ON DETERMINING THE POWER OF A TEST
AFTER DATA COLLECTION

by

WILLIAM AVRAM CHERNOFF

B.S., Chapman University, 2007

A REPORT
Submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE
Department of Statistics
College of Art and Sciences
KANSAS STATE UNIVERSITY
Manhattan, Kansas
2009

Approved by

Major Professor
Dr. Leigh Murray
ABSTRACT

The term retrospective power describes methods for estimating the true power of a test after data have been collected. These methods have been recommended by some authors when null hypothesis of a test cannot be rejected. This report uses simulations to study power as a construct of an observed effect, variance, sample size, and set level of significance under the balanced one-way analysis of variance model for normally distributed populations with constant variance.

Retrospective power, as a construct of sample data, is not recommended when the null hypothesis of a test cannot be rejected. When the p-value of the test is large, estimates for true power tend to fall below the 0.80 level and width-minimized confidence limits for true power tend to be wide.
# TABLE OF CONTENTS

List of Figures ........................................................................................................................… v  
List of Tables ............................................................................................................................ vii  
Acknowledgements ................................................................................................................ viii  
1. Introduction ............................................................................................................................ 1  
2. Literature Review ................................................................................................................... 2  
3. Methods .................................................................................................................................. 4  
   3.1 Model ................................................................................................................................ 4  
   3.2 Estimated Power ............................................................................................................... 8  
   3.3 Confidence Interval Optimization ..................................................................................... 9  
   3.4 Parameter Settings ........................................................................................................... 14  
   3.5 Iteration ............................................................................................................................ 16  
4. Results .................................................................................................................................. 18  
   4.1 Two-Sample Symmetric Case ......................................................................................... 19  
   4.2 Three-Sample Symmetric Case ....................................................................................... 21  
   4.3 Three-Sample Asymmetric Case ..................................................................................... 23  
   4.4 Four Sample Symmetric Case A ...................................................................................... 25  
   4.5 Four Sample Symmetric Case B ...................................................................................... 27  
   4.6 Four Sample Asymmetric Case ....................................................................................... 29
LIST OF FIGURES

Figure 3.1.a: Update Bracket Interval for Minimand for v less than u ........................................ 12
Figure 3.1.b: Update Bracket Interval for Minimand for v less than u ........................................ 12
Figure 4.1.a: Two-Sample Symmetric Case for Power vs Sample Size with Sigma 2.5 ............ 20
Figure 4.1.b: Two-Sample Symmetric Case for Power vs Sample Size with Sigma 5.0 ............ 20
Figure 4.1.c: Two-Sample Symmetric Case for Power vs Sample Size with Sigma 10.0 .......... 20
Figure 4.1.d: Two-Sample Symmetric Case for Power vs Sample Size with Sigma 20.0 .......... 20
Figure 4.2.a: Three-Sample Symmetric Case for Power vs Sample Size with Sigma 2.5 ........ 22
Figure 4.2.b: Three-Sample Symmetric Case for Power vs Sample Size with Sigma 5.0 ........ 22
Figure 4.2.c: Three-Sample Symmetric Case for Power vs Sample Size with Sigma 10.0 ....... 22
Figure 4.2.d: Three-Sample Symmetric Case for Power vs Sample Size with Sigma 20.0 ...... 22
Figure 4.3.a: Three-Sample Asymmetric Case for Power vs Sample Size with Sigma 2.5 ...... 24
Figure 4.3.b: Three-Sample Asymmetric Case for Power vs Sample Size with Sigma 5.0 ...... 24
Figure 4.3.c: Three-Sample Asymmetric Case for Power vs Sample Size with Sigma 10.0 ..... 24
Figure 4.3.d: Three-Sample Asymmetric Case for Power vs Sample Size with Sigma 20.0 ..... 24
Figure 4.4.a: Four-Sample Symmetric Case A for Power vs Sample Size with Sigma 2.5 ...... 26
Figure 4.4.b: Four-Sample Symmetric Case A for Power vs Sample Size with Sigma 5.0 ...... 26
Figure 4.4.c: Four-Sample Symmetric Case A for Power vs Sample Size with Sigma 10.0 ..... 26
Figure 4.4.d: Four-Sample Symmetric Case A for Power vs Sample Size with Sigma 20.0 ..... 26
Figure 4.5.a: Four-Sample Symmetric Case B for Power vs Sample Size with Sigma 2.5 ...... 28
Figure 4.5.b: Four-Sample Symmetric Case B for Power vs Sample Size with Sigma 5.0 ...... 28
Figure 4.5.c: Four-Sample Symmetric Case B for Power vs Sample Size with Sigma 10.0 ..... 28
Figure 4.5.d: Four-Sample Symmetric Case B for Power vs Sample Size with Sigma 20.0 ..... 28
Figure 4.6.a: Four-Sample Asymmetric Case for Power vs Sample Size with Sigma 2.5 ......... 30
Figure 4.6.b: Four-Sample Asymmetric Case for Power vs Sample Size with Sigma 5.0 .......... 30
Figure 4.6.c: Four-Sample Asymmetric Case for Power vs Sample Size with Sigma 10.0 ...... 30
Figure 4.6.d: Four-Sample Asymmetric Case for Power vs Sample Size with Sigma 20.0 ...... 30
LIST OF TABLES

Table 3.1: Summary of Parameter Settings for Simulations ................................................ 15
ACKNOWLEDGEMENTS

I would like to thank Dr. Leigh Murray, Dr. James Neill, and Dr. Weixing Song for contributing in the development of this report. I thank Dr. Murray for her role as major professor; for her guidance through retrospective power; and for her topical and gritty digressions on statistical consulting. I thank Dr. Neill for sitting on my committee; for always having a moment for questions; and for his ability to impart calm and fortitude during the rapidity and rigor of graduate studies. I thank Dr. Song for acting as a member of my committee; for helping me with SAS; and for his staunch encouragement and winning sense of humor.

I would also like to thank the faculty at Chapman University for preparing me for graduate studies. I owe much of my personal and academic growth to Dr. Atanas Radenski, Dr. Peter Jispen, Dr. Drew Moshier, Dr. Adrian Vajiac, and Dr. Mihaela Vajiac. For all the people I met at Chapman University, you are the most appreciated.

I would like to thank the friends I have made through the K-State Statistics department. I thank Seth Demel for all-night study sessions, Lee Goerl for ‘talking stats’ at social functions, and Aleksandra Banasik for discussing teaching strategies over early morning coffee. You made home feel larger than southern California.

Finally, I would like to thank my mother Valerie Chernoff, my sister Elyse Felder, and the friends I hold dear for putting up with all this ‘Kansas/Graduate School’ business. Your thoughts and affections are a constant source of inspiration.
1 Introduction

For many researchers, failing to reject the null hypothesis of statistical test is disconcerting. In some cases, the research design is questioned: was the study powerful enough to detect a meaningful effect? While valid statistical methods exist for calculating power before data have been collected, determining power retrospectively is seen in the literature as controversial. Here we explore power, as described in Thomas (1997), as a construct of sample data using the balanced one-way ANOVA model. Section 2 discusses methods for calculating prospective power; methods for estimating retrospective power; and concerns associated with estimating retrospective power. Section 3 outlines the balanced one-way ANOVA model; a method for estimating power using the observed effect, variance, sample size, and set significance level; and a method for constructing a minimal \((1 - \alpha)100\%\) confidence interval for true power. In addition, Section 3 contains conditions for simulations and implementation methods used for simulations. Section 4 summarizes the simulation results, displaying the location of true power, median estimated power, average estimated power, average p-value of tests and average confidence limits produced from the simulations. Section 4 also discusses the effect of population conditions on power calculations. Flow charts, code, and complete output are given in appendices A-D. By examining power after data have been collected and statistical tests have been performed, the perspective shifts from calculating power to estimating power.
2 Literature Review

Calculating the power of a statistical test before data have been collected, denoted *prospective power*, is generally regarded by statisticians as a good practice for obtaining conclusive results (American Statistical Association 1999). Power calculations are useful in determining adequate sample sizes for an upcoming study (Lenth 2001). The power of a statistical test can be computed using a set significance level, sample size, effect, and variance (Lenth 2001; Gerard et al. 1998). Lenth (2001) provides a list of discussion-techniques for eliciting meaningful effect and variance values for a potential study. Power for t-tests, multiple regression, one-way ANOVA, and other common statistical methods can be computed using the SAS procedure POWER (SAS Institute Inc. 2004b). Power for linear models with fixed class effects and contrast statements can be calculated using the SAS procedure GLMPOWER (SAS Institute Inc. 2004a). For many researchers, prospective power calculations are within reach.

Calculating the power of a statistical test after data have been collected, i.e. *retrospective power*, is less accepted by many statisticians as compared to prospective power (Hoenig and Heisey 2001; Thomas 1997; Yuan and Maxwell 2005). Retrospective power has been proposed for interpreting results when the null hypothesis cannot be rejected (Taylor and Muller 1995; Hogarty and Kromrey 2001) and for determining the sample size needed for the observed data to show statistically significant results (Steidl et al. 1997; Thomas 1997). Some popular methods for calculating retrospective power included: an observed sample size and given significance level where effect and variance come from literature (Yuan and Maxwell 2005); an observed effect, observed sample size, and given significance level (Yuan and Maxwell 2005) where variance comes from literature; an observed variance, observed sample size, and given significance level (Thomas 1997; Yuan and Maxwell 2005; Steidl et al. 1997) where effect comes from literature; and an observed effect, an observed sample size, an
observed variance, and given significance level (Gerard et al. 1998; Thomas 1997; Hoenig and Heisey 2001). Of these four methods only the last method is reviewed here.

Calculating retrospective power using an observed effect, an observed variance, an observed sample size, and given significance level has been shown to produce biased estimates when using an observed noncentrality parameter (Gerard et al. 1998). Thomas (1997) suggests adjusting the observed noncentrality parameter as shown in Wright and O’Brien (1988) and Johnson et al. (1995) to remove the bias in calculating power. The adjusted observed noncentrality parameter can produce negative values, which are typically set to zero (Gerard et al. 1998). A median estimator for the noncentrality parameter has been proposed by Taylor and Muller (1996) as a means for correcting the retrospective power calculation. The median estimator for the noncentrality parameter under-estimates half the time and over-estimates half the time. Here, the adjusted observed noncentrality parameter is explored.

Concerns about using retrospective power procedures to represent true power exist in the literature. Zumbo and Hubley (1998) provide a Bayesian argument against retrospective power as a logical representative for true power. Hoening and Heisey (2001) note that retrospective power obtain through observed values is redundant with the p-value of the test.

Due to the bias associated with retrospective power, confidence intervals for effect have been proposed in lieu of retrospective power analyses when the null hypothesis cannot be rejected (Gerard et al. 1998; Thomas 1997; Steidl et al. 1997).

In this paper, we examine the method presented in Thomas (1997) for estimating the power of a test in the context of the balanced one-way ANOVA model.
3 Methods
3.1 Model

To investigate post-hoc power, the balanced one-way analysis of variance model is considered. The following briefly reviews this model and the usual analysis as described in Kuehl (2000) and Milliken and Johnson (2009). The means model for this situation is

\[ y_{ij} = \mu_i + \epsilon_{ij}, \text{ with } \epsilon_{ij} \text{’s i.i.d } N(0, \sigma^2) \text{ for } i = 1, \ldots, p, j = 1, \ldots, n. \]  

(1.1)

Here \( y_{ij} \) represents the observed value of the \( j \)th observation from the \( i \)th treatment group and \( \mu_i \) denotes the population mean of the \( i \)th treatment group. The term \( \epsilon_{ij} \) represents the random error associated with the \( j \)th observation from the \( i \)th treatment group. The errors are assumed independent and identically distributed from a normal population with mean 0 and variance \( \sigma^2 \).

The null and alternative hypotheses are that of no difference and that of some difference among the \( p \) population means, respectively:

\[ H_0 : \mu_1 = \ldots = \mu_p \]
\[ H_a : \text{at least two } \mu_i \text{’s not equal} \]  

(1.2)

When the sample means are not “too different” from each other we deem the data consistent with the null hypothesis of equal means. We do not conclude that the population means are all the same but acknowledge that sample means as too similar to be called different. When at least two sample means exist which “strongly” differ we favor the alternative hypothesis and conclude that not all of the population means are equal to one another.

Two estimates of variance are used to measure the amount of evidence against the null hypothesis. The first estimate of variance is based only on the assumptions of the model. The second estimate of variance is based on the assumptions of the model and the assumption that
the null hypothesis is true. Of these two estimates of variance, the second estimate has a larger expected mean square than the first estimate when the null hypothesis is false.

The mean square for error is the first estimate of $\sigma^2$ and is calculated by taking the average of the sample variances $s_1^2, s_2^2, ..., s_p^2$:

$$MSE = \frac{s_1^2 + s_2^2 + \ldots + s_p^2}{p} \quad (1.3)$$

$$= \frac{1}{p(n-1)} \sum_{i=1}^{p} \sum_{j=1}^{n} (y_{ij} - \bar{y}_i)^2$$

$$= \frac{1}{p(n-1)} \left\{ \sum_{i=1}^{p} \sum_{j=1}^{n} y_{ij}^2 - n \sum_{i=1}^{p} \bar{y}_i^2 \right\}$$

$$= \frac{1}{p(n-1)} \left\{ \sum_{i=1}^{p} \sum_{j=1}^{n} y_{ij}^2 - n \bar{Y}^T \bar{Y} \right\}$$

$$= \frac{SSE}{DFE},$$

where $\bar{y}_i = \frac{1}{n} \sum_{j=1}^{n} y_{ij}$, $\bar{Y} = [\bar{y}_1, \bar{y}_2, ..., \bar{y}_p]$, SSE is the error sum of squares, and DFE is the error degrees of freedom. Under the assumptions of the model,

$$\frac{(DFE)(MSE)}{\sigma^2} \sim \chi^2_{(DFE)} \quad (1.4)$$

and

$$E(MSE) = \sigma^2. \quad (1.5)$$

The mean square for treatments ($MST$) is the second estimate of $\sigma^2$ and is calculated as the sample variance of the sample means $\bar{y}_1, \bar{y}_2, ..., \bar{y}_p$. That is,
\[ MST = \frac{n}{p-1} \sum_{i=1}^{p} (\bar{y}_i - \bar{y})^2 \]  
\[ = \frac{n}{p-1} \left\{ \sum_{i=1}^{p} \bar{y}_{i}^2 - p\bar{y}^2 \right\} \]  
\[ = \frac{1}{p-1} \left\{ \sum_{i=1}^{p} \frac{y_{i}^2}{n} - \frac{y}{n}^2 \right\} \]  
\[ = \frac{1}{p-1} \left\{ n\bar{Y}^T \bar{Y} - \frac{n}{p} (\bar{Y}^T J)^2 \right\} \]  
\[ = \frac{SST}{DFT}, \]

where \( y_i = \sum_{j=1}^{n} y_{ij} \), \( \bar{y}_i = \frac{1}{n} \sum_{j=1}^{n} y_{ij} \), \( J^T = [1, 1, \ldots, 1] \), SST is the treatment sum of squares, and DFT is the treatment degrees of freedom. Under the assumptions of the model and the assumption that the null hypothesis is true

\[ \frac{(DFT)(MST)}{\sigma^2} \sim \chi^2_{(DFT)} \]  
\[ (1.7) \]

and

\[ E(MST) = \sigma^2. \]  
\[ (1.8) \]

Under the assumptions of the model and the assumption that the null hypothesis is not true

\[ \frac{(DFT)(MST)}{\sigma^2} \sim \chi^2_{(DFT, \lambda)} \]  
\[ (1.9) \]

and

\[ E(MST) = \sigma^2 + \frac{n}{p-1} \sum_{i=1}^{p} (\mu_i - \bar{\mu})^2, \]  
\[ (1.10) \]

where

\[ \lambda = \frac{n}{\sigma^2} \sum_{i=1}^{p} (\mu_i - \bar{\mu})^2 \]  
\[ (1.11) \]

denotes the noncentrality parameter.
Under the assumptions of the model and assumption that the null hypothesis is true,

\[
\frac{\left[ (DFT)(MST) / \sigma^2 \right] / DFT}{\left[ (DFE)(MSE) / \sigma^2 \right] / DFE} \sim F_{(DFT, DFE)},
\] (1.12)

where (1.7) and (1.9) are independently distributed $\chi^2$ random variables. Under the assumptions of the model and the assumption that the alternative hypothesis is true,

\[
\frac{\left[ (DFT)(MST) / \sigma^2 \right] / DFT}{\left[ (DFE)(MSE) / \sigma^2 \right] / DFE} \sim F_{(DFT, DFE, \lambda)},
\] (1.13)

The observed $F$ -statistic for the test is calculated as the ratio of the mean square for treatments over the mean square for error. This statistic is used to measure the amount of evidence in the sample data against the null hypothesis,

\[
F = \frac{MST}{MSE} = \frac{\left[ (DFT)(MST) / \sigma^2 \right] / DFT}{\left[ (DFE)(MSE) / \sigma^2 \right] / DFE}
\] (1.14)

Given the assumptions of the means model and the assumption that the null hypothesis is true, the mean square for error should be of similar size to the mean square for treatments, practically speaking. However, given the null hypothesis is false, the mean square for error should be smaller than the mean square for treatments. An uncommonly large $F$-statistic leads to the conclusion that not all population means are equal. The $p$-value of the test gives the probability of obtaining an $F$-statistic greater than or equal to the one observed. The more evidence in the sample data against the null hypothesis of equal means the smaller the $p$-value.
3.2 Estimated Power

Power is estimated using the observed effect and the observed sampling variance. Let the cumulative distribution function of the noncentral $F$ -distribution be denoted as

$$F(DFT, DFE, \lambda).$$

(2.1)

The power of the $F$ -test is defined as

$$\text{power} = 1 - F(F_{\text{crit}} | DFT, DFE, \lambda),$$

(2.2)

where $F_{\text{crit}}$ is the $100 \cdot (1 - \alpha)$ percentile from a central $F$ -distribution with numerator degrees of freedom $DFT$, denominator degrees of freedom $DFE$, level of significance $\alpha$, and noncentrality parameter $\lambda$ (Thomas 1997). A positively-biased estimate for the power of an $F$ -test is

$$\hat{\text{power}} = 1 - F(F_{\text{crit}} | DFT, DFE, \lambda)$$

(2.3)

where $\hat{\lambda} = \frac{SST}{MSE}$ (Thomas 1997). The estimated noncentrality parameter can be adjusted to remove the upward bias. Therefore, an unbiased estimate of power (Wright and O’Brien 1988, and Johnson et al. 1995, cited in Thomas 1997) is

$$\hat{\text{power}}_{\text{adj}} = 1 - F(F_{\text{crit}} | DFT, DFE, \hat{\lambda}_{\text{adj}})$$

(2.4)

where $\hat{\lambda}_{\text{adj}}$ is the adjusted estimated noncentrality parameter and is computed as

$$\hat{\lambda}_{\text{adj}} = [\hat{\lambda} \cdot (DFE - 2)/ DFE] - DFT.$$

(2.5)

Note that for $\hat{\lambda} \cdot (DFE - 2)/ DFE$ less than $DFT$, $\hat{\lambda}_{\text{adj}}$ can be negative.
Upper and lower confidence limits for the true power are therefore

\[
\text{power}_U = 1 - F\left(F_{\text{crit}} | DF_T, DFE, \lambda_U \right)
\]

(2.6)

and

\[
\text{power}_L = 1 - F\left(F_{\text{crit}} | DF_T, DFE, \lambda_L \right).
\]

(2.7)

where \(\lambda_U\) and \(\lambda_L\) satisfy \(F\left(F_{\text{obs}} | DF_T, DFE, \lambda_U \right) = \alpha_U\) and \(F\left(F_{\text{obs}} | DF_T, DFE, \lambda_L \right) = 1 - \alpha_L\) in which \(\alpha_U\) is the upper tail probability and \(\alpha_L\) lower tail probability (Thomas 1997). The upper and lower tail probabilities \(\alpha_U\) and \(\alpha_L\) are those which define the \(100 \cdot (1 - \alpha_U - \alpha_L)\) percent confidence interval for true power.

3.3 Confidence Interval Optimization

Confidence limits for true power were minimized to explore retrospective power under a “best” case scenario. Confidence limits set under \(\alpha_L = \alpha_U = \alpha / 2\) are not generally optimal (i.e. narrowest confidence interval) because of the skewed nature of the noncentral \(F\)-distribution. Setting upper and lower probabilities equal may under-represent, or over-represent, true power.

For a user-specified level of significance \(\alpha\), we consider the \(100 (1 - \alpha)\)% confidence interval for true power, where

\[
\alpha = \alpha_L + \alpha_U.
\]

(3.1)

We denote the inverse cumulative distribution function of the noncentral \(F\)-distribution by

\[
F^{-1}\left(\alpha, DF_T, DFE, \lambda_{\text{adj}} \right),
\]

(3.2)
where $\alpha_c$ is a lower tail probability, $DFT$ the numerator degrees of freedom, $DFE$ the denominator degrees of freedom, and $\hat{\lambda}_{adj}$ the noncentrality parameter. We denote the quantile associated with $\alpha_L$ as

$$F_L = F^{-1}(\alpha_L, DFT, DFE, \hat{\lambda}_{adj}),$$

and the quantile associated with $\alpha_U$ as

$$F_U = F^{-1}(1 - \alpha_U, DFT, DFE, \hat{\lambda}_{adj}).$$

The width of the $100(1 - \alpha)$% confidence interval for true power is minimized by finding the distance between the lower and upper quantiles such that

$$W = F_U - F_L$$

is minimized. Note, for fixed $\alpha$ we have the constraint that $\alpha = \alpha_L + \alpha_U$ (3.1) which gives

$$\alpha_U = \alpha - \alpha_L.$$  

As such, we wish to find the value of $\alpha_L$ which minimize the distance

$$W(\alpha_L) = F^{-1}\left(1 - \alpha + \alpha_L, DFT, DFE, \hat{\lambda}_{adj}\right) - F^{-1}\left(\alpha_L, DFT, DFE, \hat{\lambda}_{adj}\right),$$

for

$$\alpha_L \in (0, \alpha).$$

A method for finding $\alpha_L$ such that $W(x)$ is minimal is by the Golden Section Search (GSS) as described in Press et al. (2007). For a unimodal function $W(x)$ defined for $x \in [a, b]$, the Golden Section Search method seeks to bracket the value $k$ in $[a, b]$ such that

$$W(k) < W(x) \text{ for all other } x \in [a, b].$$

The value $k$ is considered found when brackets are
reached whose distance is within a user-specified tolerance level. The GSS procedure is named for its use of the mathematical constant
\[ \varphi = \frac{-1 + \sqrt{5}}{2} \approx 0.618034 \]  
(3.9)

The ends points \( a \) and \( b \) are used to construct two new possible bracket points. We calculate the first new point, which we call \( c \), by
\[ c = a + \varphi (b - a) \]  
(3.10)
and the second new point, which we call \( d \), by
\[ d = a + \varphi^2 (b - a) \]  
(3.11)

These new points are then evaluated with the function \( W(x) \) such that
\[ u = W(c) \]  
(3.12)
and
\[ v = W(d) \]  
(3.13)

One of two cases may arise:

(1) \( u > v \)

(2) \( u \leq v \)

For the first case (Figure 3.1.a) the value \( k \) must be in \([a, c]\) since the function \( W(x) \) is unimodal. The interval \([a, c]\) will then have width \( c - a = \varphi (b - a) \) (3.10). For the second case (Figure 3.1.b) the value \( k \) must be in \([d, b]\), again since the function \( W(x) \) is unimodal. Using the equation \( d = a + \varphi^2 (b - a) \) (3.11) and the fact that \( \varphi^2 = 1 - \varphi \), it can be shown that \( b - d = \varphi (b - a) \). Therefore,
Figure 3.1.a: Update Bracket Interval for Minimand for $v$ less than $u$

Figure 3.1.b: Update Bracket Interval for Minimand for $v$ greater than $u$
that is, the interval width for case (1) equals the interval width for case (2). That is, we get a consistent reduction factor. Additionally,

\[ \varphi = \frac{\varphi^2(b-a)}{\varphi(b-a)} = \frac{d-a}{b-d} \]  

(3.15)

and

\[ \varphi = -1 + \frac{1}{\varphi} = -1 + \frac{\varphi(b-a)}{\varphi^2(b-a)} = -\frac{d-a}{d-a} + \frac{\varphi(b-a)}{d-a} = \frac{-d-a+\varphi(b-a)}{d-a} = \frac{c-d}{d-a} \]

(3.16)

so that

\[ \frac{c-d}{d-a} = \frac{d-a}{b-d} \]

(3.16)

that is, using \( \varphi \) to pick \( c \) and \( d \) maintains the spacing among points as the brackets are updated. Therefore, we only need to calculate one new point and make one new function evaluation per interval reduction.
3.4 Parameter Settings

Based on the balanced one-way ANOVA model, true power and estimated power were obtained for a range of population conditions and simulation outcomes, respectively, for three sizes of number of treatments, \( p = 2, 3, \) and 4.

The two-treatment class has a single symmetric mean arrangement. The three-treatment class has one symmetric mean arrangement and one asymmetric mean arrangement. The four-treatment class has two symmetric mean arrangements and one asymmetric mean arrangement. For each mean arrangement a range of population standard deviations and sample sizes are considered. These population conditions are summarized in Table 3.1. Table 3.1 also includes under each mean arrangement the sum of squared differences between population means and their grand population mean which is the numerator of the noncentrality parameter \( \lambda \) (1.10). These values were chosen so that treatment populations should be discernibly different at a 95% empirical interval when the population standard deviation is “small” and/or sample size is “large,” and indiscernibly different when the population standard deviation is “large” and/or sample size is “small.”
Table 3.1: Summary of Parameter Settings for Simulations

<table>
<thead>
<tr>
<th>Case</th>
<th>Mean $= \mu_i$</th>
<th>$\sum (\mu_i - \bar{\mu}_i)^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Sample Symmetric</td>
<td>30, 40</td>
<td>50.0000</td>
</tr>
<tr>
<td>Three-Sample Symmetric</td>
<td>30, 35, 40</td>
<td>50.0000</td>
</tr>
<tr>
<td>Three-Sample Asymmetric</td>
<td>30, 40, 40</td>
<td>66.6667</td>
</tr>
<tr>
<td>Four-Sample Symmetric A</td>
<td>30, 35, 35, 40</td>
<td>50.0000</td>
</tr>
<tr>
<td>Four-Sample Symmetric B</td>
<td>30, 30, 40, 40</td>
<td>100.0000</td>
</tr>
<tr>
<td>Four-Sample Asymmetric</td>
<td>30, 40, 40, 40</td>
<td>75.0000</td>
</tr>
<tr>
<td>Standard Deviation $= \sigma$</td>
<td>2.5 to 10.0 by 0.5 and 20.0</td>
<td></td>
</tr>
<tr>
<td>Sample Size $= n$</td>
<td>5 to 14 by 1 and 15 to 50 by 5</td>
<td></td>
</tr>
<tr>
<td>Significance Level $= \alpha$</td>
<td>0.05</td>
<td></td>
</tr>
</tbody>
</table>
3.5 Iteration

For each combination of mean arrangement, population standard deviation, and sample size, 100 simulations were produced. Each simulation produced $p$ random samples where $p$ denotes the number of treatment populations. The $p$ samples were produced to have equal samples size $n$. Each random sample was produced from a random seed value, an integer between 1 and 1,000,000,000, obtained through the website www.random.org. Random samples were constructed to reflect a normally distributed population with mean $\mu_i$ and equal population standard deviation where $i = 1, \ldots, p$. Note that random samples were initially generated in part by the SAS® software function RANNOR. RANNOR was seen to produce dependent samples when called multiple times within the IML procedure. Therefore, RANNOR was encapsulated in a macro which resulted in independent random number generation.

For each simulation, the following calculations were made from the generated data. An ANOVA table was calculated as outlined in 3.1.1. True power and estimated power were calculated as outlined in 3.1.2. Confidence limits for true power were calculated and optimized as outlined in 3.1.2 and 3.1.3, respectively. Tolerance for bracket intervals in 3.1.3 was set to 1E-4 as suggested by the procedure FMINBND in MATLAB (2009) and the package OPTIMIZE in R (R Development Core Team 2009).

Several problems were noted with the SAS software function PROBF which was recommended by Thomas (1997) for calculating power (SAS Institute Inc. 2006a). First, the SAS function PROBF reports an invalid argument when the supplied noncentrality parameter is large with respect to numerator degrees of freedom, denominator degrees of freedom, and $F$ critical value. This problem is noted by O'Brien (1988). Taylor and Muller (1996) suggest setting the noncentrality large enough such that PROBF evaluates to zero or one where
appropriate. The SAS software function CDF (SAS Institute Inc. 2006b) offers some relief to the PROBF issue though invalid arguments can still be had for extreme parameter values.

Second, PROBF reports an error when the supplied noncentrality parameter is negative. Gerard et al. (1998) suggest setting the noncentrality parameter to zero. For some random random seed values the data produced result with a negative adjusted estimated noncentrality parameter. The negative value occurs in some instances when adjusting the estimated noncentrality parameter (2.5), as noted previously.

Finally, FNONCT reports an error when either of the probabilities $1 - \alpha_L$ or $\alpha_U$ are greater than the probability associated with the observed $F$ statistic from a central $F$-distribution with degrees of freedom DFT and DFE. Thomas (1997) suggests setting the noncentrality parameter equal to zero. In this case the error associated with FNONCT is noted in the SAS software documentation.

For each combination of mean arrangement, population standard deviation, and sample size, calculations for the 100 simulations are averaged. Notable averages include average estimated power and average upper and lower confidence limits about true power. The median for estimated power was also determined. Missing values were deleted.

The SAS program (Version 9.1.3) used can be found in Appendix B. Flow charts for the SAS program can be found in Appendix A (arranged through OmniGraffe Pro® version 5). This document was processed using the iWork ’09® application Pages. Mathematical expression were set using MathType® version 6.
4 Results

Output for simulation summaries is plotted as power versus sample size for population standard deviations 2.5 to 10.0 by 0.5 and 20.0. Power ranges from 0.0 to 1.0 and sample size ranges from 5 to 14 by 1 and 15 to 50 by 5. The average p-value of the test is presented in each plot. The complete display of simulation plots can be found in Appendix C. In the Results, four plots are presented from each treatment class for each treatment mean arrangement, power plotted against sample size, where sample size ranges from 1 to 35 at population standard deviations of 2.5, 5.0, 10.0, and 20.0. A maximum sample size of 35 is presented since power approached 1.0 for sample sizes over 35 for the simulation configuration used here. Population standard deviations 2.5 and 20.0 represent two extreme cases for variability while population standard deviations 5.0 and 10.0 represent two medium cases for variability. Plots for each treatment arrangement appear as a moving-window like landscape from a moving vehicle.
4.1 Two-Sample Symmetric Case

Plots for population standard deviations 2.5, 5.0, 10.0, and 20.0 with a maximum sample size of 35 are seen in Figure 4.1.a, 4.1.b, 4.1.c, and 4.1.d, respectively. The p-value for the test decreases as the sample size increases, such that the null hypothesis can be rejected when $n \geq 5$, $n \geq 6$, $n \geq 14$, and $n > 35$ for respective values of $\sigma = 2.5$, $\sigma = 5.0$, $\sigma = 10.0$, and $\sigma = 20.0$. True power is always between the average lower and upper confidence limits for true power. Power and confidence limits are higher and closer together at larger sample sizes, as expected. For $\sigma = 2.5$, true power is very close to the average upper limit. For $\sigma = 5.0$, true power is closer to the average upper limit than the lower limit for $n < 30$; is half-way between the average upper and lower limits when both limits and true power approach 1.0 for $n \geq 30$. For $\sigma = 10.0$, true power starts (at $n = 5$) around the average lower limit, and is located half-way between average confidence limits at $n = 9$. For $n > 9$, true power is closer to the average upper limit than the average lower limit. When $\sigma = 20.0$, true power is located near the average lower limit (for $n = 5$), and obtains an approximate location half-way between average limits for $n = 30$. True power exhibits an increasing positive trend toward the average upper limit for each plot, though the incline of the curve decreases for larger values of $\sigma$.

Power curves appear generally ordered as,

$$\text{power}_{\text{true}} > \text{power}_{\text{med}} > \text{power}_{\text{avg}}.$$  
(4.1)

The general ordering of power curves is strictly seen for $\sigma$ of 2.5 and 5.0. For $\sigma = 10.0$, average estimated power starts (at $n = 5$) above median estimated power but adopts the general ordering for sample sizes 10 and greater. When $\sigma$ equals 20.0, average estimated power starts (at $n = 5$) above true power, and median estimated power starts below true power. Average estimated power appears (for $\sigma = 20.0$) below true power at sample sizes 13 and greater, and below median estimated power at sample sizes 35 and greater. For samples sizes greater than 35, the general ordering appears to hold for any value of $\sigma$ presented here.
Two-Sample Symmetric Case: Power vs Sample Size

Figure 4.1.a: Sigma 2.5

Figure 4.1.b: Sigma 5.0

Figure 4.1.c: Sigma 10.0

Figure 4.1.d: Sigma 20.0
4.2 Three-Sample Symmetric Case

Plots for population standard deviations 2.5, 5.0, 10.0, and 20.0 with a maximum sample size of 35 are seen in Figure 4.2.a, 4.2.b, 4.2.c, and 4.2.d, respectively. Similarly to the two-sample case, the p-value for the test decreases as the sample size increases, such that the null hypothesis can be rejected when \( n \geq 5 \), \( n \geq 8 \), \( n \geq 16 \), and \( n > 35 \) for values of \( \sigma = 2.5 \), \( \sigma = 5.0 \), \( \sigma = 10.0 \), and \( \sigma = 20.0 \), respectively. True power is always contained within the average limits. As with the two-sample case, power and confidence limits tend to be higher and closer together for larger sample sizes. True power is very close to the average upper limit for \( \sigma = 2.5 \). For \( \sigma = 5.0 \), true power is located near the upper average limit and is approximately half-way between the average limits for \( n \geq 35 \). When \( \sigma = 10.0 \), true power is close to the average lower limit (at \( n = 5 \)), and is approximately half-way between the confidence limits at \( n = 12 \). For \( \sigma = 20.0 \), true power is nearer to the average lower limit than the average upper limit for sample sizes up to 30, but for \( n = 35 \) true power is approximately half-way between the confidence limits. At all values \( \sigma \) considered here, true power tend toward the average upper limit.

As with the two-sample case, the general ordering (4.1) tends to hold. The general ordering of power is seen for \( \sigma = 2.5 \) and 5.0. For \( \sigma = 10.0 \), average estimated power starts (at \( n = 5 \)) above the median estimated power, but takes on the general ordering at sample sizes 13 and greater. For \( \sigma = 20.0 \), true power begins (at \( n = 5 \)) below average estimated power, and above median estimated power. Average estimated power (for \( \sigma = 20.0 \)) is below true power at sample sizes 30 and greater, and below median estimated power at sample sizes greater than 35. The general position of power curves (4.1) tends to hold when samples sizes are large.
Three-Sample Symmetric Case: Power vs Sample size

Figure 4.2.a: Sigma 2.5

Figure 4.2.b: Sigma 5.0

Figure 4.2.c: Sigma 10.0

Figure 4.2.d: Sigma 20.0
4.3 Three-Sample Asymmetric Case

Plots for population standard deviations 2.5, 5.0, 10.0, and 20.0 with a maximum sample size of 35 are seen in Figure 4.3.a, 4.3.b, 4.3.c, and 4.3.d, respectively. Again, the p-value for the test decreases as the sample size increases, such that the null hypothesis can be rejected when \( n \geq 5 \), \( n \geq 6 \), \( n \geq 14 \), and \( n > 35 \) for respective values of \( \sigma = 2.5 \), \( \sigma = 5.0 \), \( \sigma = 10.0 \), and \( \sigma = 20.0 \). True power is between the average upper and lower limits for all values of the population standard deviation considered here. As with the previous cases, power and confidence limits are higher and closer together for larger sample sizes. For \( \sigma = 2.5 \), true power is indistinguishable from the average upper limit. For \( \sigma = 5.0 \), true power is closer to the average upper limit than the average lower limit, and true power is approximately half-way between the average upper and lower limits when both limits approach 1.0 (\( n \geq 30 \)). For \( \sigma = 10.0 \), the average upper and lower limits are centered around true power for \( n = 9 \). When \( \sigma = 20.0 \), true power is located near the average lower limit (for \( n = 5 \)), and is approximately half-way between the average upper and lower limits at \( n = 30 \). As with previous cases, true power shows an upward trend toward the average upper limit for larger sample sizes.

As with prior cases, the general ordering of true power, median estimated power, and average estimated power (4.1) tends to hold. For \( \sigma = 2.5 \) and 5.0, the general ordering of power curves can be seen in Figure 4.3.a. For \( \sigma = 10.0 \), average estimated power starts (at \( n = 5 \)) above the median estimated power, but adopts the general ordering for sample sizes larger than 10. When \( \sigma = 20.0 \), average estimated power starts (at \( n = 5 \)) above true power, and median estimated power starts (at \( n = 5 \)) below true power. Average estimated power appears below true power for sample sizes 15 and greater, and below median estimated power for sample sizes greater than 35. As with the previous cases, the general order of the power curves (4.1) is maintained for large sample sizes.
Three-Sample Asymmetric Case: Power vs Sample size

Figure 4.3.a: Sigma 2.5

Figure 4.3.b: Sigma 5.0

Figure 4.3.c: Sigma 10.0

Figure 4.3.d: Sigma 20.0
4.4 Four-Sample Symmetric Case A

Plots for population standard deviations 2.5, 5.0, 10.0, and 20.0 with a maximum sample size of 35 are seen in Figure 4.4.a, 4.4.b, 4.4.c, and 4.4.d, respectively. For larger sample sizes, the p-value for the test decreases such that the null hypothesis can be rejected when
\[ n \geq 5, \quad n \geq 8, \quad n \geq 17, \quad \text{and} \quad n > 35 \]
for respective values of \( \sigma = 2.5, \sigma = 5.0, \sigma = 10.0, \) and \( \sigma = 20.0 \). True power, as seen in previous cases, is always between the average upper and lower limits for true power. Further, true power and average confidence limits tend toward 1.0 for large sample sizes. For \( \sigma = 2.5 \), true power is identical with the average upper confidence limit. For \( \sigma = 5.0 \), average confidence limits center are symmetric about true power when both limits equal 1.0. For \( \sigma = 10.0 \), average upper and lower confidence limits are centered around true power when \( n = 14 \). For \( \sigma = 20.0 \), true power is close to the average lower limit. True power displays a positive increasing trend toward the average upper limit, where the steepness of the curve reduces for larger values of \( \sigma \).

The general order of the power curves (4.1) again tends to occur. The typical ordering always occurs for \( \sigma = 2.5 \) and 5.0. For \( \sigma = 10.0 \), median estimated power starts (at \( n = 5 \)) below the median estimated power, but maintains the typical ordering at sample sizes 14 and greater. For \( \sigma = 20.0 \) and \( n = 5 \), power curves are ordered as average estimated power, true power, and median estimated power. The average estimated power (for \( \sigma = 20.0 \)) appears below true power at sample sizes 15 and greater, and below median estimated power at sample sizes larger than 35. As with previous cases, the typical order of power curves (4.1) is seen for larger sample sizes.
4.5 Four-Sample Symmetric Case B

Plots for population standard deviations 2.5, 5.0, 10.0, and 20.0 with a maximum sample size of 35 are seen in Figure 4.5.a, 4.5.b, 4.5.c, and 4.5.d, respectively. As the sample size increases, the p-value for the test decreases such that the null hypothesis can be rejected when \( n \geq 5 \), \( n \geq 5 \), \( n \geq 10 \), and \( n > 35 \) for \( \sigma = 2.5 \), \( \sigma = 5.0 \), \( \sigma = 10.0 \), and \( \sigma = 20.0 \), respectively. As with previous cases, average lower and upper confidence limits always contain true power. True power and confidence limits become narrower and approach 1.0 for larger sample sizes. For \( \sigma = 2.5 \), true power is identical to the average upper limit. For \( \sigma = 5.0 \), true power is just below the average upper limit and is centered between average limits when both limits approach 1.0. For \( \sigma = 10.0 \), true power begins (at \( n = 5 \)) around the average lower limit, and centers between the average confidence limits for \( n = 8 \). When \( \sigma = 20.0 \), average confidence limits are symmetric around true power for \( n = 25 \). As with prior cases, true power has an increasing positive trend toward the average upper limit for each plot. Notably, the rate of increase of the true power curve decreases for larger values of \( \sigma \).

As seen previously, average estimated power, median estimated power, and true power are consistently between the lower and upper confidence limits for true power. Power curves have the typical order (4.1) as in previous cases. The usual ordering of power is seen for \( \sigma = 2.5 \) and 5.0. For \( \sigma = 10.0 \), average estimated power starts (at \( n = 5 \)) above the median estimated power, yet takes on the general ordering at sample sizes 10 and greater. For \( \sigma = 20.0 \), true power is below average estimated power and above median estimated power for \( n = 5 \). Further, for \( \sigma = 20.0 \), average estimated power appears below true power at sample sizes 13 and greater, and below median estimated power at sample sizes 30 and greater. As seen earlier, the general ordering (4.1) holds for large sample sizes.
Four-Sample Symmetric Case B: Power vs Sample size

Figure 4.5.a: Sigma 2.5

Figure 4.5.b: Sigma 5.0

Figure 4.5.c: Sigma 10.0

Figure 4.5.d: Sigma 20.0
4.6 Four-Sample Asymmetric Case

Plots for population standard deviations 2.5, 5.0, 10.0, and 20.0 with a maximum sample size of 35 are seen in Figure 4.6.a, 4.6.b, 4.6.c, and 4.6.d, respectively. The p-value for the test decreases as the sample size increases, as expected, such that the null hypothesis can be rejected when \( n \geq 5 \), \( n \geq 6 \), \( n \geq 14 \), and \( n > 35 \) for respective values of \( \sigma = 2.5 \), \( \sigma = 5.0 \), \( \sigma = 10.0 \), and \( \sigma = 20.0 \). True power lies between the average lower and upper limits for all values of \( \sigma \). As seen in prior cases, true power and average confidence limits are higher and closer together for large sample sizes. For \( \sigma = 2.5 \), true power is indistinct from the average upper limit. For \( \sigma = 5.0 \), true power is close to the average upper limit and is approximately very close (at power approximately 1.0) with the average upper limit at \( n = 10 \). For \( \sigma = 10.0 \), average confidence limits are equally spaced about true power when \( n = 9 \). When \( \sigma = 20.0 \), average upper and lower limits are centered around true power for \( n = 30 \). As usual, true power shows an increasing upward trend toward the average upper confidence limit for each plot. Notably, the incline of the curve for true power climbs with less rapidity for larger values of \( \sigma \).

The power curves are hold the usual ordering (4.1) as seen previously. The general ranking of power curves (4.1) is exemplified for \( \sigma = 2.5 \) and 5.0. For \( \sigma = 10.0 \), average estimated power starts (at \( n = 5 \)) above the median estimated power but takes on the general ordering at sample sizes 9 and greater. For \( \sigma = 20.0 \), average estimated power starts above true power, while median estimated power starts below true power. Average estimated power (for \( \sigma = 20.0 \)) appears below true power at sample sizes 20 and greater, and below median estimated power at sample sizes greater than 35. When sample sizes are large, the power curves follows the general ordering (4.1).
Four-Sample Asymmetric Case: Power vs Sample size

Figure 4.6.a: Sigma 2.5

Figure 4.6.b: Sigma 5.0

Figure 4.6.c: Sigma 10.0

Figure 4.6.d: Sigma 20.0
4.7 Symmetric Cases

For symmetric treatment arrangements, confidence limits for the two-sample case, three-sample case, and four-sample case A do not appear substantially different. As an exception, confidence limits for the four-sample case B are narrower than the other symmetric cases. Average lower and upper confidence limit widths are ordered, from least to greatest, as four-sample case B, four-sample case A, three-sample case, and two-sample case. Average lower and upper confidence limits always contain true power.

For all symmetric cases, true power shows a positive increasing trend toward the average upper limit. For $\sigma = 2.5$ and 5.0, all symmetric cases show true power near the average upper limit. For $\sigma = 10.0$, true power starts (at $n = 5$) close to the average lower limit, reaches a half-way distance between average limits for $n = 5$ to 15, and then proceeds toward the average upper limit. For $\sigma = 20.0$, true power is generally close to the average lower limit for all symmetric cases, though the two-sample case and four-sample case B reach a half-way distance between average limits at $n = 30$ and $n = 25$, respectively.

4.8 Asymmetric Cases

For the two asymmetric cases, confidence interval widths are noticeably different from one another. The four-sample case has narrower average lower and upper confidence limits for true power than the three-sample case. True power is always between the average lower and upper confidence limits.

For both asymmetric cases, true power exhibits an upward trend toward the average upper limit. For $\sigma = 2.5$ and 5.0, true power is never half-way between the average upper and lower limits. For $\sigma = 10.0$, true power starts (at $n = 5$) around the average lower limit, reaches the half-way distance between average limits around $n = 8$ to 9, and then moves closer to the
average upper limit. For $\sigma = 20.0$, true power is close to the average lower limit, but reaches
the half-way distance between average limits around $n = 30$ to $35$.

4.9 Symmetric Cases versus Asymmetric Cases

In both symmetric and asymmetric cases, true power is always between the average
lower and upper limits. Generally, symmetric cases appear to have wider average confidence
limits than the asymmetric cases, with the exception of the four-sample case B, which has the
narrowest confidence limits among all cases considered. Symmetric and asymmetric cases
both show that true power is closer to the average upper limit than the lower limit for large
sample sizes. In terms of average limit widths, cases are ordered from least to greatest as
follows: four-sample case B, four-sample asymmetric case, three-sample asymmetric case,
four-sample case A, three-sample symmetric case, then the two-sample case.

Both symmetric cases and asymmetric cases have power curves typically ordered, from
highest to lowest, as true power, median estimated power, then average estimated power. The
typically ordering of power curves is seen for $\sigma = 2.5$ and $5.0$. For $\sigma = 10.0$, average
estimated power starts (at $n = 5$) above median estimated power, but crosses below median
estimated power for samples sizes between 10 to 15. For $\sigma = 20.0$, average estimated power
is above true power, and median estimated power is below true power. Further (for $\sigma = 20.0$),
average estimated power drops below true power, then proceeds below median estimated
power as sample sizes increase. All power curves are within the average lower and upper
limits.
5 Conclusion

Simulations were constructed for a range of population conditions under the balanced one-way ANOVA model. Each simulation case was iterated 100 times and plots for average estimated power, median estimated power, true power, average lower 95% limit, and average upper 95% limit were produced. An average p-value for F-tests was also plotted for each simulation setup.

When the population standard deviation was small (from $\sigma = 2.5$ to 5.0), confidence limits for true power were wider for small sample sizes and narrower for medium-to-large sample sizes. Under all sample sizes considered here, average limits contained true power. Notably, true power was never half-way between the average upper and lower confidence limits except when sample sizes were large and both limits were very close to 1.0. Additionally, true power was always close to the average upper limit and above the average estimate for power. When significant differences exist among the population treatment means, the average p-value is small, true power is large, and the average retrospective estimate for power provides a higher lower limit for true power than the average lower limit.

When the population standard deviation was large (from $\sigma = 10.0$ to 20.0), average confidence limits for true power were generally wide for small and large sample sizes. True power was contained within the average limits: close to the average lower limit for small sample sizes and progressive toward the average upper limit for large sample sizes. In all cases, the average estimate for power was present above true power (when sample sizes were small) and below true power (when sample sizes were large). When the null hypothesis of equal population means was difficult to reject, i.e. when the p-value of the test was large, true power was low and confidence limits for true power were wide.

When the null hypothesis cannot be rejected, the test performed is seen as under powered. The results here show that true power is hard to determine retrospectively as laid out
in Thomas (1997) when the null hypothesis cannot be rejected, since the average 95% confidence limits were wide and the average estimate for power fails to act as a lower bound for true power under all values of $\sigma$.

The presence of nonsignificant results can lead researchers to question the power of their test. When power has not been calculated a priori, researchers may see retrospective methods as a viable technique for “figuring out what power was.” As an example of Thomas’ technique to calculate power post hoc, two possible consulting situations are described below. Each uses an observed $F$-statistic, numerator and denominator degrees of freedom, and the desired level of significance. The $p$-value of the test is also provided.

For the first example, consider an observed $F$-statistic of 1.5 with 1 numerator degree of freedom and 40 denominator degrees of freedom with a $p$-value of 0.2278. An estimate for true power is

$$\hat{\text{power}} = 1 - F(4.085, 1, 40, 0.425) = 0.0975,$$

where $F_{0.05,1,40} = 4.085$ is the upper tail critical value from a central $F$-distribution with 1 numerator degree of freedom and 40 denominator degrees of freedom, and

$$\hat{\lambda}_{adj} = \left( (1.5)(1)(40 - 2) \right) / 40 - 1 = 0.425$$

is the adjusted estimated noncentrality parameter. The minimized 95% confidence limits for true power are

$$\hat{\text{power}}_L = 1 - F(4.085, 1, 40, 0.00000) = 0.05 \text{ and}$$

$$\hat{\text{power}}_U = 1 - F(4.085, 1, 40, 8.2801704) = 0.8017,$$

where the estimated lower and upper noncentrality parameters $\hat{\lambda}_L = 0.000000$ and $\hat{\lambda}_U = 8.2801704$ are determined by the lower and upper tail probabilities $\alpha_L = 0.000048$ and
\( \alpha_u = 0.049952 \) through use of the Golden Section Search method (Section 3.3). Using the estimate for power, a researcher would conclude that the test was under powered since this estimated value is below 0.8 (Cohen 1988). Notably, using the point estimate to represent true power is uninformative since the confidence limits have true power anywhere from 0.05 to 0.8017. The confidence interval here is similar to the confidence limits in Figure 4.1.d for the symmetric two population mean arrangement with \( \sigma = 20.0 \) at a sample size of 22. Figure 4.1.d likens the clients’ observations to an extreme case where discerning treatment groups as different is difficult.

For the second example, consider an observed \( F \)-statistic of 4.05 with 1 numerator degree of freedom and 40 denominator degrees of freedom with a p-value of 0.0509. Power is estimated as

\[
\text{power} = 1 - F(4.085 | 1, 40, 2.8475) = 0.3773,
\]

where \( F_{0.05,1,40} = 4.085 \) is the upper tail critical value from a central \( F \)-distribution with 1 numerator degree of freedom and 40 denominator degrees of freedom, and

\[
\hat{\lambda}_{\text{adj}} = \frac{((4.05)(1)(40 - 2))}{40 - 1} = 2.8475
\]

is the adjusted estimated noncentrality parameter.

Through the Golden Section Search (Section 3.3), the minimized lower and upper tail probabilities are found to be \( \alpha_L = 0.000048 \) and \( \alpha_U = 0.049952 \) which yield lower and upper estimated noncentrality parameters \( \hat{\lambda}_L = 0.000000 \) and \( \hat{\lambda}_U = 13.59003 \). The estimated 95% lower and upper confidence limits for true power are given by

\[
\text{power}_L = 1 - F(4.085 | 1, 40, 0.000000) = 0.05 \quad \text{and}
\]

\[
\text{power}_U = 1 - F(4.085 | 1, 40, 13.59003) = 0.9491.
\]
Using the estimate for power, a researcher would conclude that the test was under-powered since this estimated value is below 0.8 (Cohen 1988). Notably, using the point estimate to represent true power is uninformative since the confidence limits have true power anywhere from 0.05 to 0.9491. The confidence interval here is similar to the confidence limits in Figure 4.1.d for the two-sample symmetric case with $\sigma = 20.0$ at a sample size of 8. Figure 4.5.c likens the client's observations to the case where discerning treatment groups as different is difficult.

When determining the power of a test post hoc, power is not calculated but estimated. For researchers looking to resolve power after observations have been collected, confidence limits for true power can help quantify the uncertainty inherent in a post hoc power calculation. Examples 1 and 2, along with the plots in the results section, show a high degree of variability when the null hypothesis cannot be rejected. Hence, researchers are recommended to incorporate power analyses in the planning stages of their studies, as opposed to waiting until data have been collected.
References

American Statistical Association (1999), 'Ethical Guidelines for Statistical Practice'.


Appendix A: Two-Sample Flow Chart of Power Simulations

* Datasets Defined
randSeed := dataset of random seeds

* Program Description
twoSample.sas acts as a controller interface which accepts user defined values for two population means, upper and lower bounds for a range of population standard deviations, a maximum sample size, a Type I error rate, and the number of iterations per sample size to population standard deviation combination.

* rand2.csv is a two column file of random numbers generated from http://www.random.org. For each iteration two random values are used to seed two random samples.

* Program Name: twoSample.sas

* Flowchart page 2
* multipleSim2 calls the macro multipleSim2.sas.
* This macro manages the production of simulations. Calls are made to the macro simulation2.sas which performs calculations based on Thomas 1997.
* Calculations for each simulation are stored in a dataset called multipleSimOut.

* Flowchart page 14
* avgMedMultipleSim2 calls the macro avgMedMultipleSim2.sas.
* This macro manages the consolidation of simulations. Calls to the macro avgMedSim2.sas average over the rows of multipleSimOut by the user defined number of iterations. avgMedSim2 also determines the median of the calculated power values of the rows.
* Calculations for each iteration block are stored in a dataset called avgMedMultipleSimOut.
The text below the diagram reads:

* Program Description

multipleSim2.sas manages the production of simulations.

* Loop through sample sizes 5 to 14 by 1.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Loop through sample sizes 15 to the maximum user defined sample size by 5.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Flowchart page 5
* simulation2 calls the macro simulation2.sas
* This macro performs calculations based on Thomas 1997

Program Name: multipleSim2.sas
* Loop through population standard deviation 20 to user defined upper population standard deviation by 20.

* Loop through sample sizes 5 to 14 by 1.
  * Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Loop through sample sizes 15 to the maximum user defined sample size by 5.
  * Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Flowchart page 5
  * simulation2 calls the macro simulation2.sas
  * This macro performs calculations based on Thomas 1997

---

Appendix A: Two-Sample Flow Chart of Power Simulations
* Gather simulations into a dataset

* Datasets Defined:
  multipleSimOut := dataset of produced simulations

* Variables Defined:
  \( \mu_1 \) := mean for population one
  \( \mu_2 \) := mean for population two
  \( \sigma_{\text{Min}} \) := minimum population standard deviation
  \( \sigma_{\text{Max}} \) := maximum population standard deviation
  \( i_{\text{Max}} \) := number of iterations
  \( \text{sampleSizeMax} \) := maximum sample size
  \( \text{sigLevel} \) := Type I error rate
  \( \text{count} \) := simulation identification
  \( \text{block} \) := sets location of random seeds for simulation5k.sas

Output dataset multipleSimOut

Stop
Program Name: simulation2.sas

* Program Description
simulation2.sas performs a hypothesis test of no difference for two population means. Random samples are normally distributed with respective means \( \mu_A \) and \( \mu_B \) and equal population standard deviation \( \sigma \). Random samples are generated using seed values from the dataset randSeed. Power calculations are based on methods outlined in Thomas 1997. Calculations are saved to a dataset named 'sim#' where # represents a simulation count.

* Program Code

```sas
/* Merge datasets sample1 and sample2 to produce dataset sampleTwoPops */
eps = 1e-4
n = nrow( X )
p = ncol( X )
J = J( p , 1 )
muG = ( &muA + &muB ) / p
xBar = X( | +, | )` / n
sst = n * xBar` * xBar - ( n / p ) * ( xBar` * J )**2
df1 = p - 1
mst = sst / df1
sse = ssq( X ) - ( xBar` * xBar ) * n
df2 = p * ( n - 1 )
mse = sse / df2
fSamp = mst / mse
```

* Flowchart page 13
* sampleNormPop2 calls the macro sampleNormPop2.sas
* This macro generates are random samples of size 'sampleSize' with mean \( \mu_A \) and population standard deviation \( \sigma \) from a seed value in the dataset randSeed.
* Produces dataset sample1

* Merge datasets sample1 and sample2 to produce dataset sampleTwoPops

* X holds values from the dataset sampleTwoPops

* Flowchart page 13
* sampleNormPop2 calls the macro sampleNormPop2.sas
* This macro generates are random samples of size 'sampleSize' with mean \( \mu_B \) and population standard deviation \( \sigma \) from a seed value in the dataset randSeed.
* Produces dataset sample2
truePower = 1 - cdf( 'F', fCrit, df1, df2, trueLambda )
estLambda = fSamp * ( p - 1 )
adjEstLambda = ( estLambda * ( df2 - 2 ) / df2 ) - df1

adjEstLambda > 0 ?

no
adjEstLambda = 0

yes

* For some random seed pairs, adjEstLambda is negative. A negative adjusted estimated noncentrality parameter occurs when the expression '( estLambda * (df2 - 2) / df2 )' is less than df1

* alphaL takes on a value from alphaLU

* input alphaL

* Flowchart page 10

* Minimize lower rejection region by minimizing confidence interval width.

* Suggested by Dr. Leigh Murray (2009) and possibly implied by Thomas 1997.
alphaU = alpha - alphaL
adjAlphaL = 1 - alphaL
probCenF = cdf('F', fSamp, df1, df2)

lambdaL = 0

* For fixed observed F statistic, numerator degrees of freedom, and denominator degrees of freedom, fnonct reports an error when the probability supplied is greater than the probability from the cdf of a central F distribution.

* * As the probability supplied approaches the probability from a central F distribution, the noncentrality parameter tends toward 0.

* adjAlphaL is occasionally problematic.

lambdaL = fnonct(fSamp, df1, df2, adjAlphaL)

adjAlphaL > probCenF ?

lambdaL = 0

* If the adjusted lower rejection region is greater than the probability from the cdf of a central F distribution then set the noncentrality parameter to zero.

* If the upper rejection region is greater than the probability from the cdf of a central F distribution then set the noncentrality parameter to zero.

lambdaU = fnonct(fSamp, df1, df2, adjAlphaL)

alphaU > probCenF ?

lambdaU = 0

Appendix A: Two-Sample Flow Chart of Power Simulations
Appendix A: Two-Sample Flow Chart of Power Simulations

\[ \text{estPower} = 1 - \text{cdf}( 'F', fCrit, df1, df2, \text{adjEstLambda} ) \]
\[ \text{powerL} = 1 - \text{cdf}( 'F', fCrit, df1, df2, \lambda_L ) \]
\[ \text{powerU} = 1 - \text{cdf}( 'F', fCrit, df1, df2, \lambda_U ) \]

\[ Y[1] = \mu_A \]
\[ Y[2] = \mu_B \]
\[ Y[3] = \sigma \]
\[ Y[4] = \text{sampleSize} \]
\[ Y[5] = \alpha \]
\[ Y[6] = p\text{Value} \]
\[ Y[7] = \text{trueLambda} \]
\[ Y[8] = \text{truePower} \]
\[ Y[9] = \text{estLambda} \]
\[ Y[10] = \text{adjEstLambda} \]
\[ Y[11] = \lambda_L \]
\[ Y[12] = \lambda_U \]
\[ Y[13] = \text{estPower} \]
\[ Y[14] = \text{powerL} \]
\[ Y[15] = \text{powerU} \]

Output dataset sim&simNum

Stop
Appendix A: Two-Sample Flow Chart of Power Simulations

* Variables Defined

- $\mu_A$ := mean for population one
- $\mu_B$ := mean for population two
- $\sigma$ := population standard deviation
- $\text{seedNum}$ := location of random seeds
- $\text{sampleSize}$ := sample size
- $\alpha$ := level of significance
- $\text{simNum}$ := simulation identification
- $X$ := samples from normal populations
- $\epsilon$ := tolerance for Golden Section Search methods
- $n$ := number rows
- $p$ := number columns
- $\bar{x}$ := sample mean
- $\text{sst}$ := sum of squares for treatment
- $df_1$ := numerator degrees of freedom
- $\text{mst}$ := mean square for treatment
- $\text{sse}$ := sum of squares for error
- $df_2$ := denominator degrees of freedom
- $\text{mse}$ := mean square for error
- $F_{\text{samp}}$ := observed F statistic
- $F_{\text{Crit}}$ := critical value from a central F distribution
- $p_{\text{Value}}$ := p-value for the test
- $\lambda_{\text{true}}$ := noncentrality parameter
- $\lambda_{\text{est}}$ := calculated noncentrality parameter
- $\lambda_{\text{adj}}$ := adjusted calculated noncentrality parameter
- $\alpha_{\text{L}}$ := optimized lower critical region
- $\alpha_{\text{U}}$ := optimized upper critical region
- $\text{probCentF}$ := probability of observed F statistic of the central F distribution
- $\text{estPower}$ := calculated power
- $\text{powerL}$ := lower confidence interval for calculated power
- $\text{powerU}$ := upper confidence interval for calculated power

* Datasets Defined

- $\text{sampleTwoPops}$ := random samples dataset
- $\text{sim&simNum}$ := simulation dataset
Start
Input ndf1, ddf2, ncp, sAlpha, tol

alphaA = 0
alphaB = sAlpha

\[ c = \frac{-1 + \sqrt{5}}{2} \]
\[ x_1 = c \cdot \alphaA + (1 - c) \cdot \alphaB \]
\[ x_2 = (1 - c) \cdot \alphaA + c \cdot \alphaB \]
\[ w(x_1, ndf1, ddf2, ncp, sAlpha) \]
\[ w(x_2, ndf1, ddf2, ncp, sAlpha) \]

\[ A \]

abs( alphaB - alphaA ) > tol ?

no

wx1 < wx2 ?

yes

wx1 = wx2

x2 = x1

wx2 = wx1

x1 = c \cdot \alphaA + (1 - c) \cdot \alphaB

yes

Output \( \frac{\alphaA + \alphaB}{2} \)

Stop

Program Name: alphaLU

Program Description

* The function alphaLU finds the minimum confidence interval width of a noncentral F-distribution using an implementation of the Golden Section Search method by Kincaid and Cheney 2009, and Jan Verschelde http://www.math.uic.edu/~jan/mcs471/Lec9/gss.pdf

* Flowchart page 12

* w returns the confidence interval width of a noncentral F-distribution

* wx2 takes on a value from w

* wx2 takes on a value from w

* wx2 takes on a value from w

* wx2 takes on a value from w
Variables Defined

\( \text{ndf1} := \text{numerator degrees of freedom} \)
\( \text{ddf2} := \text{denominator degrees of freedom} \)
\( \text{ncp} := \text{noncentrality parameter} \)
\( \text{slAlpha} := \text{level of significance} \)
\( \text{tol} := \text{tolerance for Golden Section Search method} \)
\( \text{alphaA} := \text{lower bound for optimized lower alpha} \)
\( \text{alphaB} := \text{upper bound for optimized lower alpha} \)
\( \text{c} := \text{Golden ratio constant reduction factor} \)
\( \text{x1} := \text{percentile associated with alphaA} \)
\( \text{x2} := \text{percentile associated with alphaB} \)
\( \text{wx1} := \text{height associated with x1} \)
\( \text{wx2} := \text{height associated with x2} \)
Program Name: w

Program Description
The function w returns a confidence interval width of a noncentral F-distribution

Variables Defined
alphaLow := lower critical region
ndf := numerator degrees of freedom
ddf := denominator degrees of freedom
ncent := noncentrality parameter
sl := level of significance
width := distance between percentiles

Start
Input alphaLow, ndf, ddf, ncent, sl

width = quantile( 'F', 1 - sl + alphaLow, ndf, ddf, ncent ) - quantile( 'F', alphaLow, ndf, ddf, ncent )

Output width
Stop
Program Name: sampleNormPop2.sas

* Program Description

sampleNormPop2.sas generates a random sample of size 'numObs' with mean popMean and population standard deviation popSD from a seed value in randSeed.

The random sample is saved to the dataset sample# where '#' is identifies from which population the random sample was drawn.

* Variables Defined

popMean := population mean
popSD := population standard deviation
seedIt := seed position
numObs := sample size
sampleNum := indicates sample identification for a simulation
seed := dataset of random seed

* Datasets Defined

sample&sampleNum := random sample dataset
Program Name: avgMedMultipleSim2.sas

* Program Description
avgMedMultipleSim2 manages the consolidation of the dataset multipleSimOut.
Calculations for each iteration block are stored in a dataset called avgMedMultipleSimOut.

* Flowchart page 16
avgMedSim2 calls the macro avgMedSim2.sas
avgMedSim2.sas averages over the rows of multipleSimOut by the user defined number of iterations. avgMedSim2 also determines the median of the calculated power values of the rows.

* Loop through the dataset multipleSimOut by user defined iteration blocks.

Start

Input tM, nsXnss

bStart = 1
bFinish = tM

i = 1

i > nsXnss?

no

avgMedSim2

bStart = bStart + tM
bFinish = bFinish + tM
i = i + 1

yes

j = 1

j = j + 1

no

j > nsXnss?

Output dataset avgMedMultipleSimOut

yes

A

Stop

A

Appendix A: Two-Sample Flow Chart of Power Simulations
Appendix A: Two-Sample Flow Chart of Power Simulations

* Variables Defined

iM := maximum number of iterations
nsXms := number of population standard deviations considered multiplied by the number of sample sizes considered
bStart := start of seed block
bFinish := end of seed block

* Datasets Defined

avgMedMultipleSimOut := averaged dataset of multipleSimOut by iteration blocks including the median for the calculated power values in each iteration block
Program Name: avgMedSim2.sas

* Program Description
avgMedSim2.sas averages over the rows of multipleSimOut by the user defined number of iterations. avgMedSim2 also determines the median of the calculated power values of the rows.

Start

Input start, finish, iB, X

rows = X[ start:finish, ]
avgRows = rows[ :, ]
col = X[ start:finish, 13 ]
numRows = nrow( col )
mis = 0

i = 1

missing( ( col[ i, 1 ] ) ) < 1 ?

yes

i = i + 1

no

mis = mis // i

i <= numRows ?

Page 18
* Remove missing values if present

nr = nrow(mis)

nr < 2 ?

no → mis = remove(mis, 1)

yes

nr < 2 ?

no → col = col`

yes

nr < 2 ?

no → col = remove(col, mis)

yes

nr < 2 ?

no → col = col`

yes

med = median(col)

avgRows = avgRows || med

Output dataset amSim&itB

Stop
* rand3.csv is a three column file of random numbers generated from \url{http://www.random.org}. For each iteration three random values are used to seed three random samples.

* Datasets Defined

randSeed := dataset of random seeds

* Program Description

threeSample.sas acts as a controller interface which accepts user defined values for three population means, upper and lower bounds for a range of population standard deviations, a maximum sample size, a Type I error rate, and the number of iterations per sample size to population standard deviation combination.

Program Name: threeSample.sas
Appendix A: Three-Sample Flow Chart of Power Simulations

Program Name: multipleSim3.sas

* Program Description
multipleSim3.sas manages the production of simulations.

* Loop through sample sizes 5 to 14 by 1.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Loop through sample sizes 15 to the maximum user defined sample size by 5.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

Program Description
multipleSim3.sas manages the production of simulations.

* Loop through sample sizes 5 to 14 by 1.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Loop through sample sizes 15 to the maximum user defined sample size by 5.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

Flowchart page 5
simulation3 calls the macro simulation3.sas
This macro performs calculations based on Thomas 1997

Program Description
multipleSim3.sas manages the production of simulations.

* Loop through sample sizes 5 to 14 by 1.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Loop through sample sizes 15 to the maximum user defined sample size by 5.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

Flowchart page 5
simulation3 calls the macro simulation3.sas
This macro performs calculations based on Thomas 1997

Program Description
multipleSim3.sas manages the production of simulations.

* Loop through sample sizes 5 to 14 by 1.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Loop through sample sizes 15 to the maximum user defined sample size by 5.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

Flowchart page 5
simulation3 calls the macro simulation3.sas
This macro performs calculations based on Thomas 1997
* Loop through population standard deviation 20 to user defined upper population standard deviation by 20.

sd = std / 2

A2

std = std + 20

no

std > sigmaMax ?

C2

yes

Page 4

* Loop through sample sizes 5 to 14 by 1.

* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

nn = 5

it = block

nn > 14 ?

no

it > block + itMax - 1 ?

yes

simulation3

count = count + 1

it = it + 1

B2

yes

Page 5

* Flowchart page 5

* simulation3 calls the macro simulation3.sas

* This macro performs calculations based on Thomas 1997

* Loop through sample sizes 15 to the maximum user defined sample size by 5.

* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

nn = 15

it = block

nn > sampleSizeMax ?

no

it > block + itMax - 1 ?

yes

simulation3

count = count + 1

it = it + 1

B2

yes

C2

Page 3
* Variables Defined:
  
  \( \mu_1 \) := mean for population one  
  \( \mu_2 \) := mean for population two  
  \( \mu_3 \) := mean for population three  
  \( \sigma_{\text{min}} \) := minimum population standard deviation  
  \( \sigma_{\text{max}} \) := maximum population standard deviation  
  \( \text{itMax} \) := number of iterations  
  \( \text{sampleSizeMax} \) := maximum sample size  
  \( \text{sigLevel} \) := Type I error rate  
  \( \text{count} \) := simulation identification  
  \( \text{block} \) := sets location of random seeds for simulation5k.sas

* Datasets Defined:
  
  \( \text{multipleSimOut} \) := dataset of produced simulations

* Gather simulations into a dataset

\( k = 1 \)

\( k > \text{count} - 1 \) ?

\( k = k + 1 \)

\( \text{sim}&k \)

\( \text{Output dataset multipleSimOut} \)

Stop
Program Name: simulation3.sas

* Program Description

simulation3.sas performs a hypothesis test of no difference for three population means. Random samples are normally distributed with respective means $\mu_A$, $\mu_B$, and $\mu_C$ and equal population standard deviation $\sigma$. Random samples are generated using seed values from the dataset randSeed. Power calculations are based on methods outlined in Thomas 1997. Calculations are saved to a dataset named 'sim#' where # represents a simulation count.

Sample Code:

```sas
* Merge datasets sample1, sample2, and sample3 to produce dataset sampleThreePops
* X holds values from the dataset sampleThreePops
```

```sas
eps = 1e-4
n = nrow(X)
p = ncol(X)
J = J(p, 1)
muG = (\mu_A + \mu_B + \mu_C) / p
xBar = X(:, 1:end)' / n
sst = n * xBar * xBar - (n / p) * (xBar * J)' * 2
df1 = p - 1
mst = sst / df1
sse = ssq(X) - (xBar * xBar)' * n
df2 = p * (n - 1)
mse = sse / df2
fSamp = mst / mse
```

Appendix A: Three-Sample Flow Chart of Power Simulations
trueLambda = ( n * ( ( ( &muA - muG )**2 ) + ( ( &muB - muG )**2 ) + ( ( &muC - &muG )**2 ) ) ) / ( ( &sigma )**2 )

truePower = 1 - cdf( 'F', fCrit, df1, df2, trueLambda )
estLambda = fSamp * ( p - 1 )
adjEstLambda = ( estLambda * ( df2 - 2 ) / df2 ) - df1

* Estimated lambda and adjusted estimated lambda by Thomas 1997

* Estimated lambda and adjusted estimated lambda by Thomas 1997

adjEstLambda > 0 ?

For some random seed pairs, adjEstLambda is negatively noncentrality parameter occurs when the expression '( estLambda * (df2 - 2) / df2 ) - df1' is less than df1

alphaLU takes on a value from alphaLU

* alphaLU takes on a value from alphaLU

* Flowchart page 10

* Suggested by Dr. Leigh Murray (2009) and possibly implied by Thomas 1997.

* Flowchart page 10

* Suggested by Dr. Leigh Murray (2009) and possibly implied by Thomas 1997.

* For some random seed pairs, adjEstLambda is negatively noncentrality parameter occurs when the expression '( estLambda * (df2 - 2) / df2 ) - df1' is less than df1

* alphaLU takes on a value from alphaLU

* alphaLU takes on a value from alphaLU

* Flowchart page 10

* Suggested by Dr. Leigh Murray (2009) and possibly implied by Thomas 1997.

* Flowchart page 10

* Suggested by Dr. Leigh Murray (2009) and possibly implied by Thomas 1997.
* For fixed observed F statistic, numerator degrees of freedom, and denominator degrees of freedom, fnonct reports an error when the probability supplied is greater than the probability from the cdf of a central F distribution ~ SAS Functions & Call Routines

\[
\alpha_U = \alpha - \alpha_L \\
\text{adj} \alpha_L = 1 - \alpha_L \\
\text{probCenF} = \text{cdf}(\ 'F', \ fSamp, \ df1, \ df2 )
\]

\[\text{adj} \alpha_L > \text{probCenF} ?
\]

\[\lambda_L = 0 \text{ yes} \]

\[\text{no} \]

\[\lambda_L = \text{fnonct}(fSamp, \ df1, \ df2, \text{adj} \alpha_L ) \]

\[\text{If the adjusted lower rejection region is greater than the probability from the cdf of a central F distribution then set the noncentrality parameter to zero} \]

\[\text{* adj} \alpha_L \text{ is occasionally problematic} \]

\[\text{Page 6} \]

\[\lambda_L = 0 \]

\[\text{no} \]

\[\alpha_U > \text{probCenF} ? \]

\[\text{yes} \]

\[\lambda_U = \text{fnonct}(fSamp, \ df1, \ df2, \text{adj} \alpha_L ) \]

\[\text{If the upper rejection region is greater than the probability from the cdf of a central F distribution then set the noncentrality parameter to zero} \]

\[\text{Page 8} \]

\[\lambda_U = 0 \]

\[\text{no} \]

\[\alpha_U > \text{probCenF} ? \]

\[\text{yes} \]
Appendix A: Three-Sample Flow Chart of Power Simulations

\[ \text{estPower} = 1 - \text{cdf}(\text{'F'}, \text{fCrit}, \text{df1}, \text{df2}, \text{adjEstLambda}) \]

\[ \text{powerL} = 1 - \text{cdf}(\text{'F'}, \text{fCrit}, \text{df1}, \text{df2}, \text{lambdaL}) \]

\[ \text{powerU} = 1 - \text{cdf}(\text{'F'}, \text{fCrit}, \text{df1}, \text{df2}, \text{lambdaU}) \]

\[ Y[1] = \mu_A \]

\[ Y[2] = \mu_B \]

\[ Y[3] = \mu_C \]

\[ Y[4] = \sigma \]

\[ Y[5] = \text{sampleSize} \]

\[ Y[6] = \alpha \]

\[ Y[7] = \text{pValue} \]

\[ Y[8] = \text{trueLambda} \]

\[ Y[9] = \text{truePower} \]

\[ Y[10] = \text{estLambda} \]

\[ Y[11] = \text{adjEstLambda} \]

\[ Y[12] = \lambda_L \]

\[ Y[13] = \lambda_U \]

\[ Y[14] = \text{estPower} \]

\[ Y[15] = \text{powerL} \]

\[ Y[16] = \text{powerU} \]

Output dataset: sim&simNum

Stop
Appendix A: Three-Sample Flow Chart of Power Simulations

* Variables Defined

\[ \mu_A := \text{mean for population one} \]
\[ \mu_B := \text{mean for population two} \]
\[ \mu_C := \text{mean for population three} \]
\[ \mu_G := \text{average of the tree population means} \]
\[ \sigma := \text{population standard deviation} \]
\[ \text{seedNum} := \text{location of random seeds} \]
\[ \text{sampleSize} := \text{sample size} \]
\[ \alpha := \text{level of significance} \]
\[ \text{simNum} := \text{simulation identification} \]
\[ X := \text{samples from normal populations} \]
\[ \epsilon := \text{tolerance for Golden Section Search methods} \]
\[ n := \text{number rows} \]
\[ p := \text{number columns} \]
\[ \bar{x} := \text{sample mean} \]
\[ \text{sst} := \text{sum of squares for treatment} \]
\[ \text{df1} := \text{numerator degrees of freedom} \]
\[ \text{mst} := \text{mean square for treatment} \]
\[ \text{sse} := \text{sum of squares for error} \]
\[ \text{df2} := \text{denominator degrees of freedom} \]
\[ \text{mse} := \text{mean square for error} \]
\[ f_{\text{Samp}} := \text{observed F statistic} \]
\[ f_{\text{Crit}} := \text{critical value from a central F distribution} \]
\[ p_{\text{Value}} := p\text{-value for the test} \]
\[ \text{trueLambda} := \text{noncentrality parameter} \]
\[ \text{truePower} := \text{power for the test} \]
\[ \text{estLambda} := \text{calculated noncentrality parameter} \]
\[ \text{adjEstLambda} := \text{adjusted calculated noncentrality parameter} \]
\[ \alpha_{L} := \text{optimized lower critical region} \]
\[ \alpha_{U} := \text{optimized upper critical region} \]
\[ \text{probCent} := \text{probability of observed F statistic of the central F distribution} \]
\[ \text{estPower} := \text{calculated power} \]
\[ \text{powerL} := \text{lower confidence interval for calculated power} \]
\[ \text{powerU} := \text{upper confidence interval for calculated power} \]

* Datasets Defined

\[ \text{sampleThreePops} := \text{random samples dataset} \]
\[ \text{sim&simNum} := \text{simulation dataset} \]
Start

Input ndf1, ddf2, ncp, slAlpha, tol

alphaA = 0
alphaB = slAlpha
c = (-1 + sqrt(5)) / 2
x1 = c * alphaA + (1 - c) * alphaB
wx1 = w(x1, ndf1, ddf2, ncp, slAlpha)
x2 = (1 - c) * alphaA + c * alphaB
wx2 = w(x2, ndf1, ddf2, ncp, slAlpha)

A

abs(alphaB - alphaA) > tol?

yes

Output (alphaA + alphaB) / 2

no

wx1 < wx2?

yes

alphaB = x2
wx2 = wx1
x2 = (1 - c) * alphaA + c * alphaB
x1 = c * alphaA + (1 - c) * alphaB
wx1 = w(x1, ndf1, ddf2, ncp, slAlpha)

Input wx1

Input wx2

A

Flowchart page 12

w returns the confidence interval width of a noncentral f-distribution

wx2 takes on a value from w

wx2 takes on a value from w

Program Description

The function alphaLU finds the minimum confidence interval width of a noncentral f-distribution using an implementation of the Golden Section Search method by Kincaid and Cheney 2009, and Jan Verschelde http://www.math.uic.edu/~jan/mcs471/Lec9/gss.pdf

Program Name: alphaLU
Variables Defined

\( ndf1 := \) numerator degrees of freedom
\( ddf2 := \) denominator degrees of freedom
\( ncp := \) noncentrality parameter
\( slAlpha := \) level of significance
\( tol := \) tolerance for Golden Section Search method
\( alphaA := \) lower bound for optimized lower alpha
\( alphaB := \) upper bound for optimized lower alpha
\( c := \) Golden ratio constant reduction factor
\( x1 := \) percentile associated with alphaA
\( wx1 := \) height associated with x1
\( x2 := \) percentile associated with alphaB
\( wx2 := \) height associated with x2
Program Name: w

Start

Input alphaLow, ndf, ddf, ncent, sl

width = quantile('F', 1 - sl + alphaLow, ndf, ddf, ncent) - quantile('F', alphaLow, ndf, ddf, ncent)

Output width

Stop

* Program Description
The function w returns a confidence interval width of a noncentral F-distribution

* Variables Defined
alphaLow := lower critical region
ndf := numerator degrees of freedom
ddf := denominator degrees of freedom
ncent := noncentrality parameter
sl := level of significance
width := distance between percentiles
Program Name: sampleNormPop3.sas

* Program Description

SampleNormPop3.sas generates a random sample of size 'numObs' with mean popMean and population standard deviation popSD from a seed value in randSeed. The random sample is saved to the dataset sample# where '#' identifies from which population the random sample was drawn.

* Variables Defined

popMean := population mean
popSD := population standard deviation
seedIt := seed position
numObs := sample size
sampleNum := indicates sample identification for a simulation
seed := dataset of random seed

* Datasets Defined

sample&sampleNum := random sample dataset

* seed holds values from the dataset randSeed

x&sampleNum = popMean + popSD * rannor( J( numObs, 1, seed[ seedIt, sampleNum ] ) )

Start

Input popMean, popSD, seedIt, numObs, sampleNum, seed

Output dataset sample&sampleNum

Stop
Program Name: avgMedMultipleSim3.sas

* Program Description
avgMedMultipleSim3 manages the consolidation of the dataset multipleSimOut.
Calculations for each iteration block are stored in a dataset called avgMedMultipleSimOut.

* Flowchart page 16
* avgMedSim3 calls the macro avgMedSim3.sas
* avgMedSim3.sas averages over the rows of multipleSimOut by the user defined number of iterations. avgMedSim3 also determines the median of the calculated power values of the rows.

* Loop through the dataset multipleSimOut by user defined iteration blocks.

Start
Input itM, nsXnss
bStart = 1
bFinish = itM
i = 1
i > nsXnss?
no
avgMedSim2
bStart = bStart + itM
bFinish = bFinish + itM
i = i + 1

j = 1
j > nsXnss?
j = j + 1
no
yes
Output dataset avgMedMultipleSimOut

A

A

Stop
Appendix A: Three-Sample Flow Chart of Power Simulations

* Variables Defined

iM := maximum number of iterations
nsXnss := number of population standard deviations considered multiplied by
the number of sample sizes considered
bStart := start of seed block
bFinish := end of seed block

* Datasets Defined

avgMedMultipleSimOut := averaged dataset of multipleSimOut by iteration
blocks including the median for the calculated power values in each iteration
block
* X holds values from the dataset multipleSimOut

rows = X[start:finish,]
avgRows = rows[:,]
col = X[start:finish, 14]
numRows = nrow(col)
mis = 0

i = 1

missing( ( col[i, 1] ) ) < 1 ?

mis = mis // i

i = i + 1

i <= numRows ?

* check column vector for missing values

* Program Description
avgMedSim3.sas averages over the rows of multipleSimOut by the user defined number of iterations. avgMedSim3 also determines the median of the calculated power values of the rows.
* Remove missing values if present

nr = nrow(mis)

nr < 2 ?

yes

no

mis = remove(mis, 1)

* Variables Defined
start := beginning of seed block
finish := end of seed block
itB := label for iteration block
X := dataset multipleSimOut
rows := iteration block
avgRows := average over rows
col := column of calculated power
numRows := number of iterations
mis := stores missing value positions
nr := number of missing values
med := median calculated power

* Datasets Defined
amSim&itB := averaged dataset from multipleSimOut for a single iteration block including the median of the calculated power values

* median finds the median for a column vector of non-missing values

med = median(col)

avgRows = avgRows || med

Output dataset amSim&itB

Stop
Appendix A: Four-Sample Flow Chart of Power Simulations

* Datasets Defined

randSeed := dataset of random seeds

* Program Description

fourSample.sas acts as a controller interface which accepts user defined values for four population means, upper and lower bounds for a range of population standard deviations, a maximum sample size, a Type I error rate, and the number of iterations per sample size to population standard deviation combination.
Input $\mu_1$, $\mu_2$, $\mu_3$, $\mu_4$, $\sigma_{\text{Min}}$, $\sigma_{\text{Max}}$, $\text{itMax}$, $\text{sampleSizeMax}$, $\text{sigLevel}$

Block = 1
Count = 1
$\text{std} = \sigma_{\text{Min}}$
$\text{sd} = \text{std} / 2$

If $\text{std} > 19$?
  - No
  - $\text{std} = \text{std} + 1$

A1
C1

If $\text{std} > 19$?
  - Yes
  - Page 3

* Program Description
multipleSim4.sas manages the production of simulations.

* Loop through user defined lower population standard deviation to 19 by 1.

* Flowchart page 5
* simulation4 calls the macro simulation4.sas
* This macro performs calculations based on Thomas 1997

* Loop through sample sizes 5 to 14 by 1.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

$\text{nn} = 5$
$\text{it} = \text{block}$
$\text{count} = \text{count} + 1$
$\text{it} = \text{it} + 1$

If $\text{it} > \text{block} + \text{itMax} - 1$?
  - No
  - $\text{it} = \text{block}$
  - $\text{it} = \text{it} + 1$

If $\text{nn} > 14$?
  - Yes
  - $\text{block} = \text{it}$
  - $\text{nn} = \text{nn} + 1$

If $\text{nn} > \text{sampleSizeMax}$?
  - Yes

B1
C1

* Loop through sample sizes 15 to the maximum user defined sample size by 5.
* Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

$\text{nn} = 15$
$\text{it} = \text{block}$
$\text{count} = \text{count} + 1$
$\text{it} = \text{it} + 1$

If $\text{it} > \text{block} + \text{itMax} - 1$?
  - No
  - $\text{it} = \text{block}$
  - $\text{it} = \text{it} + 1$

If $\text{nn} > \text{sampleSizeMax}$?
  - Yes
  - $\text{block} = \text{it}$
  - $\text{nn} = \text{nn} + 5$
* Loop through population standard deviation 20 to user defined upper population standard deviation by 20.

* Loop through sample sizes 5 to 14 by 1.
  * Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Loop through sample sizes 15 to the maximum user defined sample size by 5.
  * Subloop through a user defined number of iterations by 1. When subloop quits, increase subloop starting position by number of iterations. Iteration count is maintained to locate and traverse the position of random seed values in the dataset randSeed.

* Flowchart page 5
  * simulation4 calls the macro simulation4.sas
  * This macro performs calculations based on Thomas 1997

Appendix A: Four-Sample Flow Chart of Power Simulations
Appendix A: Four-Sample Flow Chart of Power Simulations

* Gather simulations into a dataset

* Datasets Defined
  multipleSimOut := dataset of produced simulations

* Variables Defined:
  mu1 := mean for population one
  mu2 := mean for population two
  mu3 := mean for population three
  mu4 := mean for population four
  sigmaMin := minimum population standard deviation
  sigmaMax := maximum population standard deviation
  itMax := number of iterations
  sampleSizeMax := maximum sample size
  sigLevel := Type I error rate
  count := simulation identification
  block := sets location of random seeds for simulation5k.sas

Output dataset multipleSimOut
Stop
Program Name: simulation4.sas

Program Description

simulation4.sas performs a hypothesis test of no difference for four population means. Random samples are normally distributed with respective means $\mu_A, \mu_B, \mu_C, \mu_D$ and equal population standard deviation $\sigma$. Random samples are generated using seed values from the dataset randSeed. Power calculations are based on methods outlined in Thomas 1997. Calculations are saved to a dataset named 'sim#' where # represents a simulation count.

* Merge datasets sample1, sample2, sample3 and sample4 to produce dataset sampleFourPops

* X holds values from the dataset sampleFourPops

* Flowchart page 13

* sampleNormPop4 calls the macro sampleNormPop4.sas

* This macro generates random samples of size 'sampleSize' with mean $\mu_{letter}$ and population standard deviation $\sigma$ from a seed value in the dataset randSeed.

* Produces dataset sample#
$f \text{Crit} = \text{quantile}(F, 1 - \alpha, df1, df2)$

$p \text{Value} = 1 - \text{cdf}(F, f \text{Samp}, df1, df2)$

$true \Lambda = \frac{n \times ((\mu_A - \mu_G)^2 + (\mu_B - \mu_G)^2 + (\mu_C - \mu_G)^2 + (\mu_D - \mu_G)^2))}{(\sigma)^2}$

$truePower = 1 - \text{cdf}(F, f \text{Crit}, df1, df2, true \Lambda)$

$est \Lambda = f \text{Samp} \times (p - 1)$

$adjEst \Lambda = \frac{(est \Lambda \times (df2 - 2))}{df2} - df1$

For some random seed pairs, $adjEst \Lambda$ is negative. A negative adjusted estimated noncentrality parameter occurs when the expression $\frac{(est \Lambda \times (df2 - 2))}{df2}$ is less than $df1$.

$adjEst \Lambda > 0 \ ?$

$\alpha \text{LU}$

* $\alpha \text{LU}$ takes on a value from $\alpha \text{LU}$

Input $\alpha \text{LU}$

* Flowchart page 10

* Minimize lower rejection region by minimizing confidence interval width.

* Suggested by Dr. Leigh Murray (2009) and possibly implied by Thomas 1997.
alphaU = alpha - alphaL
adjAlphaL = 1 - alphaL
probCenF = cdf('F', fSamp, df1, df2)

* For fixed observed F statistic, numerator degrees of freedom, and denominator degrees of freedom, fnonct reports an error when the probability supplied is greater than the probability from the cdf of a central F distribution.

* As the probability supplied approaches the probability from a central F distribution, the noncentrality parameter tends toward 0.

lambdaL = fnonct(fSamp, df1, df2, adjAlphaL)

* If the adjusted lower rejection region is greater than the probability from the cdf of a central F distribution then set the noncentrality parameter to zero.

If adjAlphaL is occasionally problematic

lambdaU = fnonct(fSamp, df1, df2, adjAlphaL)

* If the upper rejection region is greater than the probability from the cdf of a central F distribution then set the noncentrality parameter to zero.

* For fixed observed F statistic, numerator degrees of freedom, and denominator degrees of freedom, fnonct reports an error when the probability supplied is greater than the probability from the cdf of a central F distribution.

* SAS Functions & Call Routines

* If the adjusted lower rejection region is greater than the probability from the cdf of a central F distribution then set the noncentrality parameter to zero.

* adjAlphaL is occasionally problematic.

* If the adjusted lower rejection region is greater than the probability from the cdf of a central F distribution then set the noncentrality parameter to zero.

Appendix A: Four-Sample Flow Chart of Power Simulations
estPower = 1 - cdf('F', fCrit, df1, df2, adjEstLambda)
powerL = 1 - cdf('F', fCrit, df1, df2, lambdaL)
powerU = 1 - cdf('F', fCrit, df1, df2, lambdaU)

\[
\begin{align*}
Y[1] &= \mu_A \\
Y[2] &= \mu_B \\
Y[3] &= \mu_C \\
Y[4] &= \mu_D \\
Y[5] &= \sigma \\
Y[6] &= \text{sampleSize} \\
Y[7] &= \alpha \\
Y[8] &= pValue \\
Y[9] &= \text{trueLambda} \\
Y[10] &= \text{truePower} \\
Y[11] &= \text{estLambda} \\
Y[12] &= \text{adjEstLambda} \\
Y[13] &= \lambda_L \\
Y[14] &= \lambda_U \\
Y[15] &= \text{estPower} \\
Y[16] &= \text{powerL} \\
Y[17] &= \text{powerU}
\end{align*}
\]

Output dataset sim&simNum

Stop
Variables Defined

\- \( \mu_A \) := mean for population one
\- \( \mu_B \) := mean for population two
\- \( \mu_C \) := mean for population three
\- \( \mu_D \) := mean for population four
\- \( \mu_G \) := average of the tree population means
\- \( \sigma \) := population standard deviation
\- \( \text{seedNum} \) := location of random seeds
\- \( \text{sampleSize} \) := sample size
\- \( \alpha \) := level of significance
\- \( \simNum \) := simulation identification
\- \( X \) := samples from normal populations
\- \( \epsilon \) := tolerance for Golden Section Search methods
\- \( n \) := number rows
\- \( p \) := number columns
\- \( \bar{x} \) := sample mean
\- \( sst \) := sum of squares for treatment
\- \( df_1 \) := numerator degrees of freedom
\- \( mst \) := mean square for treatment
\- \( sse \) := sum of squares for error
\- \( df_2 \) := denominator degrees of freedom
\- \( mse \) := mean square for error
\- \( f\text{Samp} \) := observed F statistic
\- \( f\text{Crit} \) := critical value from a central F distribution
\- \( p\text{Value} \) := p-value for the test
\- \( \text{trueLambda} \) := noncentrality parameter
\- \( \text{truePower} \) := power for the test
\- \( \text{estLambda} \) := calculated noncentrality parameter
\- \( \text{adjEstLambda} \) := adjusted calculated noncentrality parameter
\- \( \alpha_L \) := optimized lower critical region
\- \( \alpha_U \) := optimized upper critical region
\- \( \text{probCentF} \) := probability of observed F statistic of the central F distribution
\- \( \text{estPower} \) := calculated power
\- \( \text{powL} \) := lower confidence interval for calculated power
\- \( \text{powU} \) := upper confidence interval for calculated power

Datasets Defined

\- \( \text{sampleFourPops} \) := random samples dataset
\- \( \text{sim&simNum} \) := simulation dataset
Start

Input ndf1, ddf2, ncp, slAlpha, tol

alphaA = 0
alphaB = slAlpha
c = (-1 + sqrt(5)) / 2
x1 = c * alphaA + (1 - c) * alphaB
wx1 = w(x1, ndf1, ddf2, ncp, slAlpha)
x2 = (1 - c) * alphaA + c * alphaB
wx2 = w(x2, ndf1, ddf2, ncp, slAlpha)

alphaA = x1
x1 = x2
wx1 = wx2
x2 = (1 - c) * alphaA + c * alphaB

abs(alphaB - alphaA) > tol?
yes

alphaB = x2
x2 = x1
wx2 = wx1

x1 = c * alphaA + (1 - c) * alphaB

wx1 < wx2?
no

Output (alphaA + alphaB) / 2

Program Name: alphaLU

* Program Description

* Flowchart page 12
* w returns the confidence interval width of a noncentral F-distribution

* wx2 takes on a value from w

* wx2 takes on a value from w
Variables Defined

\[ \text{ndf1 := numerator degrees of freedom} \]
\[ \text{ddf2 := denominator degrees of freedom} \]
\[ \text{ncp := noncentrality parameter} \]
\[ \text{slAlpha := level of significance} \]
\[ \text{tol := tolerance for Golden Section Search method} \]
\[ \text{alphaA := lower bound for optimized lower alpha} \]
\[ \text{alphaB := upper bound for optimized lower alpha} \]
\[ \text{c := Golden ratio constant reduction factor} \]
\[ \text{x1 := percentile associated with alphaA} \]
\[ \text{wx1 := height associated with x1} \]
\[ \text{x2 := percentile associated with alphaB} \]
\[ \text{wx2 := height associated with x2} \]
Program Description

The function \( w \) returns a confidence interval width of a noncentral \( F \)-distribution.

Variables Defined

- \( \alpha_{\text{Low}} \): lower critical region
- \( ndf \): numerator degrees of freedom
- \( ddf \): denominator degrees of freedom
- \( ncent \): noncentrality parameter
- \( sl \): level of significance
- \( width \): distance between percentiles
Program Name: sampleNormPop4.sas

* Program Description

sampleNormPop4.sas generates a random sample of size 'numObs' with mean popMean and population standard deviation popSD from a seed value in randSeed.

The random sample is saved to the dataset sample# where '#' is identifies from which population the random sample was drawn.

* Variables Defined

  popMean := population mean
  popSD := population standard deviation
  seedIt := seed position
  numObs := sample size
  sampleNum := sample identification for a simulation
  seed := dataset of random seed

* Datasets Defined

  sample&sampleNum := random sample dataset
**Program Description**

`avgMedMultipleSim4` manages the consolidation of the dataset `multipleSimOut`. Calculations for each iteration block are stored in a dataset called `avgMedMultipleSimOut`.

**Flowchart page 16**

- `avgMedSim4` calls the macro `avgMedSim4.sas`
- `avgMedSim4.sas` averages over the rows of `multipleSimOut` by the user-defined number of iterations, `avgMedSim4` also determines the median of the calculated power values of the rows.

---

* Loop through the dataset `multipleSimOut` by user-defined iteration blocks.

Start

Input itM, nsXnss

b\(\text{Start} = 1\)  
b\(\text{Finish} = \text{itM}\)

i = 1

i > nsXnss?

yes

**Gather condensed simulations into the dataset**

`avgMedMultipleSimOut`

no

j = 1

j > nsXnss?

no

i = i + 1

Output dataset `avgMedMultipleSimOut`

yes

A

Stop

**Program Name:** `avgMedMultipleSim4.sas`
Appendix A: Four-Sample Flow Chart of Power Simulations

* Variables Defined

i#M := maximum number of iterations
nsXnss := number of population standard deviations considered multiplied by
the number of sample sizes considered
bStart := start of seed block
bFinish := end of seed block

* Datasets Defined

avgMedMultipleSimOut := averaged dataset of multipleSimOut by iteration
blocks including the median for the calculated power values in each iteration
block
* X holds values from the dataset multipleSimOut

* check column vector for missing values

* Program Description
avgMedSim4.sas averages over the rows of multipleSimOut by the user defined number of iterations. avgMedSim4 also determines the median of the calculated power values of the rows.

```
start
input start, finish, itB, X;
rows = X[start:finish, ];
avgRows = rows[:, ];
col = X[start:finish, 15 ];
numRows = nrow(col);
mis = 0
i = 1
i <= numRows?
no
yes
missing( (col[i, 1]) ) < 1 ?
no
mis = mis // i
yes
i = i + 1
i <= numRows ?
no
Page 18
```
Appendix A: Four-Sample Flow Chart of Power Simulations

* Variables Defined
- start := beginning of seed block
- finish := end of seed block
- itB := label for iteration block
- X := dataset multipleSimOut
- rows := iteration block
- avgRows := average over rows
- col := column of calculated power
- numRows := number of iterations
- mis := stores missing value positions
- nr := number of missing values
- med := median calculated power

* Datasets Defined
- amSim&itB := averaged dataset from multipleSimOut for a single iteration block including the median of the calculated power values

* median finds the median for a column vector of non-missing values
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
/* twoSample **************************************************************/
/* Requirement(s):  */
/* rand2.csv, multipleSim2.sas, avgMedMultipleSim2.sas  */
/* */
/* Program Description: */
/* twoSample.sas acts as a controller interface which accepts user */
/* defined values for two population means, upper and lower bounds */
/* for a range of population standard deviations, a maximum sample size, */
/* a type I error rate, and the number of iterations per sample size */
/* to population standard deviation combination. */
/* */
/***************************/

proc printto print='\Path\to\Output\Folder\twoSampleOut.lst' log='\Path\to\Output\Folder\twoSampleOut.log';
run;

/* Input random seed file to the dataset randseed ***************/
/* */
/* rand2.csv is a two column file of random numbers generated from */
/* http://www.random.org. For each iteration of multipleSim2 two random */
/* values are used to seed two random samples. */
/* */
/***************************/
data randSeed;
  infile '\Path\to\File\rand2.csv' dlm=',';
  input seedCol1 seedCol2;
run;

/* Produce simulations **********************************************/
/* */
/* The multipleSim2 manages the production of simulations. */
/* Calls are made to the macro simulation2.sas which performs */
/* calculations based on Thomas 1997. */
/* */
/* Calculations for each simulation are stored in a dataset called */
/* multipleSimOut. */
/* */
/***************************/
%multipleSim2(mu1=30,mu2=40,sigmaMin=5,sigmaMax=40,itMax=100,sampleSizeMax=50,sigLevel=0.05);
```
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
/* Print multipleSimOut */
proc print data=multipleSimOut;
run;
/* Produce averages and medians for simulations ***********************/
/* The avgMedMultipleSim2 manages the consolidation of simulations. */
/* Calls are made to avgMedSim2 for each sigma by sample size */
/* combination. avgMedSim2 averages over a user specified number */
/* of rows from multipleSimOut. avgMedSim2 also determines the */
/* median of the calculated power values of the rows. */
/* Calculations for each iteration block are stored in a dataset */
/* called avgMedMultipleSimOut. */
/**********************************************************************/
%avgMedMultipleSim2(itM=100,nsXnss=306);
/* Print avgMedMultipleSimOut */
proc print data=avgMedMultipleSimOut;
run;
/* Plot avgMedMultipleSimOut */
goptions ftitle=swiss ftext=swiss;
symbol1 value=square interpol=join width=0.5 color=blue;
symbol2 value=star interpol=join width=0.5 color=red;
symbol3 value=triangle interpol=join width=0.5 color=yellow;
symbol4 value='x' interpol=join width=0.5 color=orange;
symbol5 value=circle interpol=join width=0.5 color=blue;
symbol6 value=dot interpol=join width=0.5 color=green;
title2 height=1.4 "Sample Size vs Power";
axis1 order=(0.0 to 1.0 by 0.1) label=(angle=90 'Power') minor=none;
axis2 order=(5 to 50 by 5) label=(angle=0 'Sample Size');
legend label=none value=(h=.8 'Upper AVG Estimated Power' 'True Power' 'AVG Estimated Power' 'Median Power' 'Lower AVG Estimated Power' 'AVG P-Value')
    position=center;
```

Appendix B: Two-Sample SAS Code for Power Simulations
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
proc gplot data=avgMedMultipleSimOut;
  plot powerU*sampleSize truePower*sampleSize estPower*sampleSize medPower*sampleSize powerL*sampleSize
  pValue*sampleSize/overlay vaxis=axis1 haxis=axis2 hminor=4 legend=legend1;
  by sigma;
run;
```
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
/* multipleSim2 ***************************************************************/
/* */
/* Requirement(s): */
/* simulation2.sas */
/* */
/* Program Description: */
/* multipleSim2 iterates simulation2.sas and combines simulation output to a */
/* dataset. */
/* */
/* Input: */
/* mu1 := population one mean */
/* mu2 := population two mean */
/* sigmaMin := minimum population standard deviation controller */
/* sigmaMax := maximum population standard deviation controller */
/* itMax := number of iterations */
/* sampleSizeMax := largest sample size considered */
/* sigLevel := level of significance */
/* */
/* Note(s): */
/* In the subsubloop iterations run in blocks. This is done to */
/* traverse the dataset randSeed, where by each call to the macro */
/* simulation2 gets a different random seed position. */
/* */
/* This macro contains two main loops. The first main loop */
/* iterates from the minimum user define population standard deviation to 19 */
/* by 1. The second main loop iterates from 20 to the maximum user defined */
/* population standard deviation by 20. Each main loop has two subloops. */
/* The first subloop iterates from a system defined minimum sample size of 5 */
/* to 14 by 1. The second subloop iterates from 15 to the maximum user */
/* defined sample size by 5. Each subloop has a subsubloop. */
/* Each subsubloop iterates by a use defined number of iterations. After */
/* a subsubloop executes, the starting position of an iteration 'block' is */
/* increased by the the user defined number of iterations. */
/* */
/*******************************************************/

%macro multipleSim2(mu1=,mu2=,sigmaMin=,sigmaMax=,itMax=,sampleSizeMax=,sigLevel=);
%
%local std sd nn it block s n i b;
%let block = 1;
%let b = 1;
%let count = 1;
```
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
/* Produce simulations */

%do std = &sigmaMin %to 19 %by 1;
  %let sd = %sysevalf(&std/2);
  %do nn = 5 %to 14 %by 1;
    %do it = &block %to &block+&itMax-1 %by 1;
      %simulation2(muA=&mu1,muB=&mu2,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
      %let count = %eval(&count+1);
      %end;
    %let block = &it;
  %end;
%end;

%do nn = 15 %to &sampleSizeMax %by 5;
  %do it = &block %to &block+&itMax-1 %by 1;
    %simulation2(muA=&mu1,muB=&mu2,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
    %let count = %eval(&count+1);
    %end;
  %let block = &it;
%end;
%end;

%do std = 20 %to &sigmaMax %by 20;
  %let sd = %sysevalf(&std/2);
  %do nn = 5 %to 14 %by 1;
    %do it = &block %to &block+&itMax-1 %by 1;
      %simulation2(muA=&mu1,muB=&mu2,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
      %let count = %eval(&count+1);
      %end;
    %let block = &it;
  %end;
%end;

%do nn = 15 %to &sampleSizeMax %by 5;
  %do it = &block %to &block+&itMax-1 %by 1;
    %simulation2(muA=&mu1,muB=&mu2,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
    %let count = %eval(&count+1);
    %end;
  %let block = &it;
%end;
%end;
```

Appendix B: Two-Sample SAS Code for Power Simulations

/* Combine simulations into a dataset and name columns */
data multipleSimOut;
set
  %do k = 1 %to &count-1 %by 1;
  sim&k
%end;
; /* Double do-loop to reference datasets. ';' closes 'set' */
rename col1=mu1 col2=mu2 col3=sigma col4=sampleSize col5=sigLevel col6=pValue col7=trueLambda col8=truePower
  col9=estLambda col10=adjEstLambda col11=lambdaL col12=lambdaU col13=estPower col14=powerL col15=powerU;
run;
%mend multipleSim2;
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
/* simulation2 ***********************************************************/
/* Requirement(s): */
/* Program Description: */
/* simulation2.sas performs a hypothesis */
/* test of no difference for two population means. Random samples */
/* are normally distributed with respective means muA and muB with equal */
/* population standard deviation sigma. Random samples are generated */
/* are based on methods outlined in Thomas 1997. Calculations are */
/* saved to a dataset named 'sim#' where # represents the */
/* simulation count. */
/* Input: */
/* muA := mean of population one */
/* muB := mean of population two */
/* sigma := standard deviation of populations one and two */
/* seedNum := position for random seed */
/* sampleSize := sample size taken for populations one and two */
/* alpha := level of significance */
/* simNum := simulation ID */
/* Note(s): */
/* Optimal lower significance level by Golden Section Search method */
/*****************************************************************************/
%MACRO simulation2(muA=,muB=,sigma=,seedNum=,sampleSize=,alpha=,simNum=);
  /* Generate random samples from two normal populations */
  /* Merge datasets sample1 and sample2 to produce dataset sampleTwoPops */
  data sampleTwoPops;
    merge sample1 sample2;
  run;
```
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
/* Main */
proc iml;
reset nolog;
use sampleTwoPops;
read all into X;

/* Function ***************************************************/
start w(alphaLow,ndf,ddf,ncent,sl);
width = quantile('F',1-sl+alphaLow,ndf,ddf,ncent)-quantile('F',alphaLow,ndf,ddf,ncent);
return(width);
finish w;

/* Function: optimal lower critical region */
/* Finds the minimum confidence interval width using an */
/* implementation of the Golden Section Search method */
/* similar to Press et al. (2007). */
/* Note(s): */
/* The Golden Section Search method finds a value which minimizes */
/* a function. The function must be one dimensional and unimodal. */
/* Imagine a 'typical' pdf of an F distribution. Say we want to */
/* setup an interval such that the area covered by the sum of the two */
/* tails is alpha. For each alpha we can find the */
/* associated quantile using the SAS function 'quantile'. */
/* Since alpha is specified, knowing the lower area easily leads to */
/* the upper area. Here we consider only the lower */
/* area. As such we can relate interval width as a function of */
/* lower alpha size. As we increase or decrease the size of the */
/* lower area we increase or decrease the width of the interval. */
/* Assuming a minimum interval width exists, we optimize over the */
/* values [0.00, 0.05]. */
```
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
start alphaLU(ndf1,ddf2,ncp,slAlpha,tol);

alphaA = 0; /* Initial lower bracket on optimal region */
alphaB = slAlpha; /* Initial upper bracket on optimal region */

/* Note 1 *******************************************/
/* For each iteration an interval */
/* [alphaA, alphaB] is retained */
/* */

phi = (-1+sqrt(5))/2; /* Golden ratio constant reduction factor */
a1 = alphaA+phi*(alphaB-alphaA); /* Set test bracket a1 */
w1 = w(a1,ndf1,ddf2,ncp,slAlpha); /* Calculate Interval width for a1 */
a2 = alphaA+phi*phi*(alphaB-alphaA); /* Set test bracket a2 */
w2 = w(a2,ndf1,ddf2,ncp,slAlpha); /* Calculate interval width for a2 */

/* Note 2 *******************************************/
/* The initial setup invokes two function */
/* calls. The following loop uses one */
/* function call per iteration. */
/* */

do while (abs(alphaB-alphaA)>tol); /* Note 3 *******************************************/
/* We continue until the absolute difference */
/* between an upper and a lower bracket is */
/* negligible. Tolerance is set tole-4. */
/* */
/* */
/* */
/* Note 4 *******************************************/
/* Four brackets are maintained at any */
/* given time. */
/* */
/* */

if w1>wa2 then; /* If w1 > wa2 then minimum width is between */
/* alphaA and a1 */
```
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
/* Note 5 *************************************/
/* Within the interval [alphaA, a1] we have */
/* the width wa2. a2 is situated by the */
/* expression alphaA+phi*(a1-alphaA). */
/* */
/* *********************************************/

alphaB = a1;
a1 = a2;
wa1 = wa2;
a2 = alphaA+phi*phi*(alphaB-alphaA);
wa2 = w(a2,ndf1,ddf2,ncp,slAlpha);
end;
else do;
/* Updated alphaB */
*/
/* Save a2 */
*/ Save wa2 */
*/ Set new test bracket a2 */
*/ Calculate interval width for new a2 */
*/ If wa1 < wa2 then minimum width is between */
*/ a2 and alphaB */
*/
/* Note 6 *************************************/
/* Within the interval [a2, alphaB] we have */
/* the width wa1. a1 is situated by the */
/* expression alphaA+phi*phi*(alphaB-a2). */
/* */
/* *********************************************/

alphaA = a2;
a2 = a1;
wa2 = wa1;
a1 = alphaA+phi*(alphaB-alphaA);
wa1 = w(a1,ndf1,ddf2,ncp,slAlpha);
end;
end;
out = (alphaA+alphaB)/2;
return(out);

finish alphaLU;

/* Calculations */
eps = 1e-4;
/* Stopping criteria for GSS method */
n = nrow(X);
/* Number of rows */
p = ncol(X);
/* Number of columns */
J = J(p,1);
*/
xBar = X(1,1)`/n;
/* Sample means */
```
Appendix B: Two-Sample SAS Code for Power Simulations

189 \[ \text{muG} = (\text{muA} + \text{muB})/p; \] /* Mean of population means */
190 \[ \text{sst} = n \times \text{xBar}^\prime \times \text{xBar} - (n/p) \times (\text{xBar}^\prime) \times J \times 2; \] /* Sum of squares for treatments */
191 \[ \text{df1} = p - 1; \] /* Degrees of freedom for treatment */
192 \[ \text{mst} = \text{sst} / \text{df1}; \] /* Mean square for treatment */
193 \[ \text{sse} = \text{ssq}(X) - (\text{xBar}^\prime \times \text{xBar}) \times n; \] /* Degrees of freedom for error */
194 \[ \text{df2} = \text{p} \times (n - 1); \] /* Mean square for error */
195 \[ \text{mse} = \text{sse} / \text{df2}; \] /* Observed F */
196 \[ \text{fSamp} = \text{mst} / \text{mse}; \] /* Degrees of freedom for error */
197 \[ \text{fCrit} = \text{quantile}('F', 1 - \text{alpha}, \text{df1}, \text{df2}); \] /* F critical value */
198 \[ \text{pValue} = 1 - \text{cdf}('F', \text{fSamp}, \text{df1}, \text{df2}); \] /* P-Value */
199
200 \[ \text{trueLambda} = (n \times ((\text{muA} - \text{muG})^2 + (\text{muB} - \text{muG})^2)) / ((\sigma)^2); \] /* trueLambda */
201 \[ \text{truePower} = 1 - \text{cdf}('F', \text{fCrit}, \text{df1}, \text{df2}, \text{trueLambda}); \] /* truePower */
202 \[ \text{estLambda} = \text{fSamp} \times (p - 1); \] /* Estimated lambda */
203 \[ \text{adjEstLambda} = (\text{estLambda} \times (\text{df2} - 2) / \text{df2}) - \text{df1}; \] /* Adjusted estimated lambda */
204
205 if \( \text{adjEstLambda} < 0 \) then \( \text{adjEstLambda} = 0; \) /* Note 7 ****************************/
206
207 \[ \text{alphaL} = \text{alphaLU}(\text{df1}, \text{df2}, \text{adjEstLambda}, \text{alpha}, \text{eps}); \] /* Lower significance */
208 \[ \text{alphaU} = \text{alpha} - \text{alphaL}; \] /* Upper significance */
209
210
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
adjAlphaL = 1-alphaL; /* Note 8 *******************************************/
probCenF = cdf('F',fSamp,df1,df2); /* fnonct reports an error for a supplied */ /* probability greater than the */ /* probability from the cdf of a central */ /* F distribution. */ /* */ /* As the probability supplied gets large */ /* fnonct tends toward zero. probCenF */ /* with fnonct is small around zero. */ /* Using adjAlphaL would then be smaller */ /* about zero. Set to zero. */ /* */ /* *******************************************************/

if adjAlphaL > probCenF then lambdaL = 0; /* See note 8 */
   else lambdaL = fnonct(fSamp,df1,df2,adjAlphaL);
if alphaU > probCenF then lambdaU = 0; /* See note 8 */
   else lambdaU = fnonct(fSamp,df1,df2,alphaU);

estPower = 1-cdf('F',fCrit,df1,df2,adjEstLambda); /* Estimated power */
powerL = 1-cdf('F',fCrit,df1,df2,lambdaL); /* Lower limit for power */
powerU = 1-cdf('F',fCrit,df1,df2,lambdaU); /* Upper limit for power */

/* Output simulation to a data set */
Y = {"&muA" "&muB" "&sigma" "&sampleSize" "&alpha" "." "." "." "." "." "." "." "." "." "." "};
Y = num(Y);
Y[6] = pValue;
Y[7] = trueLambda;
Y[8] = truePower;
Y[9] = estLambda;
Y[10] = adjEstLambda;
Y[12] = lambdaU;
Y[13] = estPower;
Y[14] = powerL;
Y[15] = powerU;
create sim&simNum from Y;
append from Y;
quit;
```
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
proc datasets lib=work nolist;
delete sample1 sample2 sampleTwoPops;
quit;
run;
%MEND simulation2;
```
Appendix B: Two-Sample SAS Code for Power Simulations

/* sampleNormPop2 *******************************************/
/* Program Description: */
/* sampleNormPop2 generates a random sample from a normal population */
/* and outputs the values to a dataset */
/* Input: */
/* seed := dataset of random seeds from randSeed */
/* popMean := population mean */
/* popSD := population standard deviation */
/* seedIt := seed value position for generating random observations */
/* numObs := sample size */
/* sampleNum := identifies sample */
/******************************************************************************/
%macro sampleNormPop2(popMean=,popSD=,seedIt=,numObs=,sampleNum=);
  proc iml;
    reset nolog;
    use randSeed;
    read all into seed;
    x&sampleNum = &popMean+&popSD*rannor(J(&numObs,1,seed[&seedIt,&sampleNum]));
    test = seed[&seedIt,&sampleNum];
    create sample&sampleNum from x&sampleNum[colname={obs&sampleNum}];
    append from x&sampleNum;
  quit;
%mend sampleNormPop2;
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
/* avgMedMultipleSim2 ***************************************************/
/* Requirement(s): */
/* Program Description: */
/* avgMedMultipleSim2 calls avgMedSim2 for each sigma by sample size */
/* combination. */
/* Input: */
/* itM := number of iterations */
/* nsXnss := number of iteration groups */
/* (sigmas times number of sample sizes) */
/*******************************************************************/
%macro avgMedMultipleSim2(itM=,nsXnss=);
  /* Produce average and median of simulations */
  %let bStart = 1;
  %let bFinish = &itM;
  %do i = 1 %to &nsXnss %by 1;
    %avgMedSim2(start=&bStart,finish=&bFinish,itB=&i);
    %let bStart = &bStart+&itM;
    %let bFinish = &bFinish+&itM;
  %end;

  /* Combine average simulations into a data set and name columns */
  data avgMedMultipleSimOut;
    set
      amSim&j;
    %do j = 1 %to &nsXnss %by 1;
    %end;
  run;
%mend avgMedMultipleSim2;
```
/* Program Description: */
/* avgMedSim2 averages over a user specified number of rows from multipleSimOut. avgMedSim2 also determines the median of the calculated power values of the rows. */

/* Input: */
/* X := gets dataset multipleSimOut */
/* start := beginning of iteration block */
/* finish := end of iteration block */
/* itB := number of iterations */

/**************************
%macro avgMedSim2(start=,finish=,itB=);
proc iml;
reset nolog;
use multipleSimOut;
read all into X;

/* Average rows for an iteration block */
rows = X[&start:&finish,];
avgRows = rows[:,];

/* Median for column calculated power */
col = X[&start:&finish,13];
numRows = nrow(col);
mis = 0;
i = 1;
do while (i <= numRows);
   if missing((col[i,1])) > 0 then mis=mis//i;
i=i+1;
end;
***********************************************************/
Appendix B: Two-Sample SAS Code for Power Simulations

```sas
nr = nrow(mis); /* Number of missing values */

/* Note 2 ***************
*/
/* If 'nr' is greater than */
/* one then we have missing */
/* values. These values are */
/* removed from the column. */
/* */
/* ***********************************/

if nr>1 then mis = remove(mis,1); /* Remove initialize position */
if nr>1 then col = col`; /* Transpose a column to row */
if nr>1 then col = remove(col,mis); /* Remove missing values */
if nr>1 then col = col`; /* Transpose row to a column */

med = median(col); /* Find median */
/* Concatinate median to average row */
avgRows=avgRows||med;

create amSim&itB from avgRows; /* Output to a dataset */
append from avgRows;
quitting from avgMedSim2;
```
/* Requirement(s): */
/* rand3.csv, multipleSim3.sas, avgMedMultipleSim3.sas */
/* */
/* Program Description: */
/* threeSample.sas acts as a controller interface which accepts user */
/* defined values for three population means, upper and lower bounds */
/* for a range of population standard deviations, a maximum sample size, */
/* a type I error rate, and the number of iterations per sample size */
/* to population standard deviation combination. */
/* */
/***************************************************************************/
/* */
/* Input random seed file to the dataset randseed ***********************/
/* */
/* rand3.csv is a three column file of random numbers generated from */
/* http://www.random.org. For each iteration of multipleSim3 three random */
/* values are used to seed three random samples. */
/* */
/***************************************************************************/
/* Produce simulations **********************************************/
/* */
/* The multipleSim3 manages the production of simulations. */
/* Calls are made to the macro simulation3.sas which performs */
/* calculations based on Thomas 1997. */
/* */
/* Calculations for each simulation are stored in a dataset called */
/* multipleSimOut. */
/* */
/***************************************************************************/
%multipleSim3(mu1=30,mu2=40,mu3=35,sigmaMin=5,sigmaMax=40,itMax=5,sampleSizeMax=50,sigLevel=0.05);
Appendix B: Three-Sample SAS Code for Power Simulations

```sas
/* Print multipleSimOut */
proc print data=multipleSimOut;
run;
/* Produce averages and medians for simulations ***************************/
/* The avgMedMultipleSim3 manages the consolidation of simulations. */
/* Calls are made to avgMedSim3 for each sigma by sample size */
/* combination. avgMedSim3 averages over a user specified number */
/* of rows from multipleSimOut. avgMedSim3 also determines the */
/* median of the calculated power values of the rows. */
/* Calculations for each iteration block are stored in a dataset */
/* called avgMedMultipleSimOut. */
**************************************************************************
%avgMedMultipleSim3(itM=5,nsXnss=306);
/* Print avgMedMultipleSimOut */
proc print data=avgMedMultipleSimOut;
run;
/* Plot avgMedMultipleSimOut */
goptions ftitle=swiss ftext=swiss;
symbol1 value=square interpol=join width=0.5 color=blue;
symbol2 value=star interpol=join width=0.5 color=red;
symbol3 value=triangle interpol=join width=0.5 color=yellow;
symbol4 value='x' interpol=join width=0.5 color=orange;
symbol5 value=circle interpol=join width=0.5 color=blue;
symbol6 value=dot interpol=join width=0.5 color=green;
title2 height=1.4 "Sample Size vs Power";
axis1 order=(0.0 to 1.0 by 0.1) label=(angle=90 'Power') minor=none;
axis2 order=(5 to 50 by 5) label=(angle=0 'Sample Size');
legend label=none value=(h=.8 'Upper AVG Estimated Power' 'True Power' 'AVG Estimated Power' 'Median Power' 'Lower AVG Estimated Power' 'AVG P-Value')
   position=center;
```

Appendix B: Three-Sample SAS Code for Power Simulations

proc gplot data=avgMedMultipleSimOut;
plot powerU*sampleSize truePower*sampleSize estPower*sampleSize medPower*sampleSize powerL*sampleSize
pValue*sampleSize/overlay vaxis=axis1 haxis=axis2 hminor=4 legend=legend1;
by sigma;
run;
Appendix B: Three-Sample SAS Code for Power Simulations

/* multipleSim3 ***************************************************************/

/* Requirement(s): */
/* simulation3.sas */
/* Program Description: */
/* multipleSim3 iterates simulation3.sas and combines simulation output to a */
/* dataset. */

/* Input: */
/* mu1 := population one mean */
/* mu2 := population two mean */
/* mu3 := population three mean */
/* sigmaMin := minimum population standard deviation controller */
/* sigmaMax := maximum population standard deviation controller */
/* itMax := number of iterations */
/* sampleSizeMax := largest sample size considered */
/* sigLevel := level of significance */

/* Note(s): */
/* In the subsubloop iterations run in blocks. This is done to */
/* traverse the dataset randSeed, where by each call to the macro */
/* simulation2 gets a different random seed position. */
/* This macro contains two main loops. The first main loop */
/* iterates from the minimum user define population standard deviation to 19 */
/* by 1. The second main loop iterates from 20 to the maximum user defined */
/* population standard deviation by 20. Each main loop has two subloops. */
/* The first subloop iterates from a system defined minimum sample size of 5 */
/* to 14 by 1. The second subloop iterates from 15 to the maximum user */
/* defined sample size by 5. Each subloop has a subsubloop. */
/* Each subsubloop iterates by a use defined number of iterations. After */
/* a subsubloop executes, the starting position of an iteration 'block' is */
/* increased by the the user defined number of iterations. */

******************************************************************************/

%macro multipleSim3(mu1=,mu2=,mu3=,sigmaMin=,sigmaMax=,itMax=,sampleSizeMax=,sigLevel=);

%local std sd nn it block s n i b;
%let block = 1;
%let b = 1;
%let count = 1;

*/
Appendix B: Three-Sample SAS Code for Power Simulations

/* Produce simulations */

%do std = &sigmaMin %to 19 %by 1;
  %let sd = %sysevalf(&std/2);
  %do nn = 5 %to 14 %by 1;
    %do it = &block %to &block+&itMax-1 %by 1;
      %simulation3(muA=&mu1,muB=&mu2,muC=&mu3,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
    %let count = %eval(&count+1);
  %end;
  %let block = &it;
%end;

%do nn = 15 %to &sampleSizeMax %by 5;
  %do it = &block %to &block+&itMax-1 %by 1;
    %simulation3(muA=&mu1,muB=&mu2,muC=&mu3,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
    %let count = %eval(&count+1);
  %end;
  %let block = &it;
%end;

%do std = 20 %to &sigmaMax %by 20;
  %let sd = %sysevalf(&std/2);
  %do nn = 5 %to 14 %by 1;
    %do it = &block %to &block+&itMax-1 %by 1;
      %simulation3(muA=&mu1,muB=&mu2,muC=&mu3,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
    %let count = %eval(&count+1);
    %let block = &it;
  %end;
%end;

%do nn = 15 %to &sampleSizeMax %by 5;
  %do it = &block %to &block+&itMax-1 %by 1;
    %simulation3(muA=&mu1,muB=&mu2,muC=&mu3,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
    %let count = %eval(&count+1);
    %let block = &it;
%end;
Appendix B: Three-Sample SAS Code for Power Simulations

/* Combine simulations into a dataset and name columns */
data multipleSimOut;
  set
  %do k = 1 %to &count-1 %by 1;
  sim&k
  %end;
  /* Double do-loop to reference datasets. ';' closes 'set' */
rename col1=mu1 col2=mu2 col3=mu3 col4=sigma col5=sampleSize col6=sigLevel col7=pValue col8=trueLambda col9=truePower col10=estLambda col11=adjEstLambda col12=lambdaL col13=lambdaU col14=estPower col15=powerL col16=powerU;
run;
%mend multipleSim3;
Appendix B: Three-Sample SAS Code for Power Simulations

/* simulation3 **************************************************************/
/*                                                                         */
/* Requirement(s):                                                         */
/* sampleNormPop3.sas                                                     */
/*                                                                         */
/* Program Description:                                                    */
/* simulation3.sas performs a hypothesis test of no difference for three population means. Random samples are normally distributed with respective means muA, muB and muC with equal population standard deviation sigma. Random samples are generated using seed values from the dataset randSeed. Power calculations are based on methods outlined in Thomas 1997. Calculations are saved to a dataset named 'sim#' where # represents the simulation count.                                                                         */
/*                                                                         */
/* Input:                                                                  */
/* muA := mean of population one */
/* muB := mean of population two */
/* muC := mean of population three */
/* sigma := standard deviation of populations one, two, and three */
/* seedNum := position for random seed */
/* sampleSize := sample size taken for populations one, two, and three */
/* alpha := level of significance */
/* simNum := simulation ID */
/*                                                                         */
/* Note(s):                                                                */
/* Optimal lower significance level by Golden Section Search method */
/*                                                                         */
/******************************************************************************/

%MACRO simulation3(muA=,muB=,muC=,sigma=,seedNum=,sampleSize=,alpha=,simNum=);
/* Generate random samples from three normal populations */
%sampleNormPop3(popMean=&muA,popSD=&sigma,seedIt=&seedNum,numObs=&sampleSize,sampleNum=1);
%sampleNormPop3(popMean=&muB,popSD=&sigma,seedIt=&seedNum,numObs=&sampleSize,sampleNum=2);
%sampleNormPop3(popMean=&muC,popSD=&sigma,seedIt=&seedNum,numObs=&sampleSize,sampleNum=3);
/* Merge datasets sample1, sample2 and sample3 to produce dataset sampleThreePops */
data sampleThreePops;
    merge sample1 sample2 sample3;
run;

%*/
/* Main */
proc iml;
reset nolog;
use sampleThreePops;
read all into X;

/* Function ***************************************************/
/* Return the confidence interval width for a noncentral */
/* f-dsitribution */
/*******************************************************************/
start w(alphaLow,ndf,ddf,ncent,sl);
    width = quantile('F',1-sl+alphaLow,ndf,ddf,ncent)-quantile('F',alphaLow,ndf,ddf,ncent);
    return(width);
finish w;

/* Function: optimal lower critical region *******************/
/* Finds the minimum confidence interval width using an */
/* implementation of the Golden Section Search method */
/* similar to Press et al. (2007). */
/* Note(s): */
/* The Golden Section Search method finds a value which minimizes */
/* a function. The function must be one dimensional and unimodal. */
/* Imagine a 'typical' pdf of an F distribution. Say we want to */
/* setup an interval such that the area covered by the sum of the two */
/* tails is alpha. For each alpha we can find the */
/* associated quantile using the SAS function 'quantile'. */
/* Since alpha is specified, knowing the lower area easily leads to */
/* the upper area. Here we consider only the lower */
/* area. As such we can relate interval width as a function of */
/* lower alpha size. As we increase or decrease the size of the */
/* lower area we increase or decrease the width of the interval. */
/* Assuming a minimum interval width exists, we optimize over the */
/* values [0.00, 0.05]. */
/*******************************************************************/

Appendix B: Three-Sample SAS Code for Power Simulations
Appendix B: Three-Sample SAS Code for Power Simulations

```sas
start alphaLU(ndf1,ddf2,ncp,slAlpha,tol);
    alphaA = 0;  /* Initial lower bracket on optimal region */
    alphaB = slAlpha;  /* Initial upper bracket on optimal region */
    /* Note 1 *******************************************/
    phi = (-1+sqrt(5))/2;  /* Golden ratio constant reduction factor */
    a1 = alphaA+phi*(alphaB-alphaA);  /* Set test bracket a1 */
    wa1 = w(a1,ndf1,ddf2,ncp,slAlpha);  /* Calculate interval width for a1 */
    a2 = alphaA+phi*phi*(alphaB-alphaA);  /* Set test bracket a2 */
    wa2 = w(a2,ndf1,ddf2,ncp,slAlpha);  /* Calculate interval width for a2 */
    /* Note 2 *******************************************/
    do while (abs(alphaB-alphaA)>tol);  /* Note 3 *******************************************/
    /* We continue until the absolute difference */
    /* between an upper and a lower bracket is */
    /* negligible. Tolerance is set to 1e-4. */
    /* Note 4 *******************************************/
    if wa1>wa2 then do;  /* If wa1 > wa2 then minimum width is between */
        /* alphaA and a1 */
    endif;
```
Appendix B: Three-Sample SAS Code for Power Simulations

alphaB = a1;
alphaA = a2;
a1 = a2;
a2 = alphaA + phi*phi*(alphaB - alphaA);
wa1 = w(a2, ndf1, ddf2, ncp, siAlpha);
end;
else do;
  alphaB = a1;
a2 = alphaA + phi*phi*(alphaB - alphaA);
  wa2 = w(a2, ndf1, ddf2, ncp, siAlpha);
  end;
else do;
  alphaA = a2;
a1 = a2;
a2 = alphaA + phi*phi*(alphaB - alphaA);
  wa1 = w(a1, ndf1, ddf2, ncp, siAlpha);
end;
end;
out = (alphaA + alphaB)/2;
return(out);
finish alphaLU;

eps = 1e-4;
n = nrow(X);
p = ncol(X);
J = J(p, 1);
muG = (&muA + &muB + &muC)/p;
xBar = X(|+,|)`/n;
*/ Calculations */
/* Note 5 **********************************************/
/* Within the interval [alphaA, a1] we have */
/* the width wa2.  a2 is situated by the */
/* expression alphaA + phi*(a1 - alphaA). */
/* */
/* Note 6 **********************************************/
/* Within the interval [a2, alphaB] we have */
/* the width wa1.  a1 is situated by the */
/* expression alphaA + phi*phi*(alphaB - a2). */
/* */
/* Calculations */
/* Stopping criteria for GSS method */
/* Number of rows */
/* Number of columns */
/* Mean of population means */
/* Sample means */
Appendix B: Three-Sample SAS Code for Power Simulations

193  sst = n*xBar~xBar-(n/p)*(xBar~J)**2; /* Sum of squares for treatments */
194  df1 = p-1; /* Degrees of freedom for treatment */
195  mst = sst/df1; /* Mean square for treatment */
196  sse = ssq(X)-(xBar~xBar)*n; /* Sum of squares for error */
197  df2 = p*(n-1); /* Degrees of freedom for error */
198  mse = sse/df2; /* Mean square for error */
199  fSamp = mst/mse; /* Observed F */
200
201  fCrit = quantile('F',1-&alpha,df1,df2); /* F critical value */
202  pValue = 1-cdf('F',fSamp,df1,df2); /* P-Value */
203
204  trueLambda = (n*(((&muA-muG)**2)+((&muB-muG)**2)+((&muC-muG)**2)))/((&sigma)**2); /* trueLambda */
205  truePower = 1-cdf('F',fCrit,df1,df2,trueLambda); /* truePower */
206
207  estLambda = fSamp*(p-1); /* Estimated lambda */
208  adjEstLambda = (estLambda*(df2-2)/df2)-df1; /* Adjusted estimated lambda */
209
210  if adjEstLambda < 0 then adjEstLambda = 0; /* Note 7 **********************/
211
212  alphaL = alphaLU(df1,df2,adjEstLambda,&alpha,eps); /* Lower significance */
213  alphaU = &alpha-alphaL; /* Upper significance */
214
215 /* For some random seeds the */
216 /* noncentrality parameter is */
217 /* negative. */
218 /* */
219 /* A negative noncentrality */
220 /* parameter occurs when */
221 /* the expression */
222 /* estLambda*(df2-2)/df2 is */
223 /* less than df1. */
224 /* */
225 /* */
226 /* */
227 /* */
228 /* */
229 /* */
230 /* */
231 /* */
232 /* */
233 /* */
234 /* */
235 /* */
236 /* */
237 /* */
238 /* */
239 /* */
240
Appendix B: Three-Sample SAS Code for Power Simulations

```sas
adjAlphaL = 1 - alphaL; /* Note 8 ***************************************************/
probCenF = cdf('F', fSamp, df1, df2); /* fnonct reports an error for a supplied */
/* probability greater than the */
/* probability from the cdf of a central */
/* F distribution. */
/* As the probability supplied gets large */
/* fnonct tends toward zero. probCenF */
/* with fnonct is small around zero. */
/* Using adjAlphaL would then be smaller */
/* about zero. Set to zero. */
/* */
/* ***************************************************/

if adjAlphaL > probCenF then lambdaL = 0; /* See note 8 */
extPower = 1 - cdf('F', fCrit, df1, df2, adjEstLambda); /* Estimated power */

if alphaU > probCenF then lambdaU = 0; /* See note 8 */

powerL = 1 - cdf('F', fCrit, df1, df2, lambdaL); /* Lower limit for power */

powerU = 1 - cdf('F', fCrit, df1, df2, lambdaU); /* Upper limit for power */

estPower = 1 - cdf('F', fCrit, df1, df2, adjEstLambda); /* Estimated power */

/* Output simulation to a data set */

Y = {'&muA' '&muB' '&muC' '&sigma' '&sampleSize' '&alpha' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.');
Y = num(Y);
Y[7] = pValue;
Y[8] = trueLambda;
Y[9] = truePower;
Y[10] = estLambda;
Y[12] = lambdaL;
Y[13] = lambdaU;
Y[14] = estPower;
Y[15] = powerL;
Y[16] = powerU;

create sim&simNum from Y;
append from Y;
quit;
```
Appendix B: Three-Sample SAS Code for Power Simulations

proc datasets lib=work nolist;
delete sample1 sample2 sample3 sampleThreePops;
quit;
run;
%MEND simulation3;
Appendix B: Three-Sample SAS Code for Power Simulations

```sas
/* sampleNormPop3 ***********************************************************/
/* Program Description: */
/* sampleNormPop3 generates a random sample from a normal population */
/* and outputs the values to a dataset */
/* */
/* */
/* Input: */
/* */
/* seed := dataset of random seeds from randSeed */
/* */
/* popMean := population mean */
/* */
/* popSD := population standard deviation */
/* */
/* seedIt := seed value position for generating random observations */
/* */
/* numObs := sample size */
/* */
/* sampleNum := identifies sample */
/* */
/******************************************************************************/
%macro sampleNormPop3(popMean=,popSD=,seedIt=,numObs=,sampleNum=);
proc iml;
  reset nolog;
  use randSeed;
  read all into seed;
  x&sampleNum = &popMean+&popSD*rannor(J(&numObs,1,seed[&seedIt,&sampleNum]));
  test = seed[&seedIt,&sampleNum];
  create sample&sampleNum from x&sampleNum[colname={obs&sampleNum}];
  append from x&sampleNum;
quit;
%mend sampleNormPop3;
```
Appendix B: Three-Sample SAS Code for Power Simulations

```sas
/* avgMedMultipleSim3 ***************************************************/
/* Requirement(s): */
/* Program Description: */
/* Program MultipleSim3 calls avgMedSim3 for each sigma by sample size */
/* combination. */
/* Input: */
/* itM := number of iterations */
/* nsXnss := number of iteration groups */
/* (sigmas times number of sample sizes) */
/*********************************************************************/
%macro avgMedMultipleSim3(itM=,nsXnss=);
  /* Produce average and median of simulations */
  %let bStart = 1;
  %let bFinish = &itM;
  %do i = 1 %to &nsXnss %by 1;
    %avgMedSim3(start=&bStart,finish=&bFinish,itB=&i);
    %let bStart = &bStart+&itM;
    %let bFinish = &bFinish+&itM;
  %end;
  /* Combine average simulations into a data set and name columns */
  data avgMedMultipleSimOut;
    set amSim&j;
   (rename col1=mu1 col2=mu2 col3=mu3 col4=sigma col5=sampleSize col6=sigLevel col7=pValue col8=trueLambda 
     col9=truePower col10=estLambda col11=adjEstLambda col12=lambdaL col13=lambdaU col14=estPower col15=powerL 
     col16=powerU col17=medPower)
    run;
%mend avgMedMultipleSim3;
```
Appendix B: Three-Sample SAS Code for Power Simulations

```sas
/* avgMedSim3 ******************************************************/
1 /*
2 /* Program Description:
3 /* avgMedSim3 averages over a user specified number of rows from multipleSimOut. avgMedSim3 also determines the median of the calculated power values of the rows.
4 /*
5 /* Input:
6 /*
7 /* X ^= gets dataset multipleSimOut
8 /* start ^= beginning of iteration block
9 /* finish ^= end of iteration block
10 /* itB ^= number of iterations
11 /*
12 /*********************************************************************/
13
14 %macro avgMedSim3(start=,finish=,itB=);
15 proc iml;
16     reset nolog;
17     use multipleSimOut;
18     read all into X;
19     /* Average rows for an iteration block */
20     rows = X[&start:&finish,];
21     avgRows = rows[:,];
22     /* Median for column calculated power */
23     col = X[&start:&finish,14];
24     numRows = nrow(col);
25     mis = 0;
26     i = 1;
27     do while (i <= numRows);
28         if missing((col[i,1])) > 0 then mis=mis//i;
29             i=i+1;
30     end;
31     /* Note 1 ***************************/
32     /* Loop through column of calculated power values. */
33     /* If a missing value is found */
34     /* record array position. */
35     /*
36     **************************************************************************/
37 ```
Appendix B: Three-Sample SAS Code for Power Simulations

```sas
nr = nrow(mis); /* Number of missing values */

/* Note 2 *******************/
/* If 'nr' is greater than */
/* one then we have missing */
/* values. These values are */
/* removed from the column. */
/* */
/* */
/******************************/

if nr>1 then mis = remove(mis,1); /* Remove initialize position */
if nr>1 then col = col'; /* Transpose a column to row */
if nr>1 then col = remove(col,mis); /* Remove missing values */
if nr>1 then col = col'; /* Transpose row to a column */
med = median(col); /* Find median */

/* Concatinate median to average row */
avgRows=avgRows||med;

create amSim&itB from avgRows; /* Output to a dataset */
append from avgRows;
quit;
$mend avgMedSim3;
```
/* fourSample *************************************************************/
/* Requirement(s):                                                   */
/* rand4.csv, multipleSim4.sas, avgMedMultipleSim4.sas              */
/* Program Description:                                             */
/* fourSample.sas acts as a controller interface which accepts user */
/* defined values for four population means, upper and lower bounds */
/* for a range of population standard deviations, a maximum sample size, */
/* a type I error rate, and the number of iterations per sample size */
/* to population standard deviation combination.                    */
/*********************************************************************/

proc printto print='\Path\to\Output\Folder\fourSampleOut.lst'
    log='\Path\to\Output\Folder\fourSampleOut.log';
run;

/* Input random seed file to the dataset randseed ***********************/
/* rand4.csv is a four column file of random numbers generated from */
/* http:/www.random.org. For each iteration of multipleSim4 four random */
/* values are used to seed four random samples.                      */
/*********************************************************************/

data randSeed;
    infile '\Path\to\File\rand4.csv' dlm=',';
    input seedCol1 seedCol2 seedCol3 seedCol4;
run;

/* Produce simulations *************************************************/
/* The multipleSim4 manages the production of simulations.          */
/* Calls are made to the macro simulation4.sas which performs        */
/* calculations based on Thomas 1997.                              */
/* Calculations for each simulation are stored in a dataset called   */
/* multipleSimOut.                                                  */
/*********************************************************************/

%multipleSim4(mu1=30,mu2=40,mu3=35,mu4=35,sigmaMin=5,sigmaMax=40,itMax=100,sampleSizeMax=50,sigLevel=0.05);
Appendix B: Four-Sample SAS Code for Power Simulations

/* Print multipleSimOut */
proc print data=multipleSimOut;
run;

/* Produce averages and medians for simulations */
/* The avgMedMultipleSim4 manages the consolidation of simulations. */
/* Calls are made to avgMedSim4 for each sigma by sample size */
/* combination. avgMedSim4 averages over a user specified number */
/* of rows from multipleSimOut. avgMedSim4 also determines the */
/* median of the calculated power values of the rows. */
/* Calculations for each iteration block are stored in a dataset */
/* called avgMedMultipleSimOut. */
/* *****************************************************************************/
%avgMedMultipleSim4(itM=100,nsXnss=306);

/* Print avgMedMultipleSimOut */
proc print data=avgMedMultipleSimOut;
run;

/* Plot avgMedMultipleSimOut */
goptions ftitle=swiss ftext=swiss;
symbol1 value=square interpol=join width=0.5 color=blue;
symbol2 value=star interpol=join width=0.5 color=red;
symbol3 value=triangle interpol=join width=0.5 color=yellow;
symbol4 value='x' interpol=join width=0.5 color=orange;
symbol5 value=circle interpol=join width=0.5 color=blue;
symbol6 value=dot interpol=join width=0.5 color=green;
axis1 order=(0.0 to 1.0 by 0.1) label=(angle=90 'Power') minor=none;
axis2 order=(5 to 50 by 5) label=(angle=0 'Sample Size');
legend label=none value=(h=.8 'Upper AVG Estimated Power' 'True Power' 'AVG Estimated Power' 'Median Power'
           'Lower AVG Estimated Power' 'AVG P-Value')
   position=center;
Appendix B: Four-Sample SAS Code for Power Simulations

```
proc gplot data=avgMedMultipleSimOut;
  plot powerU*sampleSize truePower*sampleSize estPower*sampleSize medPower*sampleSize powerL*sampleSize pValue*sampleSize/overlay vaxis=axis1 haxis=axis2 hminor=4 legend=legend1;
  by sigma;
run;
```
Appendix B: Four-Sample SAS Code for Power Simulations

```sas
%macro multipleSim4(mu1=,mu2=,mu3=,mu4=,sigmaMin=,sigmaMax=,itMax=,sampleSizeMax=,sigLevel=);
  %local std sd nn it block s n i b;
  %let block = 1;
  %let b = 1;
  %let count = 1;

%macroname mu1 mu2 mu3 mu4 sigmaMin sigmaMax itMax sampleSizeMax sigLevel
```
Appendix B: Four-Sample SAS Code for Power Simulations

```sas
/* Produce simulations */
%do std = &sigmaMin %to 19 %by 1;
%let sd = %sysevalf(&std/2);
%do nn = 5 %to 14 %by 1;
%do it = &block %to &block+&itMax-1 %by 1;
  %simulation4(muA=&mu1,muB=&mu2,muC=&mu3,muD=&mu4,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
%let count = %eval(&count+1);
%end;
%let block = &it;
%end;
%do nn = 15 %to &sampleSizeMax %by 5;
%do it = &block %to &block+&itMax-1 %by 1;
  %simulation4(muA=&mu1,muB=&mu2,muC=&mu3,muD=&mu4,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
%let count = %eval(&count+1);
%end;
%let block = &it;
%end;
%do std = 20 %to &sigmaMax %by 20;
%let sd = %sysevalf(&std/2);
%do nn = 5 %to 14 %by 1;
%do it = &block %to &block+&itMax-1 %by 1;
  %simulation4(muA=&mu1,muB=&mu2,muC=&mu3,muD=&mu4,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
%let count = %eval(&count+1);
%end;
%let block = &it;
%end;
%do std = &sigmaMin %to 19 %by 1;
%let sd = %sysevalf(&std/2);
%do nn = 5 %to 14 %by 1;
%do it = &block %to &block+&itMax-1 %by 1;
  %simulation4(muA=&mu1,muB=&mu2,muC=&mu3,muD=&mu4,sigma=&sd,seedNum=&it,sampleSize=&nn,alpha=&sigLevel,simNum=&count);
%let count = %eval(&count+1);
%end;
%let block = &it;
%end;
```
Appendix B: Four-Sample SAS Code for Power Simulations

```sas
/* Combine simulations into a dataset and name columns */
data multipleSimOut;
  set
    %do k = 1 %to &count-1 %by 1;
      sim&k
    %end;
  /* Double do-loop to reference datasets. ';' closes 'set' */
  rename col1=mu1 col2=mu2 col3=mu3 col4=mu4 col5=sigma col6=sampleSize col7=sigLevel col8=pValue
    col9=trueLambda col10=truePower col11=estLambda col12=adjEstLambda col13=lambdaL col14=lambdaU col15=estPower
    col16=powerL col17=powerU;
run;
%mend multipleSim4;
```
Appendix B: Four-Sample SAS Code for Power Simulations

/* simulation4 ***********************************************************/
/* Requirement(s): */
/* sampleNormPop4.sas */
/* Program Description: */
/* simulation4.sas performs a hypothesis */
/* test of no difference for four population means. Random samples */
/* are normally distributed with respective means \( \mu_A \), \( \mu_B \), \( \mu_C \) and \( \mu_D \) */
/* with equal population standard deviation \( \sigma \). Random samples are */
/* generated using seed values from the dataset randSeed. Power */
/* calculations are based on methods outlined in Thomas 1997. */
/* Calculations are saved to a dataset named 'sim#' where # represents */
/* the simulation count. */
/* Input: */
/* \( \mu_A \) := mean of population one */
/* \( \mu_B \) := mean of population two */
/* \( \mu_C \) := mean of population three */
/* \( \mu_D \) := mean of population four */
/* \( \sigma \) := standard deviation of populations one and two */
/* \( \text{seedNum} \) := position for random seed */
/* \( \text{sampleSize} \) := sample size taken for populations one and two */
/* \( \alpha \) := level of significance */
/* \( \text{simNum} \) := simulation ID */
/* Note(s): */
/* Optimal lower significance level by Golden Section Search method */
/*************************************************************************/
%MACRO simulation4(muA=,muB=,muC=,muD=,sigma=,seedNum=,sampleSize=,alpha=,simNum=);
/* Generate random samples from four normal populations */
%sampleNormPop4(popMean=&muA,popSD=&sigma,seedIt=&seedNum,numObs=&sampleSize,sampleNum=1);
%sampleNormPop4(popMean=&muB,popSD=&sigma,seedIt=&seedNum,numObs=&sampleSize,sampleNum=2);
%sampleNormPop4(popMean=&muC,popSD=&sigma,seedIt=&seedNum,numObs=&sampleSize,sampleNum=3);
%sampleNormPop4(popMean=&muD,popSD=&sigma,seedIt=&seedNum,numObs=&sampleSize,sampleNum=4);
/* Merge datasets sample1, sample2, sample3 and sample4 to produce dataset sampleFourPops */
data sampleFourPops;
  merge sample1 sample2 sample3 sample4;
run;
/* Main */
proc iml;
reset nolog;
use sampleFourPops;
read all into X;

/* Function **********************************************************/
/* Return the confidence interval width for a noncentral f-distribution */
/*****************************************************************************/
start w(alphaLow,ndf,ddf,ncent,sl);
    width = quantile('F',1-sl+alphaLow,ndf,ddf,ncent)-quantile('F',alphaLow,ndf,ddf,ncent);
    return(width);
finish w;

/* Function: optimal lower critical region **************************/
/* Finds the minimum confidence interval width using an implementation of the Golden Section Search method similar to Press et al. (2007). */
/* Note(s): Imagine a 'typical' pdf of an F distribution. Say we want to setup an interval such that the area covered by the sum of the two tails is alpha. For each alpha we can find the associated quantile using the SAS function 'quantile'. Since alpha is specified, knowing the lower area easily leads to the upper area. Here we consider only the lower area. As such we can relate interval width as a function of lower alpha size. As we increase or decrease the size of the lower area we increase or decrease the width of the interval. Assuming a minimum interval width exists, we optimize over the values [0.00, 0.05]. */
Appendix B: Four-Sample SAS Code for Power Simulations

```sas
start alphaLU(nda1, ddf2, ncp, slAlpha, tol);
  alphaA = 0; /* Initial lower bracket on optimal region */
  alphaB = slAlpha; /* Initial upper bracket on optimal region */
  /* Note 1 *******************************************/
  /* For each iteration an interval */
  /* [alphaA, alphaB] is retained */
  /* ***********************************************************************/

  phi = (-1+sqrt(5))/2; /* Golden ratio constant reduction factor */
  a1 = alphaA+phi*(alphaB-alphaA); /* Set test bracket a1 */
  wa1 = w(a1, ndf1, ddf2, ncp, slAlpha); /* Calculate Interval width for a1 */
  a2 = alphaA+phi*phi*(alphaB-alphaA); /* Set test bracket a2 */
  wa2 = w(a2, ndf1, ddf2, ncp, slAlpha); /* Calculate interval width for a2 */
  /* Note 2 *******************************************/
  /* The intial setup invokes two function */
  /* calls. The following loop uses one */
  /* function call per iteration. */
  /* ***********************************************************************/

  do while (abs(alphaB-alphaA)>tol); /* Note 3 *******************************************/
    /* We continue until the absolute difference */
    /* between an upper and a lower bracket is */
    /* negligible. Tolerance is set tole-4. */
    /* ***********************************************************************/

    if wa1>wa2 then do; /* Note 4 *******************************************/
      /* If wa1 > wa2 then minimum width is between */
      /* alphaA and a1 */
      /* ***********************************************************************/
  ```
Appendix B: Four-Sample SAS Code for Power Simulations

```sas
/* Note 5 *********************************************/
142 /* */
143 /* Within the interval [alphaA, a1] we have */
144 /* the width wa2. a2 is situated by the */
145 /* expression alphaA+phi*(a1-alphaA). */
146 /* */
147 /* ***********************************************/
148
149 alphaB = a1;
150 a1 = a2;
151 wa1 = wa2;
152 a2 = alphaA*phi*phi*(alphaB-alphaA);
153 wa2 = w(a2,ndf1,ddf2,ncp,slAlpha);
154 end;
155 else do;
156 */
157 /* Set new test bracket a2 */
158 Calculate interval width for new a2 */
159 /* If wa1 < wa2 then minimum width is between */
160 /* a2 and alphaB */
161 /* */
162 /* Note 6 *********************************************/
163 /* */
164 /* Within the interval [a2, alphaB] we have */
165 /* the width wa1. a1 is situated by the */
166 /* expression alphaA+phi*phi*(alphaB-a2). */
167 /* */
168 /* ***********************************************/
169
170 alphaA = a2;
171 a2 = a1;
172 wa2 = wa1;
173 a1 = alphaA*phi*phi*(alphaB-alphaA);
174 wa1 = w(a1,ndf1,ddf2,ncp,slAlpha);
175 end;
176 end;
177
178 out = (alphaA+alphaB)/2;
179 return(out);
180
181 finish alphaLU;
/* Calculations */
182 /* Stopping criteria for GSS method */
183 eps = 1e-4;
184 n = NROW(X);
185 p = NCOL(X);
186 J = J(p,1);
187 muG = (muA+muB+muC+muD)/p;
188 xBar = X(+$,`)/n;
/* Mean of population means */
/* Sample means */
```

Appendix B: Four-Sample SAS Code for Power Simulations

```sas
    sst = n*xBar'xBar-(n/p)*(xBar'J)**2; /* Sum of squares for treatments */
    df1 = p-1; /* Degrees of freedom for treatment */
    mst = sst/df1; /* Mean square for treatment */
    sse = ssq(X)-(xBar'xBar)*n; /* Sum of squares for error */
    df2 = p*(n-1); /* Degrees of freedom for error */
    mse = sse/df2; /* Mean square for error */
    fSamp = mst/mse; /* Observed F */
    fCrit = quantile('F',1-&alpha,df1,df2); /* F critical value */
    pValue = 1-cdf('F',fSamp,df1,df2); /* P-Value */
    trueLambda = (n*((&muA-muG)**2)+((&muB-muG)**2)+((&muC-muG)**2)+((&muD-muG)**2))/((&sigma)**2); /* trueLambda */
    truePower = 1-cdf('F',fCrit,df1,df2,trueLambda); /* truePower */
    estLambda = fSamp*(p-1); /* Estimated lambda */
    adjEstLambda = (estLambda*(df2-2)/df2)-df1; /* Adjusted estimated lambda */
    if adjEstLambda < 0 then adjEstLambda = 0; /* Note 7 */
    alphaL = alphaLU(df1,df2,adjEstLambda,&alpha,eps); /* Lower significance */
    alphaU = &alpha-alphaL; /* Upper significance */
```

*Notes:

1. For some random seeds the noncentrality parameter is negative.
2. A negative noncentrality parameter occurs when the expression estLambda*(df2-2)/df2 is less than df1.
Appendix B: Four-Sample SAS Code for Power Simulations

```sas
adjAlphaL = 1-alphaL;            /* Note 8 ***************************/
probCenF = cdf('F',fSamp,df1,df2); /* fnonct reports an error for a supplied */
                  /* probability greater than the */
                  /* probability from the cdf of a central */
                  /* F distribution. */
                  /* */
                  /* As the probability supplied gets large */
                  /* fnonct tends toward zero. probCenF */
                  /* with fnonct is small around zero. */
                  /* Using adjAlphaL would then be smaller */
                  /* about zero. Set to zero. */
                  /* */
                  /**************************************************************************/

if adjAlphaL > probCenF then lambdaL = 0; /* See note 8 */
else lambdaL = fnonct(fSamp,df1,df2,adjAlphaL);

if alphaU > probCenF then lambdaU = 0; /* See note 8 */
else lambdaU = fnonct(fSamp,df1,df2,alphaU);

estPower = 1-cdf('F',fCrit,df1,df2,adjEstLambda); /* Estimated power */
powerL = 1-cdf('F',fCrit,df1,df2,lambdaL); /* Lower limit for power */
powerU = 1-cdf('F',fCrit,df1,df2,lambdaU); /* Upper limit for power */

/* Output simulation to a data set */
Y = {'&muA' '&muB' '&muC' '&muD' '&sigma' '&sampleSize' '&alpha' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' .''};
Y = num(Y);
Y[8] = pValue;
Y[9] = trueLambda;
Y[10] = truePower;
Y[12] = adjEstLambda;
Y[13] = lambdaL;
Y[14] = lambdaU;
Y[15] = estPower;
Y[16] = powerL;
Y[17] = powerU;
create sim&simNum from Y;
append from Y;
quit;
```
Appendix B: Four-Sample SAS Code for Power Simulations

```sas
proc datasets lib=work nolist;
   delete sample1 sample2 sample3 sample4 sampleFourPops;
   quit;
run;
%MEND simulation4;
```
Appendix B: Four-Sample SAS Code for Power Simulations

%macro sampleNormPop4(popMean=,popSD=,seedIt=,numObs=,sampleNum=);
proc iml;
   reset nolog;
   use randSeed;
   read all into seed;
   x&sampleNum = &popMean+&popSD*rannor(J(&numObs,1,seed[&seedIt,&sampleNum]));
   test = seed[&seedIt,&sampleNum];
   create sample&sampleNum from x&sampleNum[colname={obs&sampleNum}];
   append from x&sampleNum;
quit;
%mend sampleNormPop4;
Appendix B: Four-Sample SAS Code for Power Simulations

%macro avgMedMultipleSim4(itM=,nsXnss=);
  %let bStart = 1;
  %let bFinish = &itM;
  %do i = 1 %to &nsXnss %by 1;
    %avgMedSim4(start=&bStart,finish=&bFinish,itB=&i);
    %let bStart = &bStart+&itM;
    %let bFinish = &bFinish+&itM;
  %end;

  data avgMedMultipleSimOut;
    set
      %do j = 1 %to &nsXnss %by 1;
        amSim&j
      %end;
    run;
  %mend avgMedMultipleSim4;
Appendix B: Four-Sample SAS Code for Power Simulations

```sas
/* avgMedSim4  *********************************************/
/* Program Description: */
/* avgMedSim4 averages over a user specified number of rows from */
/* multipleSimOut. avgMedSim4 also determines the median */
/* of the calculated power values of the rows. */
/* Input: */
/* X : gets dataset multipleSimOut */
/* start : beginning of iteration block */
/* finish : end of iteration block */
/* itB : number of iterations */
/*********************************************************************/
%macro avgMedSim4(start=,finish=,itB=);
proc iml;
    reset nolog;
    use multipleSimOut;
    read all into X;
    /* Average rows for an iteration block */
    rows = X[&start:&finish,];
    avgRows = rows[1,];
    /* Median for column calculated power */
    col = X[&start:&finish,15];
    numRows = nrow(col);
    mis = 0;
    i = 1;
    do while (i <= numRows);
        if missing((col[i,1])) > 0 then mis=mis/i;
        i=i+1;
    end;
/*/ Note 1 **********************************************/
```

139
nr = nrow(mis);  /* Number of missing values */
/* Note 2 **********************/
/* If 'nr' is greater than */
/* one then we have missing */
/* values. These values are */
/* removed from the column. */
/* */
/***********************
if nr>1 then mis = remove(mis,1);
if nr>1 then col = col`;  /* Remove initialize position */
if nr>1 then col = remove(col,mis);  /* Remove missing values */
if nr>1 then col = col`;  /* Transpose row to a column */
med = median(col);  /* Find median */
/* Concatinate median to average row */
avgRows=avgRows||med;
create amSim&itB from avgRows;  /* Output to a dataset */
quit;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
append from avgRows;
%mend avgMedSim4;
Appendix C: Graphs of Treatment Arrangement 30, 40
Appendix C: Graphs of Treatment Arrangement 30, 40
Appendix C: Graphs of Treatment Arrangement 30, 40

Sample Size vs Power

\( \sigma = 4 \)
Appendix C: Graphs of Treatment Arrangement 30, 40

Sample Size vs Power

\( \sigma_{\text{true}} = 4.5 \)

- [Legend]
  - AVG Upper 95% Estimated Power
  - True Power
  - AVG Lower 95% Estimated Power
  - AVG Estimated Power
  - Median Estimated Power
  - AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40
Appendix C: Graphs of Treatment Arrangement 30, 40
Appendix C: Graphs of Treatment Arrangement 30, 40

![Sample Size vs Power Graph](image)

- □ □ AVG Upper 95% Estimated Power
- - - True Power
- ○ ○ ○ AVG Lower 95% Estimated Power
- △ △ △ AVG Estimated Power
- ★ ★ ★ Median Estimated Power
- ● ● ● AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40

Sample Size vs Power

gamma = 5.5

Power

Sample Size

AVG Upper 95% Estimated Power
True Power
AVG Lower 95% Estimated Power
AVG Estimated Power
Median Estimated Power
AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40
Appendix C: Graphs of Treatment Arrangement 30, 40

Sample Size vs Power

\[ \sigma = 7.5 \]

- AVG Upper 95\% Estimated Power
- True Power
- AVG Lower 95\% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40

Sample Size vs Power

\( \sigma = 8 \)

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG p-value
Appendix C: Graphs of Treatment Arrangement 30, 40

Sample Size vs Power

\[ \sigma = 0.5 \]

Power

Sample Size

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40
Appendix C: Graphs of Treatment Arrangement 30, 40
Appendix C: Graphs of Treatment Arrangement 30, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

\( \sigma = 2.5 \)

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

σ = 3

Power vs Sample Size

Key:
- □ □ AVG Upper 95% Estimated Power
- --- True Power
- ○ ○ ○ AVG Lower 95% Estimated Power
- △ △ △ AVG Estimated Power
- * * * Median Estimated Power
- ● ● ● AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

σ = 3.5

Power

Sample Size

Legend:
- □ □ AVG Upper 95% Estimated Power
- + - True Power
- ○ ○ AVG Lower 95% Estimated Power
- Δ △ AVG Estimated Power
- ★ ★ Median Estimated Power
- ● ● AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

Sample Size

Power

σ = 4.5

Legend:
- □ □ □ AVG Upper 95% Estimated Power
- □ □ □ True Power
- ○ ○ ○ AVG Lower 95% Estimated Power
- ▲ ▲ ▲ AVG Estimated Power
- ◊ ◊ ◊ Median Estimated Power
- ⋄ ⋄ AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

Sample Size

Power

5 10 15 20 25 30 35 40 45 50

- - AVG Upper 95% Estimated Power
- - True Power
- - AVG Lower 95% Estimated Power
- - AVG Estimated Power
- - Median Estimated Power
- - AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

Sample Size

Power

5 10 15 20 25 30 35 40 45 50

AVG Upper 95% Estimated Power
True Power
AVG Lower 95% Estimated Power
AVG Estimated Power
Median Estimated Power
AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

Sample Size

Power
Appendix C: Graphs of Treatment Arrangement 30, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

$\sigma = 8$

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

\( \sigma \text{mea}=0.5 \)

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

$\sigma = 9$

Sample Size vs Power

Power vs Sample Size

Graph showing the relationship between sample size and power, with various markers indicating different estimations and true power values.
Appendix C: Graphs of Treatment Arrangement 30, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 40

Sample Size vs Power

| sigma = 20 |

---

Sample Size

Power

| AVG Upper 95% Estimated Power |
| True Power |
| AVG Estimated Power |
| Median Estimated Power |
| AVG P-Value |
Appendix C: Graphs of Treatment Arrangement 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40

Sample Size vs Power
\( \sigma = 3 \)

Sample Size

Power

AVG Upper 95% Estimated Power

True Power

AVG Lower 95% Estimated Power

AVG Estimated Power

Median Estimated Power

AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40

Sample Size vs Power

Sample Size vs Power

Sample Size

Power

Legend:
- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40, 40

Sample Size vs Power

\( \sigma_{\text{true}} = 4.5 \)

- \( \square\square\square\square \) AVG Upper 95\% Estimated Power
- \( \triangle\triangle\triangle\triangle \) AVG Estimated Power
- \( \star\star\star\star \) Median Estimated Power
- \( \bigcirc\bigcirc\bigcirc\bigcirc \) AVG Lower 95\% Estimated Power
- \( \bullet\bullet\bullet\bullet \) AVG P-value

Power

Sample Size

5 10 15 20 25 30 35 40 45 50
Appendix C: Graphs of Treatment Arrangement 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40

Sample Size vs Power
$\sigma_{true}=5.5$

![Graph showing sample size vs power with different markers for AVG Upper 95% Estimated Power, True Power, AVG Lower 95% Estimated Power, AVG Estimated Power, Median Estimated Power, AVG P-value.](image-url)
Appendix C: Graphs of Treatment Arrangement 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40

Sample Size vs Power

\( \sigma_{\text{me}} = 7.5 \)

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40

Sample Size vs Power

σ = 3.5

Sample Size

Power

- - - - AVG Upper 95% Estimated Power

True Power

- - - - AVG Lower 95% Estimated Power

AVG Estimated Power

Median Estimated Power

AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40, 40

Sample Size vs Power

Sample Size

Power

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40

Sample Size vs Power

Sample Size

Power

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40

Sample Size vs Power

\( \text{sigma}=4.5 \)

- AVG Upper 95\% Estimated Power
- True Power
- AVG Lower 95\% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-value
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40

Sample Size vs Power

$\sigma = 5.5$

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40

Sample Size vs Power
\( \sigma = 6 \)

- [Legend for graph elements]

- [Graph description and analysis]
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40

Sample Size vs Power

sigma=8

![Graph showing sample size vs power]
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40

Sample Size vs Power

σ = 5.5
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40

Sample Size vs Power

σ = 10

Power

Sample Size

Legend:
- □□ AVG Upper 95% Estimated Power
- ✔✔ True Power
- ○○ AVG Lower 95% Estimated Power
- ▲▲ AVG Estimated Power
- ★★ Median Estimated Power
- •• AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 35, 35, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40

Sample Size vs Power

\[ \text{Sample Size vs Power} \]

\( \sigma = 5 \)

- **AVG Upper 95\% Estimated Power**
- **True Power**
- **AVG Lower 95\% Estimated Power**
- **AVG Estimated Power**
- **Median Estimated Power**
- **AVG P-Value**
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40

Sample Size vs Power

\[ \sigma = 5.5 \]

![Graph showing sample size vs power with various markers for different estimates and true power and p-value.](image-url)
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40

Sample Size vs Power

\( \sigma = 7.5 \)

Power vs Sample Size

- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40

![Sample Size vs Power Graph](image-url)

- **Sample Size vs Power**
- **σ = 0.5**

- **Legend:**
  - □ □ □ AVG Upper 95% Estimated Power
  - + + + True Power
  - ○ ○ ○ AVG Lower 95% Estimated Power
  - △ △ △ AVG Estimated Power
  - ★ ★ ★ Median Estimated Power
  - ● ● ● AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40

Sample Size vs Power

Sample Size vs Power Graph for σ = 0.5

Legend:
- □ □ □ AVG Upper 95% Estimated Power
- ▲ ▲ ▲ AVG Lower 95% Estimated Power
- ● ● ● AVG P-Value
- ○ ○ ○ AVG Estimated Power
- ▲ ▲ ▲ AVG Estimated Power
- ⬤ ⬤ ⬤ Median Estimated Power
- — — — True Power
Appendix C: Graphs of Treatment Arrangement 30, 30, 40, 40

Sample Size vs Power

sigma = 20

Legend:
- AVG Upper 95% Estimated Power
- True Power
- AVG Lower 95% Estimated Power
- AVG Estimated Power
- Median Estimated Power
- AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40

Sample Size vs Power

σ = 2.5

Power

Sample Size

- Average Upper 95% Estimated Power
- True Power
- Average Lower 95% Estimated Power
- Average Estimated Power
- Median Estimated Power
- Average P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40

Sample Size vs Power

$\sigma = 5$

Sample Size vs Power

- **AVG Upper 95% Estimated Power**
- **True Power**
- **AVG Lower 95% Estimated Power**
- **AVG Estimated Power**
- **Median Estimated Power**
- **AVG P-Value**
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40

Sample Size vs Power

\( \sigma = 6 \)

![Sample Size vs Power Graph](image)

Legend:
- **AVG Upper 95% Estimated Power**
- **True Power**
- **AVG Lower 95% Estimated Power**
- **AVG Estimated Power**
- **Median Estimated Power**
- **AVG P-Value**

Sample Size (x-axis)

Power (y-axis)
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40

Sample Size vs Power
\( \sigma_{\text{true}} = 2.5 \)

---

Legend:
- □ □ □ AVG Upper 95% Estimated Power
- + + + True Power
- ○ ○ ○ AVG Lower 95% Estimated Power
- △ △ △ AVG Estimated Power
- × × × Median Estimated Power
- • • • AVG P-Value
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix C: Graphs of Treatment Arrangement 30, 40, 40, 40
Appendix D: SAS Code for Estimating Power for an Observed $F$-Statistic

```sas
/* Application *******************************************/
/* Input: significance level, observed $F$-statistic, and */
/* numerator and denominator degrees of freedom. */
/* Output: estimate for power (1-alpha)% confidence interval for power, and */
/* p-value of the test */
/***************************************************************************/
%macro application1(F=, df1=, df2=, alpha=);
proc iml;
reset nolog;
  /* Function *************************************************/
  /* Return the confidence interval width for a noncentral */
  /* f-distribution */
  /***************************************************************************/
  start w(alphaLow, ndf, ddf, ncent, sl);
    width = quantile('F', 1-sl+alphaLow, ndf, ddf, ncent) - quantile('F', alphaLow, ndf, ddf, ncent);
    return(width);
  finish w;
```
Appendix D: SAS Code for Estimating Power for an Observed $F$-Statistic

```sas
/* Function: optimal lower critical region ******************************* */
/* Finds the minimum confidence interval width using an */
/* implementation of the Golden Section Search method by */
/* Kincaid and Cheney 2009, and Jan Verschelde */
/* http://www.math.uic.edu/~jan/mcs471/Lec9/gss.pdf */
/* */
/* Note(s):
*/
/* The Golden Section Search method finds a value which minimizes */
/* a function. The function must be one dimensional and unimodal. */
/* */
/* Imagine a 'typical' pdf of an $F$ distribution. Say we want to */
/* setup an interval such that the area covered by the sum of the two */
/* tails is alpha. For each alpha we can find the */
/* associated quantile using the SAS function 'quantile'. */
/* Since alpha is specified, knowing the lower area easily leads to */
/* the upper area. Here we consider only the lower */
/* area. As such we can relate interval width as a function of */
/* lower alpha size. As we increase or decrease the size of the */
/* lower area we increase or decrease the width of the interval. */
/* Assuming a minimum interval width exists, we optimize over the */
/* values [0.00, 0.05]. */
/* */
/**********************************************************************/
start alphaLU(ndf1,ddf2,ncp,slAlpha,tol);
alphaA = 0;  /* Initial lower bracket on optimal region */
alphaB = slAlpha;  /* Initial upper bracket on optimal region */
/* Note 1 *******************************************/
/* For each iteration an interval */
/* [alphaA, alphaB] is retained */
phi = (-1+sqrt(5))/2;  /* Golden ratio constant reduction factor */
a1 = alphaA+phi*(alphaB-alphaA);  /* Set test bracket a1 */
a2 = alphaA+phi*phi*(alphaB-alphaA);  /* Set test bracket a2 */
a1 = w(a1,ndf1,ddf2,ncp,slAlpha);  /* Calculate Interval width for a1 */
a2 = w(a2,ndf1,ddf2,ncp,slAlpha);  /* Calculate Interval width for a2 */
```
Appendix D: SAS Code for Estimating Power for an Observed $F$-Statistic

```sas
/* Note 2 *********************************************/
/* The initial setup invokes two function calls. The following loop uses one function call per iteration. */
do while (abs(alphaB-alphaA)>tol);
/* Note 3 *********************************************/
/* We continue until the absolute difference between an upper and a lower bracket is negligible. Tolerance is set to 1e-4. */
if wa1>wa2 then do;
/* Note 4 *********************************************/
/* Four brackets are maintained at any given time. */
alphaB = a1;
a1 = a2;
wa1 = wa2;
a2 = alphaA+phi*(alphaB-alphaA);
wa2 = w(a2,ndf1,ddf2,ncp,slAlpha);
end;
else do;
/* Note 5 *********************************************/
/* Within the interval $[alphaA, a1]$ we have the width $wa2$. $a2$ is situated by the expression $alphaA+phi*(a1-alphaA)$. */
alphaB = w(a2,ndf1,ddf2,ncp,slAlpha);
a1 = a2;
wa1 = wa2;
a2 = alphaA+phi*(alphaB-alphaA);
wa2 = w(a2,ndf1,ddf2,ncp,slAlpha);
end;
else do;
/* If $wa1 < wa2$ then minimum width is between $alphaB$ and $a1$ */
alphaB = w(a2,ndf1,ddf2,ncp,slAlpha);
a1 = a2;
wa1 = wa2;
a2 = alphaA+phi*(alphaB-alphaA);
wa2 = w(a2,ndf1,ddf2,ncp,slAlpha);
end;
```

Appendix D: SAS Code for Estimating Power for an Observed $F$-Statistic
/* Note 6 *********************************************/
/* Within the interval [a2, alphaB] we have */
/* the width wal. a1 is situated by the */
/* expression alphaA+phi*(alphaB-a2). */
/* */
/* ************************************************/

alphaA = a2;
a2 = a1;
wal = wal;
a1 = alphaA+phi*(alphaB-alphaA);
wal = w(a1,ndf1,ddf2,ncp,slAlpha);
end;
end;

out = (alphaA+alphaB)/2;
return(out);

finish alphaLU;

/* Calculations */
eps = 1e-4;
fCrit = quantile('F',1-alpha,ndf1,ddf2);
pValue = 1-cdf('F',F,ndf1,ddf2);
estLambda = F*ddf1;
adjEstLambda = (estLambda*(ddf2-2)/ddf2)-ndf1;
if adjEstLambda < 0 then adjEstLambda = 0;

/* Note 7 *********************************************/
/* For some random seeds the */
/* noncentrality parameter is */
/* negative. */
/* */
/* A negative noncentrality */
/* parameter occurs when */
/* /* */
/* /* F critical value */
/* P-Value */
/* Estimated lambda */
/* Adjusted estimated lambda */
/* Note 7 *********************************************/
/* For some random seeds the */
/* noncentrality parameter is */
/* negative. */
/* */
/* A negative noncentrality */
/* parameter occurs when */
/* /* */
/* /* F critical value */
/* P-Value */
/* Estimated lambda */
/* Adjusted estimated lambda */
/* Note 7 *********************************************/
Appendix D: SAS Code for Estimating Power for an Observed $F$-Statistic

```sas
alphaL = alphaLU(&df1,&df2,adjEstLambda,&alpha,eps);  /* Lower significance */
alphaU = &alpha-alphaL;  /* Upper significance */
adjAlphaL = 1-alphaL;
probCenF = cdf('F',&F,&df1,&df2);  /* Note 8 */

if adjAlphaL > probCenF then lambdaL = 0;  /* See note 8 */
if alphaU > probCenF then lambdaU = 0;  /* See note 8 */
else lambdaL = fnonct(&F,&df1,&df2,adjAlphaL);
else lambdaU = fnonct(&F,&df1,&df2,alphaU);

estPower = 1-cdf('F',fCrit,&df1,&df2,adjEstLambda);  /* Estimated power */
powerL = 1-cdf('F',fCrit,&df1,&df2,lambdaL);  /* Lower limit for power */
powerU = 1-cdf('F',fCrit,&df1,&df2,lambdaU);  /* Upper limit for power */

print estPower;
print powerL, powerU, pValue;
quit;
%mend application1;
%application1(F=1.5,df1=4,df2=10,alpha=0.05);
```
Appendix D: Section 5 Output for Example 1

The SAS System

1. ESTPOWER
2. 0.0975334
3. POWERL
4. 0.05
5. POWERU
6. 0.8017428
7. PVALUE
8. 0.2278353
Appendix D: Section 5 Output for Example 2

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

The SAS System

ESTPOWER
0.0777412

POWEL
0.05

POWERU
0.6713596

PVALUE
0.2741814