLR of the ordered categorical variables and writing to an external file
powerstorage=cbind(byshape50,byshape5)  #making a matrix to store as a csv
rownames(powerstorage)=c("2-Level","3-Level","4-Level","5-Level","7-Level","10-Level")
write.csv(powerstorage,file="powerfororordered categoricalwOLSLR.csv")
```

```
#making an external file of my probabilities used in creation of the ordered categorical pdfs
probs=cbind(triangular,exp,up,bp)
write.csv(probs,file="probs_used_for_cutting_ordered categorical.csv")


#writing some example histograms to an external file
png("pmf%1d.png")
hist(ufour[,1],breaks=c(0,1,2,3,4),main="Example Uniformorm Probability Mass Function", xlab="Ordered
categorical Categories",axes=FALSE,ylab="Probability")
hist(bfive[,1],breaks=c(0,1,2,3,4,5),main="Example Bell Probability Mass Function", xlab="Ordered categorical
Categories",axes=FALSE,ylab="Probability")
hist(lthree[,1],breaks=c(0,1,2,3),main="Example Triangularear Probability Mass Function", xlab="Ordered
categorical Categories",axes=FALSE,ylab="Probability")
hist(eseven[,1],breaks=c(0,1,2,3,4,5,6,7),main="Example Uniformorm Probability Mass Function", xlab="Ordered
categorical Categories",axes=FALSE,ylab="Probability")

hist(useven[,101],breaks=c(0,1,2,3,4,5,6,7),main="", xlab="Category Ranks",xaxt="n",ylab="Frequency")
axis(side=1,at=c(.5,1.5,2.5,3.5,4.5,5.5,6.5),labels=c(1,2,3,4,5,6,7))
hist(bseven[,101],breaks=c(0,1,2,3,4,5,6,7),main="", xlab="Category Ranks",xaxt="n",ylab="Frequency")
axis(side=1,at=c(.5,1.5,2.5,3.5,4.5,5.5,6.5),labels=c(1,2,3,4,5,6,7))
hist(lseven[,101],breaks=c(0,1,2,3,4,5,6,7),main="", xlab="Category Ranks",xaxt="n",ylab="Frequency")
axis(side=1,at=c(.5,1.5,2.5,3.5,4.5,5.5,6.5),labels=c(1,2,3,4,5,6,7))
hist(eseven[,101],breaks=c(0,1,2,3,4,5,6,7),main="", xlab="Category Ranks",xaxt="n",ylab="Frequency")
axis(side=1,at=c(.5,1.5,2.5,3.5,4.5,5.5,6.5),labels=c(1,2,3,4,5,6,7))
dev.off()


#Part 11: Interval Computation


########################################
#looking at power by distribution shape #
#exponential the only one diff from prob#
########################################

n=38

#Uniformorm between probit and OLSLR
llu10cont=(uprobit[6,1] - uper10cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((uprobit[6,1]*(1-
uprobit[6,1]))/4000 + (uper10cont/100)*(1-uper10cont/100)/4000)
ulu10cont=(uprobit[6,1] - uper10cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((uprobit[6,1]*(1-
uprobit[6,1]))/4000 + (uper10cont/100)*(1-uper10cont/100)/4000)
#between uniform and probit 51 there is no diff
llu10cat=(uprobit[6,2] - uper10cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((uprobit[6,2]*(1-
uprobit[6,2]))/4000 + (uper10cat/100)*(1-uper10cat/100)/4000)
ulu10cat=(uprobit[6,2] - uper10cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((uprobit[6,2]*(1-
uprobit[6,2]))/4000 + (uper10cat/100)*(1-uper10cat/100)/4000)
#between uniform and probit 5 there is no diff

#Bell between probit and OLSLR
llb10cont=(bprobit[6,1] - bper10cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((bprobit[6,1]*(1-
bprobit[6,1]))/4000 + (bper10cont/100)*(1-bper10cont/100)/4000)
ulb10cont=(bprobit[6,1] - bper10cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((bprobit[6,1]*(1-
bprobit[6,1]))/4000 + (bper10cont/100)*(1-bper10cont/100)/4000)
```

```
llb10cat=(bprobit[6,1] - bper10cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((bprobit[6,2]*(1-
bprobit[6,2]))/4000 + (bper10cat/100)*(1-bper10cat/100)/4000)
ulb10cat=(bprobit[6,1] - bper10cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((bprobit[6,2]*(1-
bprobit[6,2]))/4000 + (bper10cat/100)*(1-bper10cat/100)/4000)

#Triangularear between probit and OLSLR
lll10cont=(lprobit[6,1] - lper10cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((lprobit[6,1]*(1-
lprobit[6,1]))/4000 + (lper10cont/100)*(1-lper10cont/100)/4000)
ull10cont=(lprobit[6,1] - lper10cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((lprobit[6,1]*(1-
lprobit[6,1]))/4000 + (lper10cont/100)*(1-lper10cont/100)/4000)
lll10cat=(lprobit[6,1] - lper10cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((lprobit[6,2]*(1-lprobit[6,2]))/4000 +
(lper10cat/100)*(1-lper10cat/100)/4000)
ull10cat=(lprobit[6,1] - lper10cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((lprobit[6,2]*(1-lprobit[6,2]))/4000
+ (lper10cat/100)*(1-lper10cat/100)/4000)

#Exponential between probit and OLSLR
lle10cont=(eprobit[6,1] - eper10cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[6,1]*(1-
eprobit[6,1]))/4000 + (eper10cont/100)*(1-eper10cont/100)/4000)
ule10cont=(eprobit[6,1] - eper10cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[6,1]*(1-
eprobit[6,1]))/4000 + (eper10cont/100)*(1-eper10cont/100)/4000)
lle10cat=(eprobit[6,1] - eper10cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[6,2]*(1-eprobit[6,2]))/4000
+ (eper10cat/100)*(1-eper10cat/100)/4000)
ule10cat=(eprobit[6,1] - eper10cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[6,2]*(1-
eprobit[6,2]))/4000 + (eper10cat/100)*(1-eper10cat/100)/4000)

lle7cont=(eprobit[5,1] - eper7cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[5,1]*(1-
eprobit[5,1]))/4000 + (eper7cont/100)*(1-eper7cont/100)/4000)
ule7cont=(eprobit[5,1] - eper7cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[5,1]*(1-
eprobit[5,1]))/4000 + (eper7cont/100)*(1-eper7cont/100)/4000)
lle7cat=(eprobit[5,1] - eper7cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[5,2]*(1-eprobit[5,2]))/4000 +
(eper7cat/100)*(1-eper7cat/100)/4000)
ule7cat=(eprobit[5,1] - eper7cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[5,2]*(1-eprobit[5,2]))/4000
+ (eper7cat/100)*(1-eper7cat/100)/4000)

lle5cont=(eprobit[4,1] - eper5cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[4,1]*(1-
eprobit[4,1]))/4000 + (eper5cont/100)*(1-eper5cont/100)/4000)
ule5cont=(eprobit[4,1] - eper5cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[4,1]*(1-
eprobit[4,1]))/4000 + (eper5cont/100)*(1-eper5cont/100)/4000)
lle5cat=(eprobit[4,1] - eper5cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[4,2]*(1-eprobit[4,2]))/4000 +
(eper5cat/100)*(1-eper5cat/100)/4000)
ule5cat=(eprobit[4,1] - eper5cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[4,2]*(1-eprobit[4,2]))/4000
+ (eper5cat/100)*(1-eper5cat/100)/4000)

lle4cont=(eprobit[3,1] - eper4cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[3,1]*(1-
eprobit[3,1]))/4000 + (eper4cont/100)*(1-eper4cont/100)/4000)
ule4cont=(eprobit[3,1] - eper4cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[3,1]*(1-
eprobit[3,1]))/4000 + (eper4cont/100)*(1-eper4cont/100)/4000)
lle4cat=(eprobit[3,1] - eper4cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[3,2]*(1-eprobit[3,2]))/4000 +
(eper4cat/100)*(1-eper4cat/100)/4000)
ule4cat=(eprobit[3,1] - eper4cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt((eprobit[3,2]*(1-eprobit[3,2]))/4000
+ (eper4cat/100)*(1-eper4cat/100)/4000)


########################################
#looking at power per x generation level#
```

```
#between bell, uniform, and triangular, the only  #
#place there is evidence for a diff is  #
#is between bell and uniform at 10 levels  #
#in the x5 scenario                  #
########################################

#51 10 Levels

#Bell 51 vs Triangular 51 at 10 levels #no diff
llbl51=(bper10cont/100-lper10cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper10cont/100)*(1-
bper10cont/100))/4000 + ((lper10cont/100)*(1-lper10cont/100))/4000)
ulbl51=(bper10cont/100-lper10cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper10cont/100)*(1-
bper10cont/100))/4000 + ((lper10cont/100)*(1-lper10cont/100))/4000)

#Bell 51 vs Uniform 51 at 10 levels #no diff
llbu51=(bper10cont/100-uper10cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper10cont/100)*(1-
bper10cont/100))/4000 + ((uper10cont/100)*(1-uper10cont/100))/4000)
ulbu51=(bper10cont/100-uper10cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper10cont/100)*(1-
bper10cont/100))/4000 + ((uper10cont/100)*(1-uper10cont/100))/4000)

#Triangular 51 vs Uniform 51 at 10 levels #no diff
lllu51=(lper10cont/100-uper10cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper10cont/100)*(1-
lper10cont/100))/4000 + ((uper10cont/100)*(1-uper10cont/100))/4000)
ullu51=(lper10cont/100-uper10cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper10cont/100)*(1-
lper10cont/100))/4000 + ((uper10cont/100)*(1-uper10cont/100))/4000)


#5 10 Levels

#Bell 5 vs Triangular 5 at 10 levels #no diff
llbl5=(bper10cat/100-lper10cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper10cat/100)*(1-
bper10cat/100))/4000 + ((lper10cat/100)*(1-lper10cat/100))/4000)
ulbl5=(bper10cat/100-lper10cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper10cat/100)*(1-
bper10cat/100))/4000 + ((lper10cat/100)*(1-lper10cat/100))/4000)

#Bell 5 vs Uniform 5 at 10 levels #diff
llbu5=(bper10cat/100-uper10cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper10cat/100)*(1-
bper10cat/100))/4000 + ((uper10cat/100)*(1-uper10cat/100))/4000)
ulbu5=(bper10cat/100-uper10cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper10cat/100)*(1-
bper10cat/100))/4000 + ((uper10cat/100)*(1-uper10cat/100))/4000)

#Triangular 5 vs Uniform 5 at 10 levels
lllu5=(lper10cat/100-uper10cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper10cat/100)*(1-
lper10cat/100))/4000 + ((uper10cat/100)*(1-uper10cat/100))/4000)
ullu5=(lper10cat/100-uper10cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper10cat/100)*(1-
lper10cat/100))/4000 + ((uper10cat/100)*(1-uper10cat/100))/4000)


#51 7 Levels

#Bell 51 vs Triangular 51 at 7 levels #no diff
llbl517=(bper7cont/100-lper7cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper7cont/100)*(1-
bper7cont/100))/4000 + ((lper7cont/100)*(1-lper7cont/100))/4000)
ulbl517=(bper7cont/100-lper7cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper7cont/100)*(1-
bper7cont/100))/4000 + ((lper7cont/100)*(1-lper7cont/100))/4000)
```

```
#Bell 51 vs Uniform 51 at 7 levels #no diff
llbu517=(bper7cont/100-uper7cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper7cont/100)*(1-
bper7cont/100))/4000 + ((uper7cont/100)*(1-uper7cont/100))/4000)
ulbu517=(bper7cont/100-uper7cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper7cont/100)*(1-
bper7cont/100))/4000 + ((uper7cont/100)*(1-uper7cont/100))/4000)

#Triangular 51 vs Uniform 51 at 7 levels #no diff
lllu517=(lper7cont/100-uper7cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper7cont/100)*(1-
lper7cont/100))/4000 + ((uper7cont/100)*(1-uper7cont/100))/4000)
ullu517=(lper7cont/100-uper7cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper7cont/100)*(1-
lper7cont/100))/4000 + ((uper7cont/100)*(1-uper7cont/100))/4000)


#5 7 Levels

#Bell 5 vs Triangular 5 at 7 levels #no diff
llbl57=(bper7cat/100-lper7cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper7cat/100)*(1-
bper7cat/100))/4000 + ((lper7cat/100)*(1-lper7cat/100))/4000)
ulbl57=(bper7cat/100-lper7cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper7cat/100)*(1-
bper7cat/100))/4000 + ((lper7cat/100)*(1-lper7cat/100))/4000)

#Bell 5 vs Uniform 5 at 7 levels #diff
llbu57=(bper7cat/100-uper7cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper7cat/100)*(1-
bper7cat/100))/4000 + ((uper7cat/100)*(1-uper7cat/100))/4000)
ulbu57=(bper7cat/100-uper7cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper7cat/100)*(1-
bper7cat/100))/4000 + ((uper7cat/100)*(1-uper7cat/100))/4000)
#Bell x5 7 level diff from uniform x5 7 level

#Triangular 5 vs Uniform 5 at 7 levels #no diff
lllu57=(lper7cat/100-uper7cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper7cat/100)*(1-lper7cat/100))/4000
+ ((uper7cat/100)*(1-uper7cat/100))/4000)
ullu57=(lper7cat/100-uper7cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper7cat/100)*(1-
lper7cat/100))/4000 + ((uper7cat/100)*(1-uper7cat/100))/4000)


#51 5 Levels

#Bell 51 vs Triangular 51 at 5 levels #diff
llbl515=(bper5cont/100-lper5cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper5cont/100)*(1-
bper5cont/100))/4000 + ((lper5cont/100)*(1-lper5cont/100))/4000)
ulbl515=(bper5cont/100-lper5cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper5cont/100)*(1-
bper5cont/100))/4000 + ((lper5cont/100)*(1-lper5cont/100))/4000)

#Bell 51 vs Uniform 51 at 5 levels #diff
llbu515=(bper5cont/100-uper5cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper5cont/100)*(1-
bper5cont/100))/4000 + ((uper5cont/100)*(1-uper5cont/100))/4000)
ulbu515=(bper5cont/100-uper5cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper5cont/100)*(1-
bper5cont/100))/4000 + ((uper5cont/100)*(1-uper5cont/100))/4000)

#Triangular 51 vs Uniform 51 at 5 levels #diff
lllu515=(lper5cont/100-uper5cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper5cont/100)*(1-
lper5cont/100))/4000 + ((uper5cont/100)*(1-uper5cont/100))/4000)
ullu515=(lper5cont/100-uper5cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper5cont/100)*(1-
lper5cont/100))/4000 + ((uper5cont/100)*(1-uper5cont/100))/4000)
```

#5 at 5 levels

#Bell 5 vs Triangular 5 at 5 levels #diff
llbl55=(bper5cat/100-lper5cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper5cat/100)*(1-bper5cat/100))/4000 + ((lper5cat/100)*(1-lper5cat/100))/4000)
ulbl55=(bper5cat/100-lper5cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper5cat/100)*(1-bper5cat/100))/4000 + ((lper5cat/100)*(1-lper5cat/100))/4000)

#Bell 5 vs Uniform 5 at 5 levels #no diff
llbu55=(bper5cat/100-uper5cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper5cat/100)*(1-bper5cat/100))/4000 + ((uper5cat/100)*(1-uper5cat/100))/4000)
ulbu55=(bper5cat/100-uper5cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper5cat/100)*(1-bper5cat/100))/4000 + ((uper5cat/100)*(1-uper5cat/100))/4000)

#Triangular 5 vs Uniform 5 at 5 levels #no diff
lllu55=(lper5cat/100-uper5cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper5cat/100)*(1-lper5cat/100))/4000 + ((uper5cat/100)*(1-uper5cat/100))/4000)
ullu55=(lper5cat/100-uper5cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper5cat/100)*(1-lper5cat/100))/4000 + ((uper5cat/100)*(1-uper5cat/100))/4000)

#51

#Bell 51 vs Triangular 51 at 4 Levels #diff
llbl514=(bper4cont/100-lper4cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper4cont/100)*(1-bper4cont/100))/4000 + ((lper4cont/100)*(1-lper4cont/100))/4000)
ulbl514=(bper4cont/100-lper4cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper4cont/100)*(1-bper4cont/100))/4000 + ((lper4cont/100)*(1-lper4cont/100))/4000)

#Bell 51 vs Uniform 51 at 4 Levels #no diff
llbu514=(bper4cont/100-uper4cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper4cont/100)*(1-bper4cont/100))/4000 + ((uper4cont/100)*(1-uper4cont/100))/4000)
ulbu514=(bper4cont/100-uper4cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper4cont/100)*(1-bper4cont/100))/4000 + ((uper4cont/100)*(1-uper4cont/100))/4000)

#Triangular 51 vs Uniform 51 at 4 Levels #diff
lllu514=(lper4cont/100-uper4cont/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper4cont/100)*(1-lper4cont/100))/4000 + ((uper4cont/100)*(1-uper4cont/100))/4000)
ullu514=(lper4cont/100-uper4cont/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper4cont/100)*(1-lper4cont/100))/4000 + ((uper4cont/100)*(1-uper4cont/100))/4000)

#5

#Bell 5 vs Triangular 5 at 4 Levels #diff
llbl54=(bper4cat/100-lper4cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper4cat/100)*(1-bper4cat/100))/4000 + ((lper4cat/100)*(1-lper4cat/100))/4000)
ulbl54=(bper4cat/100-lper4cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper4cat/100)*(1-bper4cat/100))/4000 + ((lper4cat/100)*(1-lper4cat/100))/4000)

#Bell 5 vs Uniform 5 at 4 Levels #diff
llbu54=(bper4cat/100-uper4cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper4cat/100)*(1-bper4cat/100))/4000 + ((uper4cat/100)*(1-uper4cat/100))/4000)

```
ulbu54=(bper4cat/100-uper4cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((bper4cat/100)*(1-
bper4cat/100))/4000 + ((uper4cat/100)*(1-uper4cat/100))/4000)

#Triangular 5 vs Uniform 5 at 4 Levels #diff
lllu54=(lper4cat/100-uper4cat/100) - qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper4cat/100)*(1-lper4cat/100))/4000
+ ((uper4cat/100)*(1-uper4cat/100))/4000)
ullu54=(lper4cat/100-uper4cat/100) + qnorm(.025/n,lower.tail=FALSE)*sqrt(((lper4cat/100)*(1-
lper4cat/100))/4000 + ((uper4cat/100)*(1-uper4cat/100))/4000)
```