Canonical Correlation Analysis II

Mike Strube

December 5, 2018

1 Preliminaries

The RStudio workspace and console panes are cleared of old output, variables, and other miscellaneous debris. Then some packages are loaded and the required data files are input.

1.1 Clear the Console Panes and Load Packages

```
options(replace.assign = TRUE, width = 65, digits = 4, scipen = 4, fig.width = 4,
    fig.height = 4)
# Clear the workspace and console.
rm(list = ls(all = TRUE))
cat("\f")
```

```
# Turn off showing of significance asterisks.
options(show.signif.stars = F)
# Set the contrast option; important for ANOVAs.
options(contrasts = c("contr.sum", "contr.poly"))
how_long <- Sys.time()
set.seed(123)
library(knitr)
```

```
library(psych)

## Warning:  package 'psych' was built under R version 3.5.1

library(ggplot2)

## Warning:  package 'ggplot2' was built under R version 3.5.1
##
## Attaching package:  'ggplot2'
## The following objects are masked from 'package:psych':
##
##     %+%, alpha

library(MASS)

## Warning:  package 'MASS' was built under R version 3.5.1

library(sciplot)
library(dplyr)
```

```
## Warning:  package 'dplyr' was built under R version 3.5.1
##
## Attaching package:  'dplyr'
## The following object is masked from 'package:MASS':
##
##    select
## The following objects are masked from 'package:stats':
##
##    filter, lag
## The following objects are masked from 'package:base':
##
##    intersect, setdiff, setequal, union
```

```
library(aod)
library(MVN)
```

```
## sROC 0.1-2 loaded
```

```
library(boot)
```

```
##
## Attaching package:  'boot'
## The following object is masked from 'package:psych':
##
##    logit
```

```
library(car)
```

```
## Warning:  package 'car' was built under R version 3.5.1
## Loading required package:  carData
## Warning:  package 'carData' was built under R version 3.5.1
##
## Attaching package:  'car'
## The following object is masked from 'package:boot':
##
##    logit
## The following object is masked from 'package:dplyr':
##
##    recode
## The following object is masked from 'package:psych':
##
##    logit
```

```
library(LogisticDx)
library(biotools)
```

```
## Loading required package:  rpanel
## Loading required package:  tcltk
## Package 'rpanel', version 1.1-4:  type help(rpanel) for summary information
##
## Attaching package:  'rpanel'
## The following object is masked from 'package:boot':
##
##    poisons
## Loading required package:  tkrplot
## Warning:  package 'tkrplot' was built under R version 3.5.1
## Loading required package:  lattice
```

```
## Warning:  package 'lattice' was built under R version 3.5.1
##
## Attaching package:  'lattice'
## The following object is masked from 'package:boot':
##
##    melanoma
## Loading required package:  SpatialEpi
## Loading required package:  sp

## ---
## biotools version 3.1

##

library(multcomp)

## Loading required package:  mvtnorm
## Loading required package:  survival
## Warning:  package 'survival' was built under R version 3.5.1
##
## Attaching package:  'survival'
## The following object is masked from 'package:boot':
##
##    aml
## The following object is masked from 'package:aod':
##
##    rats
## Loading required package:  TH.data
## Warning:  package 'TH.data' was built under R version 3.5.1
##
## Attaching package:  'TH.data'
## The following object is masked from 'package:MASS':
##
##    geyser

library(candisc)

## Loading required package:  heplots
##
## Attaching package:  'heplots'
## The following object is masked from 'package:biotools':
##
##    boxM
##
## Attaching package:  'candisc'
## The following object is masked from 'package:stats':
##
##    cancor

library(ez)
library(GGally)

##
## Attaching package:  'GGally'
## The following object is masked from 'package:dplyr':
##
##    nasa
```

```
library(qqplotr)
```

```
## Warning: package 'qqplotr' was built under R version 3.5.1
```

```
##
## Attaching package: 'qqplotr'
## The following objects are masked from 'package:ggplot2':
##
##     stat_qq_line, StatQqLine
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(reshape)
```

```
## Warning: package 'reshape' was built under R version 3.5.1
```

```
##
## Attaching package: 'reshape'
## The following object is masked from 'package:dplyr':
##
##     rename
```

```
library(emmeans)
```

```
## Warning: package 'emmeans' was built under R version 3.5.1
```

```
##
## Attaching package: 'emmeans'
## The following object is masked from 'package:GGally':
##
##     pigs
## The following object is masked from 'package:multcomp':
##
##     cld
```

```
library(profileR)
```

```
## Warning: package 'profileR' was built under R version 3.5.1
## Loading required package: RColorBrewer
## Loading required package: lavaan
## Warning: package 'lavaan' was built under R version 3.5.1
## This is lavaan 0.6-3
## lavaan is BETA software! Please report any bugs.
##
## Attaching package: 'lavaan'
## The following object is masked from 'package:psych':
##
##     cor2cov
```

```
library(Rmisc)
```

```
## Loading required package: plyr
## ------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------
##
## Attaching package:  'plyr'
## The following objects are masked from 'package:reshape':
##
##    rename, round_any
## The following objects are masked from 'package:dplyr':
##
##    arrange, count, desc, failwith, id, mutate, rename,
##    summarise, summarize
```

```
library(CCP)
library(caTools)
```

```
## Warning:  package 'caTools' was built under R version 3.5.1
```

1.2 Data

```
setwd("C:\\Courses\\Psychology 516\\PowerPoint\\2018")

# Get the data for the cross-validation example.
CCA_Cross <- read.table("cancorr_cross.csv", sep = ",", header = TRUE)
CCA_Cross <- as.data.frame(CCA_Cross)

# Separate into the calibration and hold-out samples.  Standardize
# the variables within each sample.
Calibration <- CCA_Cross[CCA_Cross$Sample == 1, ]
Calibration[, 3:11] <- scale(Calibration[, 3:11])
Hold_Out <- CCA_Cross[CCA_Cross$Sample == 2, ]
Hold_Out[, 3:11] <- scale(Hold_Out[, 3:11])

# The CCA_Cross file has a variable, Sample, that represents a
# random splitting of the data file.  That can also be done in R,
# using a variety of methods.  Here is one based on the package,
# caTools.  To make the results replicable, set the seed:
# set.seed(123).

# sample = sample.split(CCA_Cross$ID, SplitRatio = .5) Calibration
# = subset(CCA_Cross, sample == TRUE) Hold_Out = subset(CCA_Cross,
# sample == FALSE).
```

2 Capitalizing on Chance

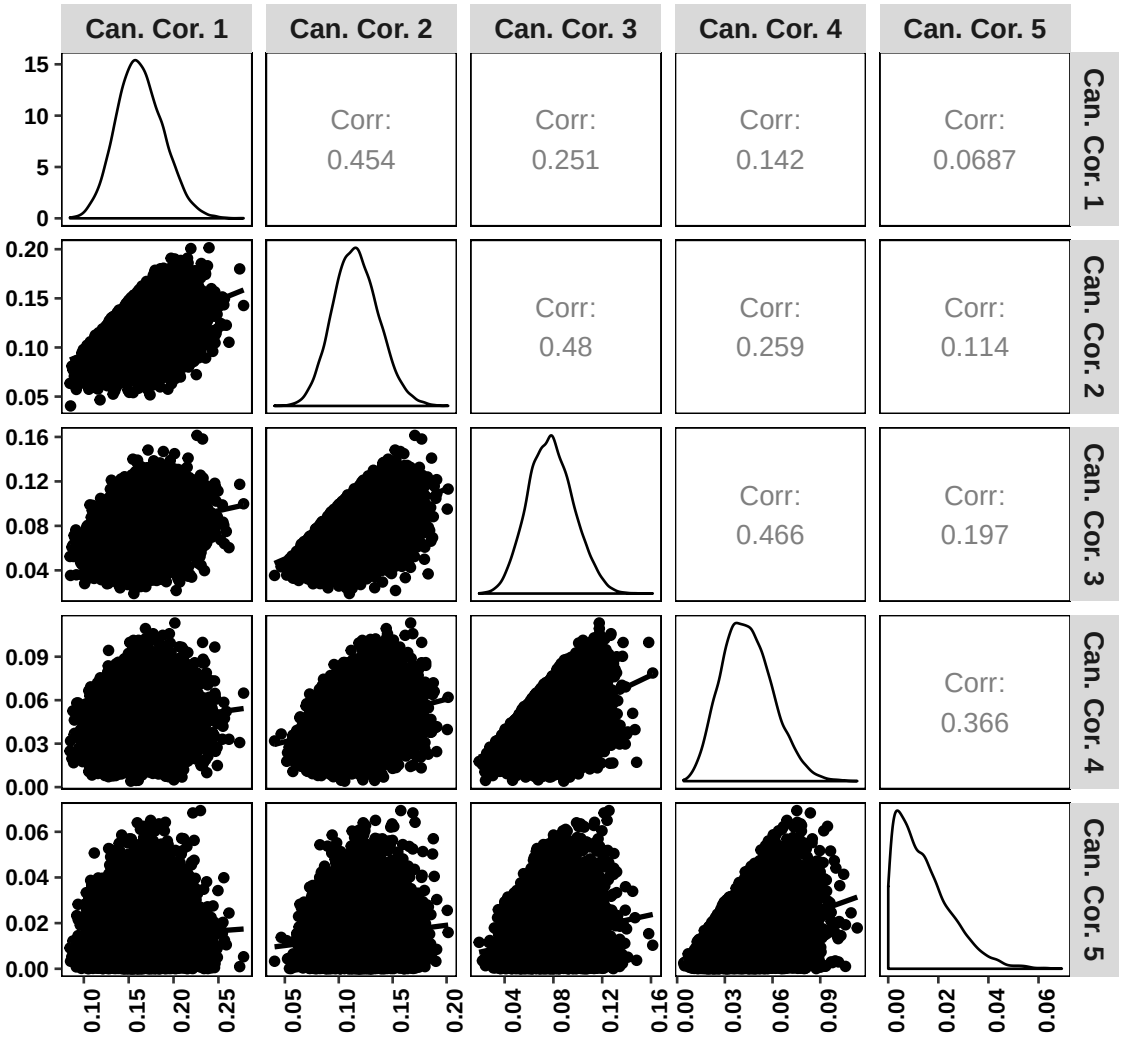
Canonical correlation analysis has elements of two procedures-principal components analysis and multiple regression analysis-that are often criticized as susceptible to capitalizing on chance. How worried should we be? An easy way to answer questions like this is through simulation with data generated to have particular properties.

In this demonstration, we draw variables from a multivariate normal distribution in which the correlations among variables within sets is set at .5 but the correlations between sets is set to 0. The simulation repeatedly draws samples, conducts a canonical correlation analysis, and saves results for inspection.

```
# Set the population variance-covariance matrix and mean vector.
VC <- matrix(c(1, 0.5, 0.5, 0.5, 0.5, 0, 0, 0, 0, 0, 0.5, 1, 0.5,
    0.5, 0.5, 0, 0, 0, 0, 0, 0.5, 0.5, 1, 0.5, 0.5, 0, 0, 0, 0, 0,
    0.5, 0.5, 0.5, 1, 0.5, 0, 0, 0, 0, 0, 0.5, 0.5, 0.5, 0.5, 1, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0.5, 0.5, 0.5, 0.5, 0, 0, 0, 0,
    0, 0.5, 1, 0.5, 0.5, 0.5, 0, 0, 0, 0, 0, 0.5, 0.5, 1, 0.5, 0.5,
    0, 0, 0, 0, 0, 0.5, 0.5, 0.5, 1, 0.5, 0, 0, 0, 0, 0, 0.5, 0.5,
    0.5, 0.5, 1), nrow = 10, ncol = 10)
VC

##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]  1.0  0.5  0.5  0.5  0.5  0.0  0.0  0.0  0.0   0.0
##  [2,]  0.5  1.0  0.5  0.5  0.5  0.0  0.0  0.0  0.0   0.0
##  [3,]  0.5  0.5  1.0  0.5  0.5  0.0  0.0  0.0  0.0   0.0
##  [4,]  0.5  0.5  0.5  1.0  0.5  0.0  0.0  0.0  0.0   0.0
##  [5,]  0.5  0.5  0.5  0.5  1.0  0.0  0.0  0.0  0.0   0.0
##  [6,]  0.0  0.0  0.0  0.0  0.0  1.0  0.5  0.5  0.5   0.5
##  [7,]  0.0  0.0  0.0  0.0  0.0  0.5  1.0  0.5  0.5   0.5
##  [8,]  0.0  0.0  0.0  0.0  0.0  0.5  0.5  1.0  0.5   0.5
##  [9,]  0.0  0.0  0.0  0.0  0.0  0.5  0.5  0.5  1.0   0.5
## [10,]  0.0  0.0  0.0  0.0  0.0  0.5  0.5  0.5  0.5   1.0

M <- matrix(rep(0, 10), nrow = 1, ncol = 10)

CCA_Sim_Results <- matrix(NA, nrow = 10000, ncol = 20)
CCA_Sim_Results <- as.data.frame(CCA_Sim_Results)
names(CCA_Sim_Results) <- c("CC1", "CC2", "CC3", "CC4", "CC5", "AL1",
    "BL1", "AW1", "BW1", "AC1", "BC1", "AA1", "BA1", "AR", "BR", "p1",
    "p2", "p3", "p4", "p5")

for (sim in 1:10000) {
    sample <- as.data.frame(mvrnorm(n = 500, M, VC))
    names(sample) <- c("A1", "A2", "A3", "A4", "A5", "B1", "B2", "B3",
        "B4", "B5")
    CCA_Sim <- cancor(cbind(A1, A2, A3, A4, A5) ~ B1 + B2 + B3 + B4 +
        B5, data = sample, prefix = c("B_Set_", "A_Set_"), set.names = c("B_Set",
        "A_Set"))
    B_Structure <- as.matrix(CCA_Sim$structure$X.xscores)
    A_Structure <- as.matrix(CCA_Sim$structure$Y.yscores)
    CCA_CanCor <- as.matrix(CCA_Sim$cancor)
    B_Weights <- as.matrix(coef(CCA_Sim, standardize = TRUE, type = "both")[[1]])
    A_Weights <- as.matrix(coef(CCA_Sim, standardize = TRUE, type = "both")[[2]])
    CCA_Red_B <- as.matrix(redundancy(CCA_Sim))[[1]]
```

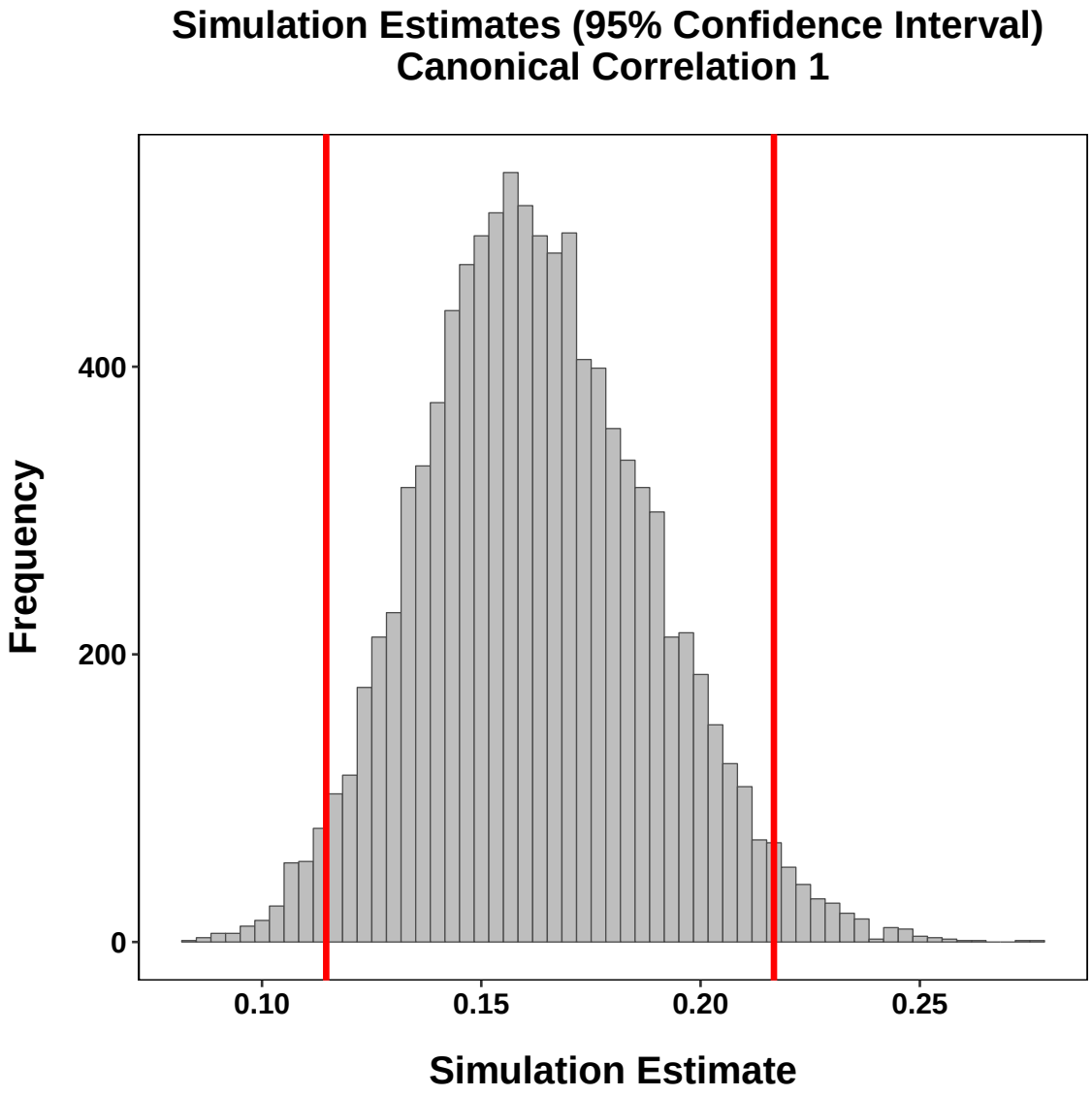
```
    CCA_Red_A <- as.matrix(redundancy(CCA_Sim))[[2]]
    A_Communalities <- rowSums((A_Structure)^2)
    B_Communalities <- rowSums((B_Structure)^2)
    A_Adequacy <- colSums((A_Structure)^2)/length(A_Structure[, 1])
    B_Adequacy <- colSums((B_Structure)^2)/length(B_Structure[, 1])
    capture.output(pv <- p.asym(CCA_Sim$cancor, 500, 5, 5, tstat = "Wilks"),
        file = "NUL")
    CCA_Sim_Results[sim, ] <- cbind(t(CCA_CanCor[, 1]), A_Structure[1,
        1], B_Structure[1, 1], A_Weights[1, 1], B_Weights[1, 1], A_Communalities[1],
        B_Communalities[1], A_Adequacy[1], B_Adequacy[1], sum(CCA_Red_A),
        sum(CCA_Red_B), t(as.matrix(pv$p.value)))
}
```

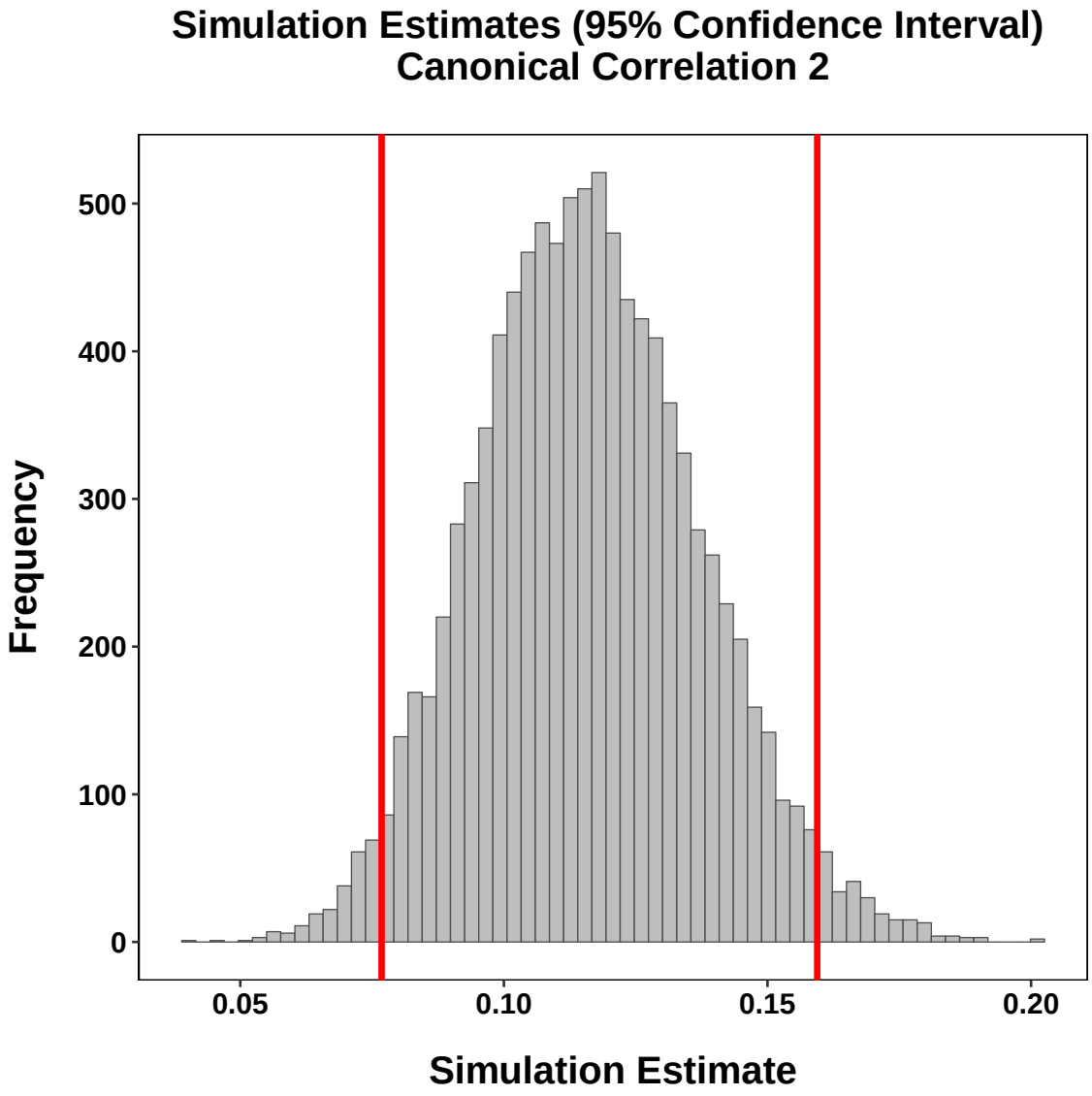
```
ggpairs(CCA_Sim_Results[, 1:5], lower = list(continuous = "smooth"),
    upper = list(continuous = "cor"), columnLabels = c("Can. Cor. 1",
        "Can. Cor. 2", "Can. Cor. 3", "Can. Cor. 4", "Can. Cor. 5")) +
    theme(text = element_text(size = 14, family = "sans", color = "black",
        face = "bold"), axis.text.y = element_text(colour = "black",
        size = 9, face = "bold"), axis.text.x = element_text(colour = "black",
        size = 9, face = "bold", angle = 90), axis.title.x = element_text(margin = margin(15,
        0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
        15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", margin = margin(0,
            0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
            linetype = 1, color = "black"), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
        plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
        legend.title = element_blank()) + ggtitle("Canonical Correlations")
```

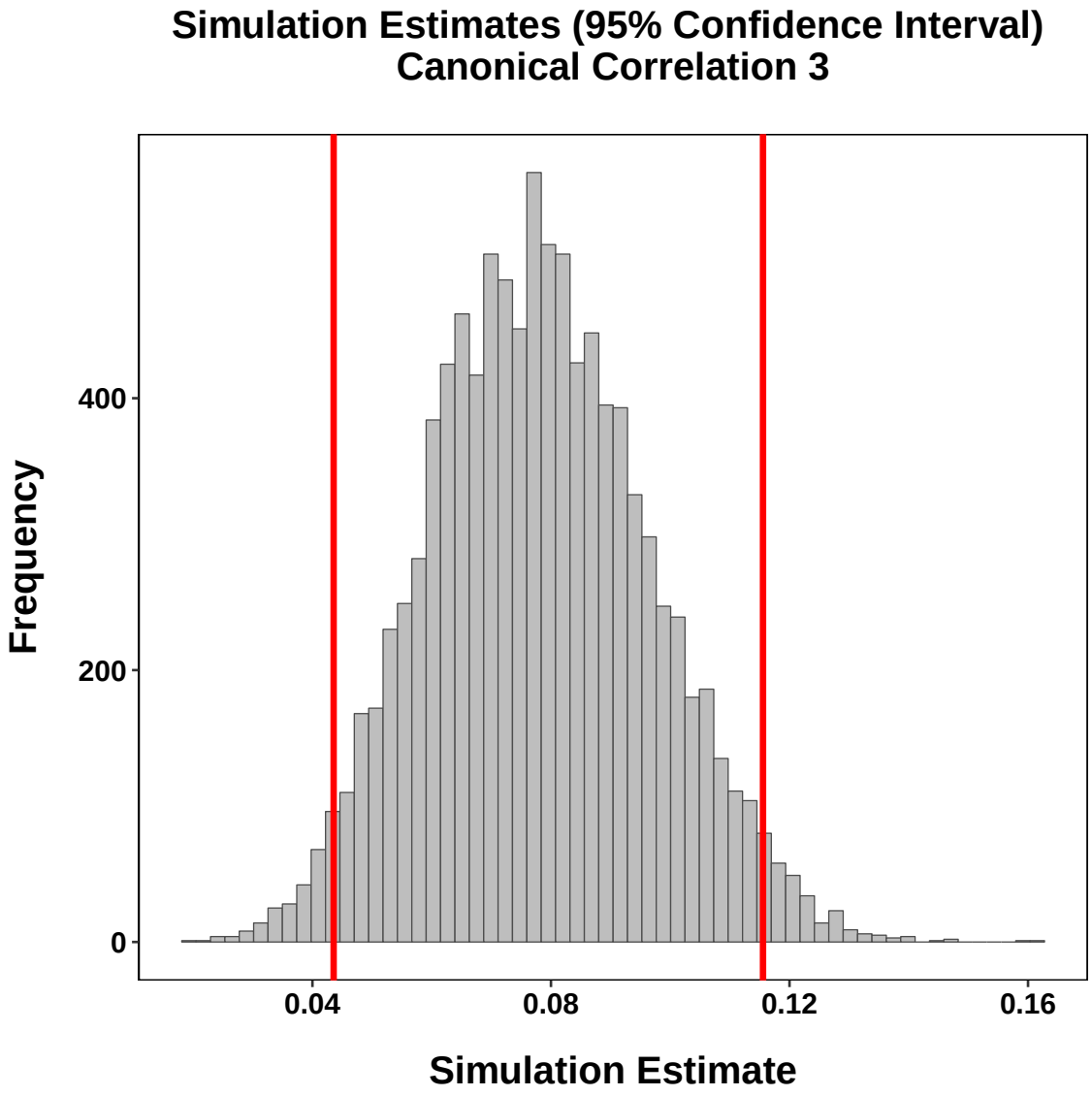


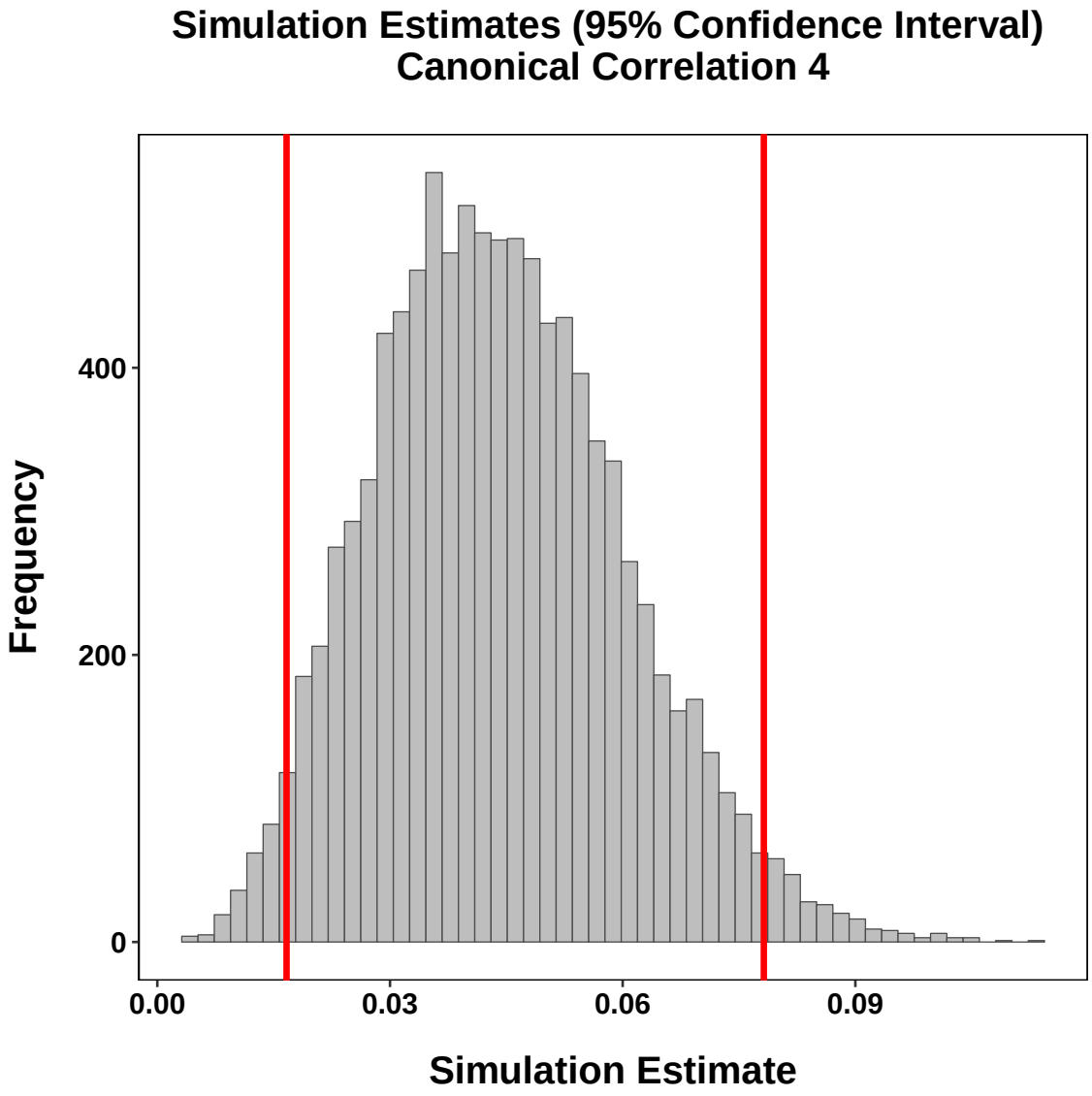

```
Effects <- c("Canonical Correlation 1", "Canonical Correlation 2",
    "Canonical Correlation 3", "Canonical Correlation 4", "Canonical Correlation 5",
    "A Set, Loading A1 on Function 1", "B Set, Loading B1 on Function 1",
    "A Set, Weight A1 on Function 1", "B Set, Weight B1 on Function 1",
    "A Set, Adequacy for Function 1", "B Set, Adequacy for Function 1",
    "A Set Redundancy", "B Set Redundancy")

can_corr_data <- CCA_Sim_Results[, c(1:9, 12:15)]

# Number of bins specified using the Friedman-Diaconis rule.
for (j in seq(1, 13, 1)) {
    plot_data <- as.data.frame(can_corr_data[, j])
    names(plot_data) <- c("t")
    plot <- ggplot(plot_data, aes(x = t)) + geom_histogram(bins = round((max(plot_data$t) -
```

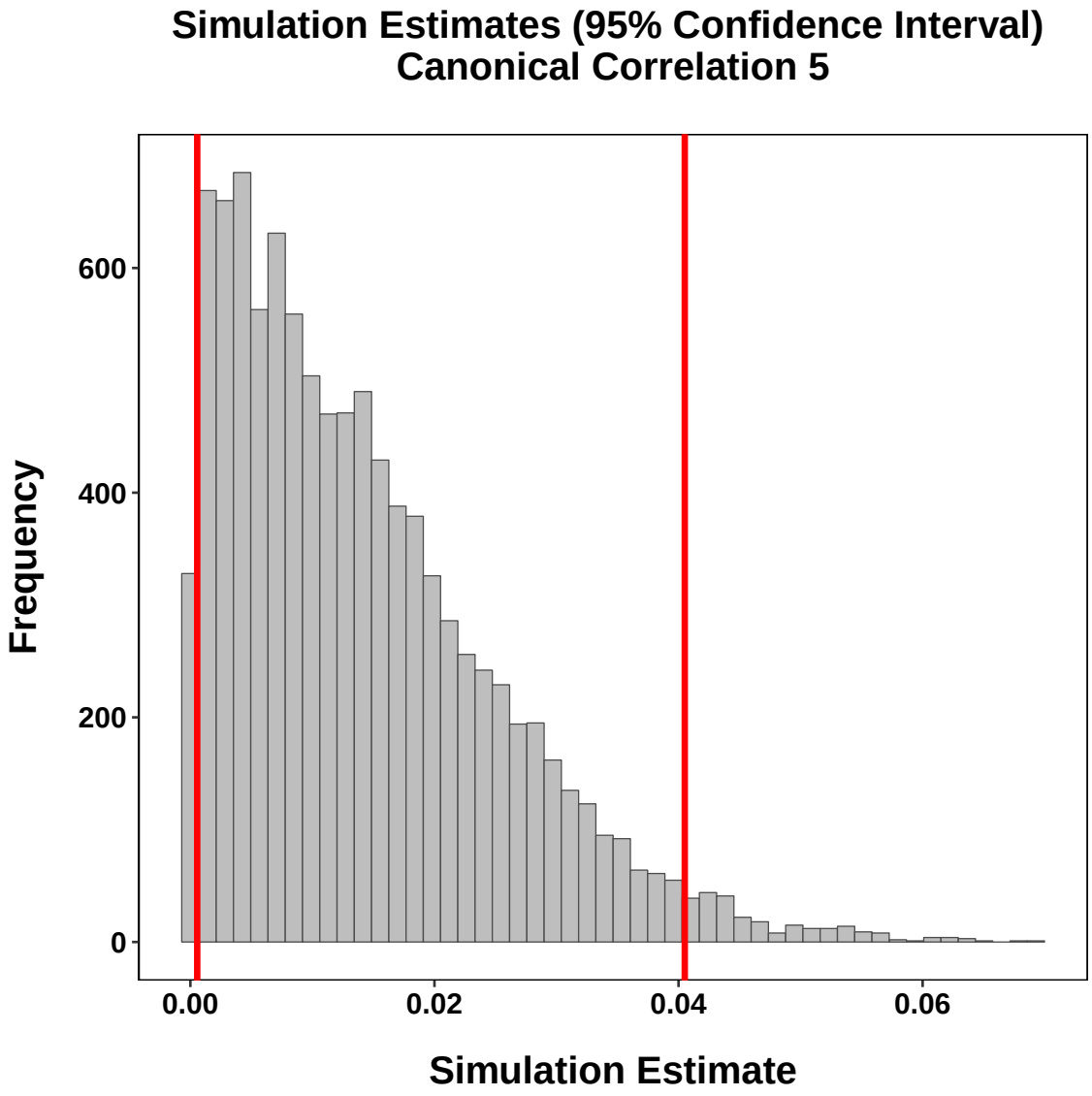
```
        min(plot_data$t))/(2 * IQR(plot_data$t) * length(plot_data$t)^(-1/3))),
        color = "grey30", fill = "grey", size = 0.01, na.rm = TRUE)

    p <- ggplot(plot_data, aes(x = t)) + geom_histogram(bins = round((max(plot_data$t) -
        min(plot_data$t))/(2 * IQR(plot_data$t) * length(plot_data$t)^(-1/3))),
        color = "grey30", fill = "grey", size = 0.25, na.rm = TRUE) +
        xlab("Simulation Estimate") + ylab("Frequency") + theme(text = element_text(size = 14,
        family = "sans", color = "black", face = "bold"), axis.text.y = element_text(colour = "black",
        size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
        size = 12, angle = 0, face = "bold"), axis.title.x = element_text(margin = margin(15,
        0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
        15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", margin = margin(0,
            0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
            linetype = 1, color = "black"), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
        plot.margin = unit(c(1, 1, 1, 1), "cm")) + geom_vline(xintercept = quantile(plot_data$t,
        probs = 0.025), size = 1.25, color = "red") + geom_vline(xintercept = quantile(plot_data$t,
        probs = 0.975), size = 1.25, color = "red") + ggtitle(paste("Simulation Estimates (95% Confidenc
        toString(Effects[j]), sep = ""))
    print(p)
}
```

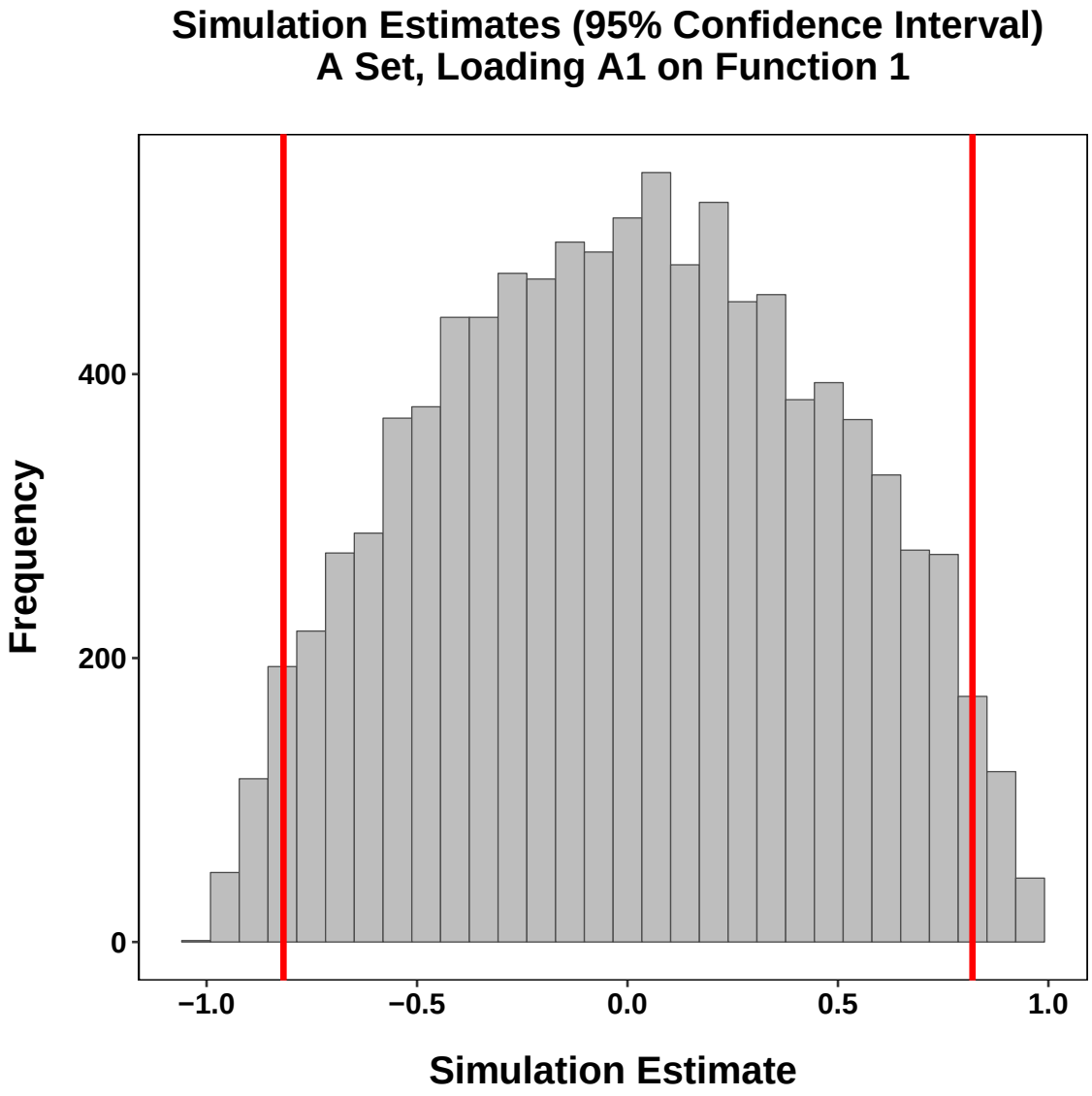

Simulation Estimates (95% Confidence Interval)
Canonical Correlation 1
Frequency
400
200
0
0.10
0.15
0.20
0.25
Simulation Estimate

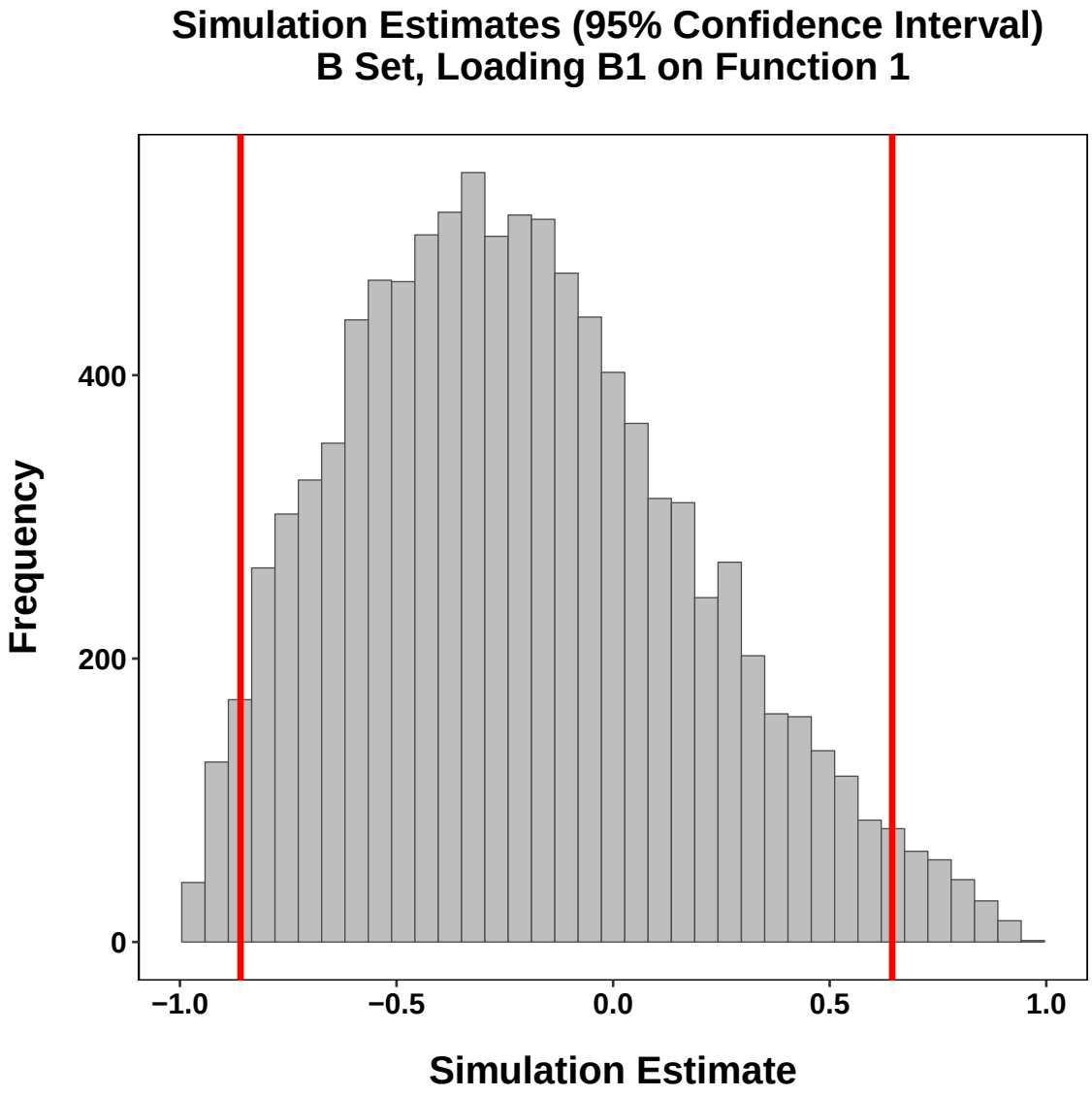

Simulation Estimates (95% Confidence Interval)
Canonical Correlation 2
Frequency
500
400
300
200
100
0
0.05
0.10
0.15
0.20
Simulation Estimate

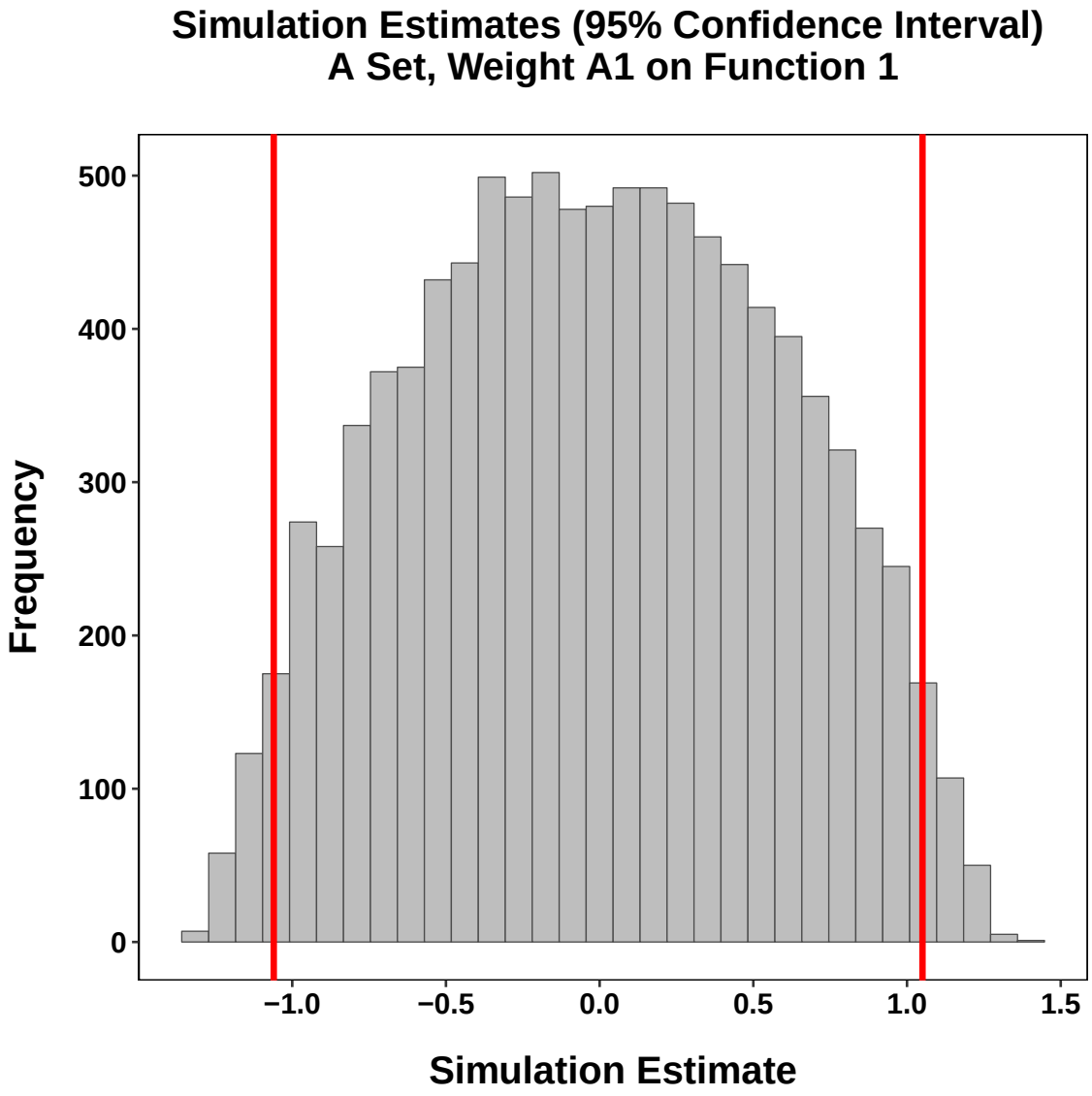

Simulation Estimates (95% Confidence Interval)
Canonical Correlation 3
Frequency
400
200
0
0.04
0.08
0.12
0.16
Simulation Estimate

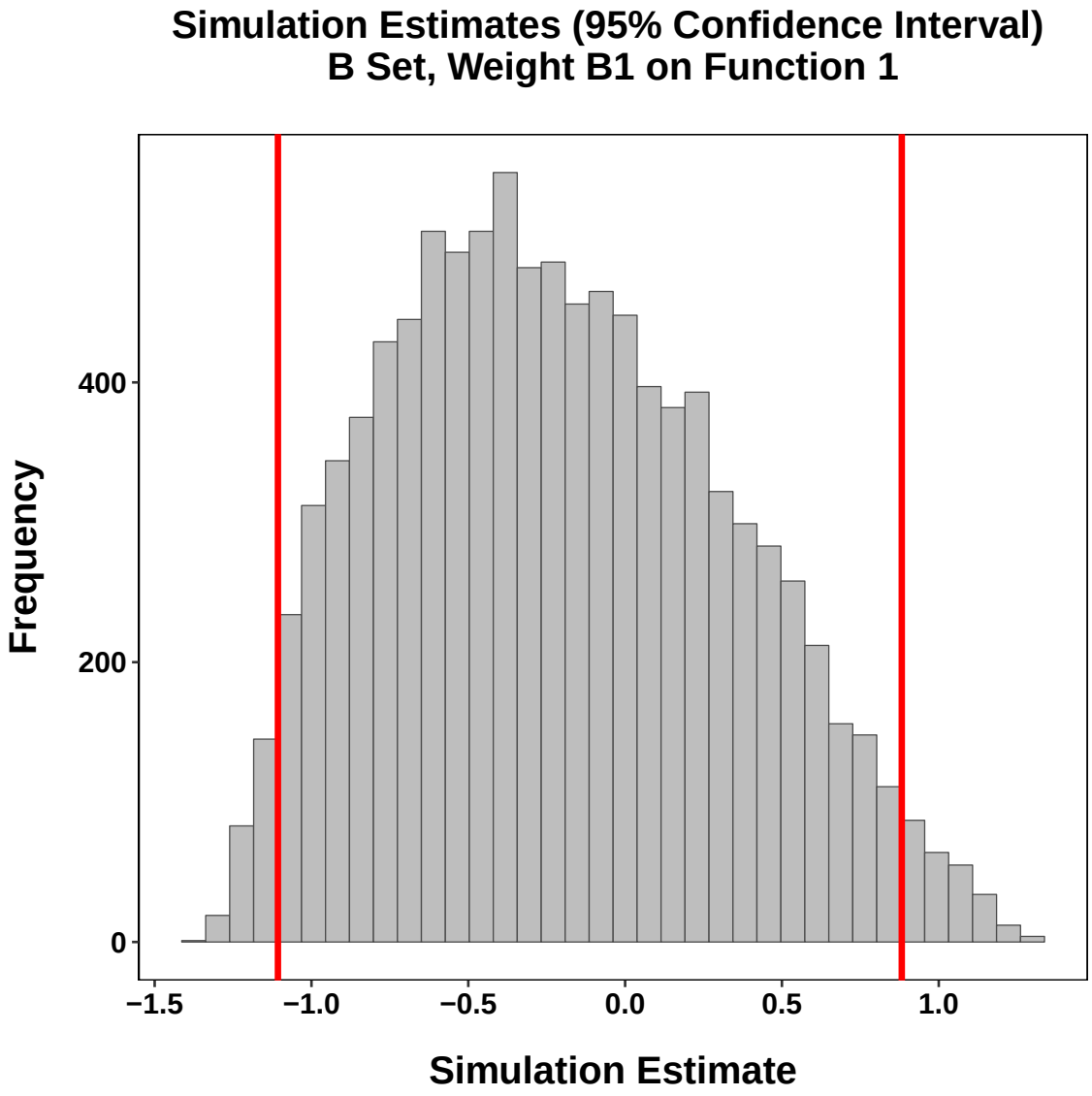

Simulation Estimates (95% Confidence Interval)
Canonical Correlation 4
Frequency
400
200
0
0.00
0.03
0.06
0.09
Simulation Estimate

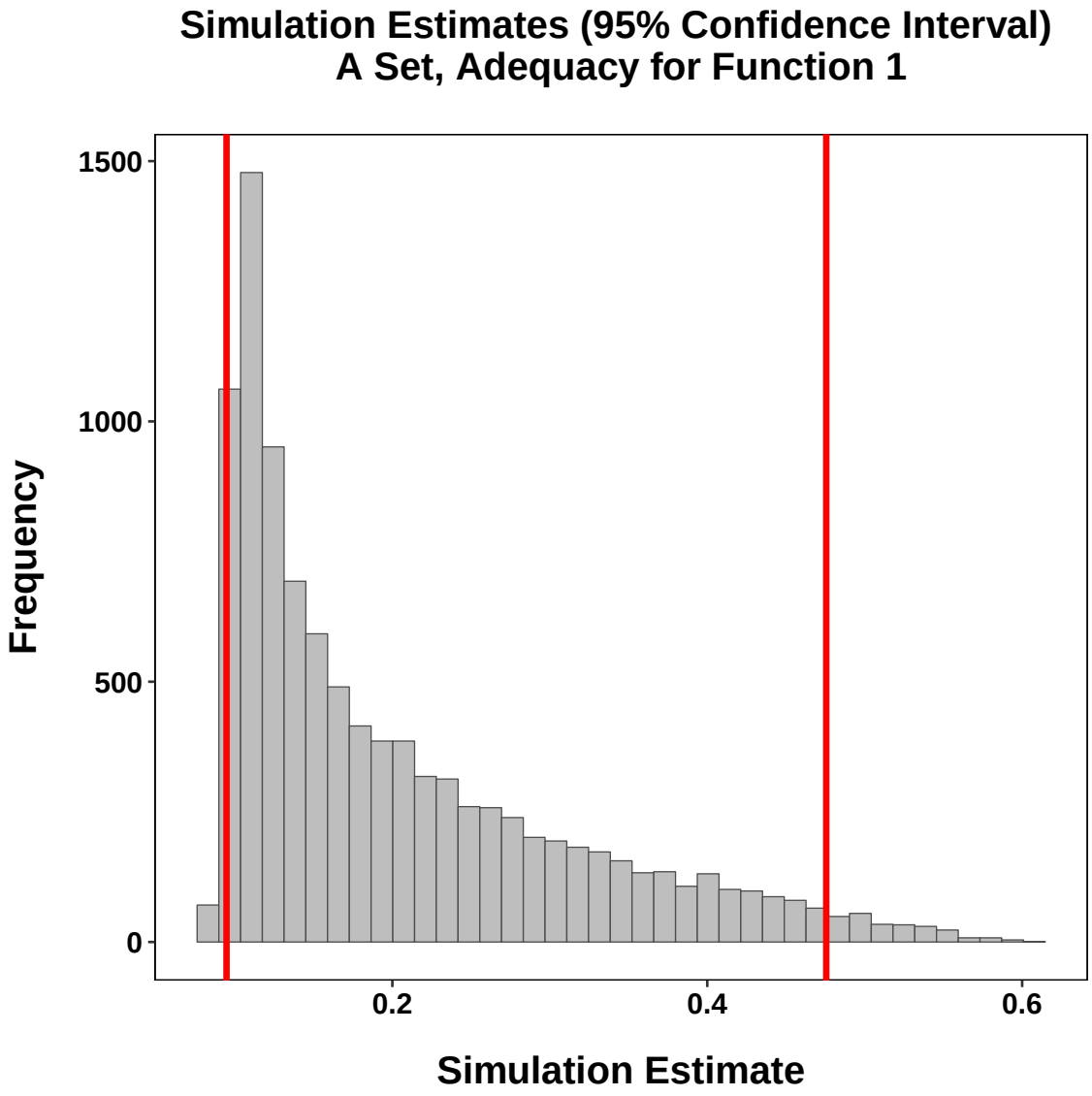

Simulation Estimates (95% Confidence Interval)
Canonical Correlation 5
Frequency
600
400
200
0
0.00
0.02
0.04
0.06
Simulation Estimate

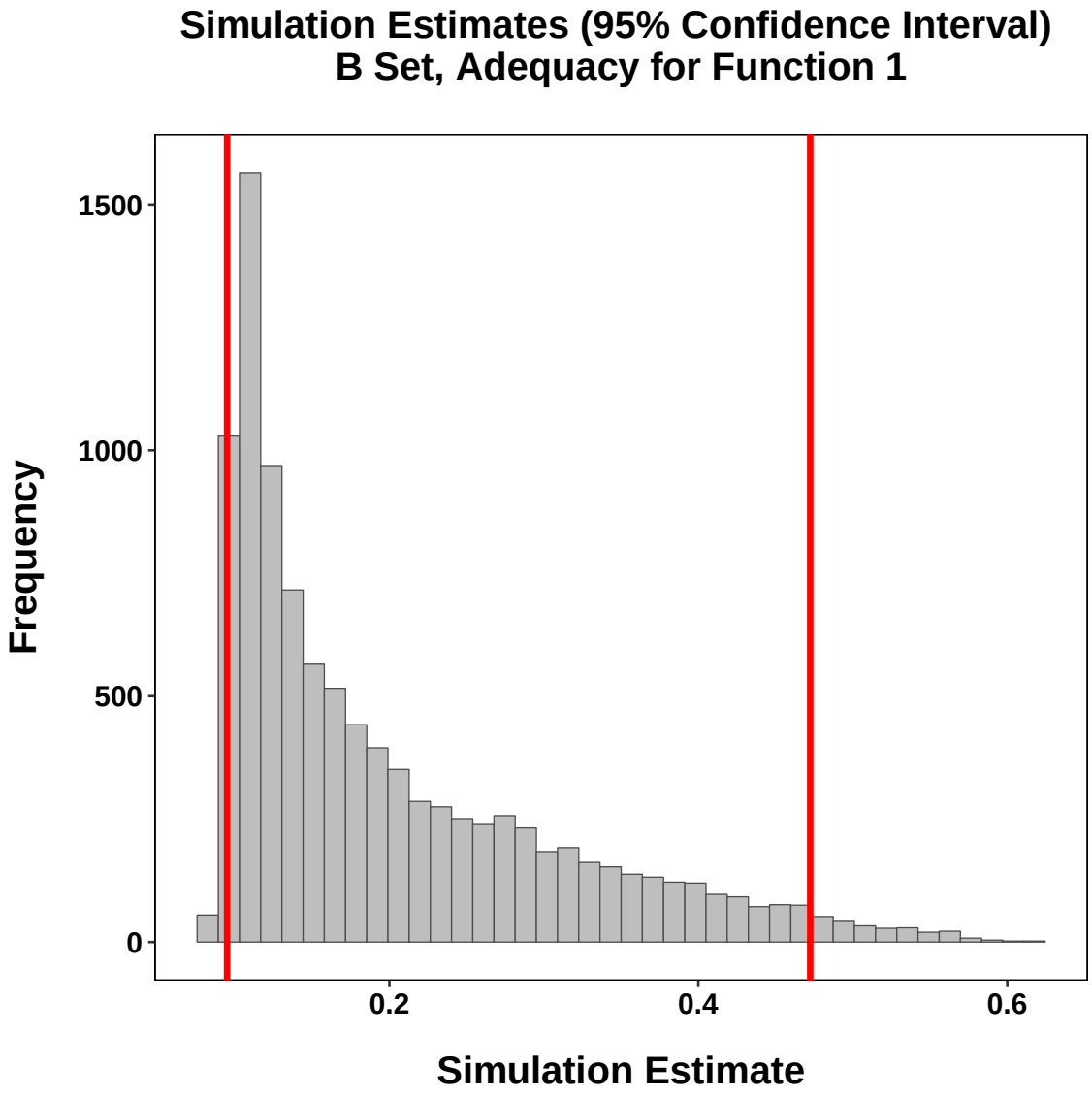

Simulation Estimates (95% Confidence Interval)
A Set, Loading A1 on Function 1
Frequency
400
200
0
−1.0
−0.5
0.0
0.5
1.0
Simulation Estimate

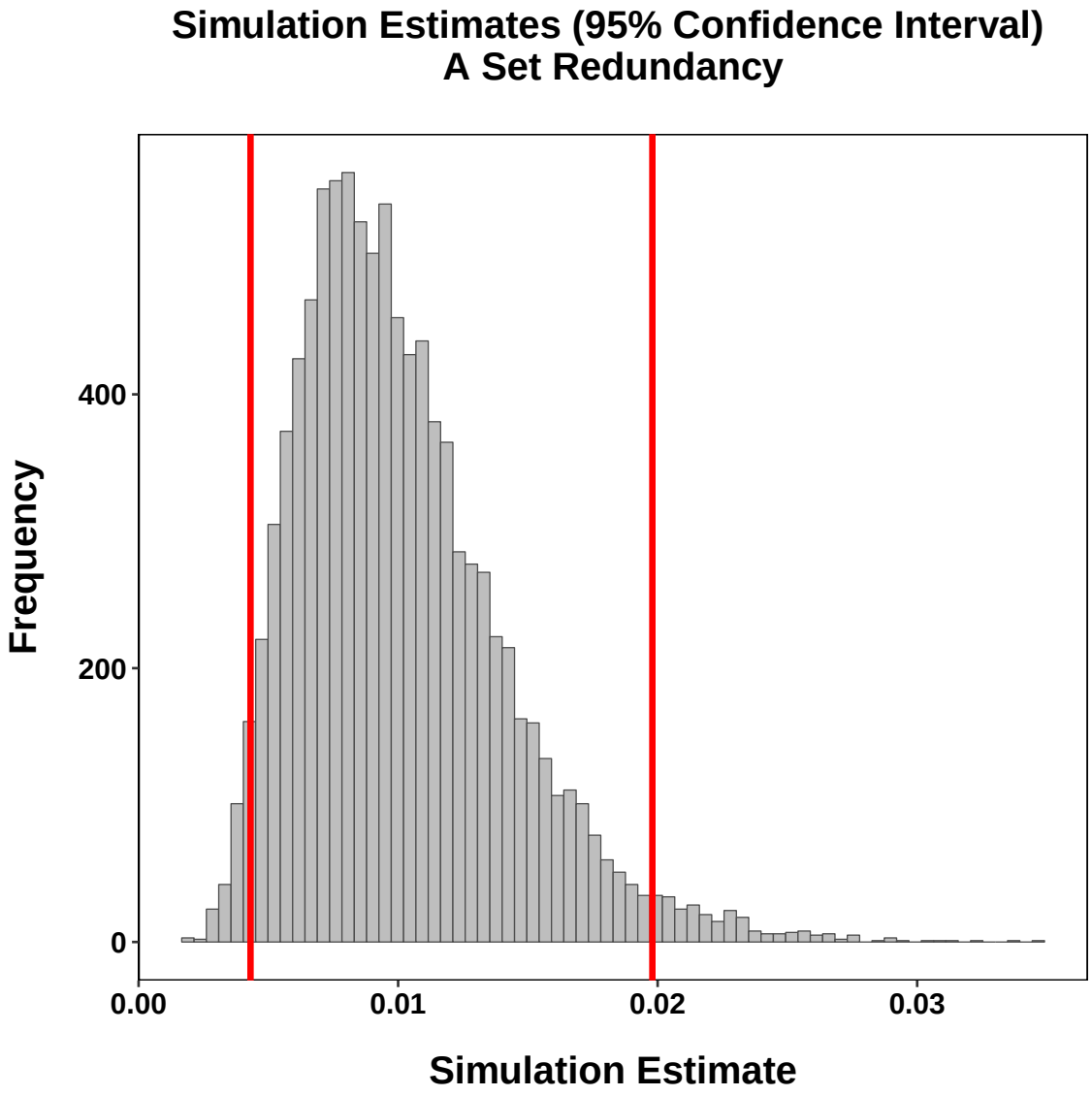

Simulation Estimates (95% Confidence Interval)
B Set, Loading B1 on Function 1
Frequency
400
200
0
−1.0
−0.5
0.0
0.5
1.0
Simulation Estimate

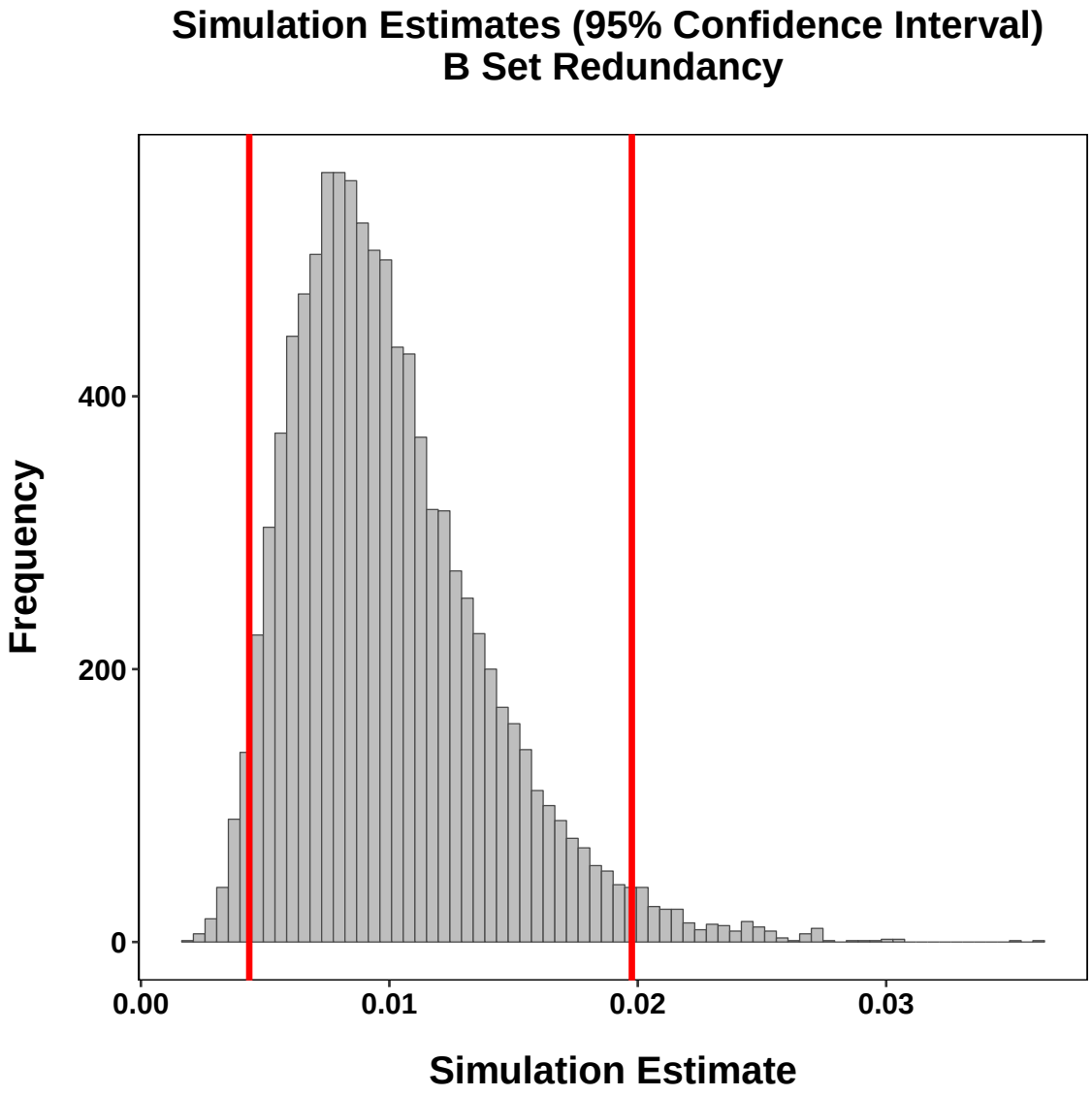

Simulation Estimates (95% Confidence Interval)
A Set, Weight A1 on Function 1
Frequency
500
400
300
200
100
0
−1.0
−0.5
0.0
0.5
1.0
1.5
Simulation Estimate


Simulation Estimates (95% Confidence Interval)
B Set, Weight B1 on Function 1
Frequency
400
200
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
Simulation Estimate


Simulation Estimates (95% Confidence Interval)
A Set, Adequacy for Function 1
Frequency
1500
1000
500
0
0.2
0.4
0.6
Simulation Estimate


Simulation Estimates (95% Confidence Interval)
B Set, Adequacy for Function 1
Frequency
1500
1000
500
0
0.2
0.4
0.6
Simulation Estimate


Simulation Estimates (95% Confidence Interval)
A Set Redundancy
Frequency
400
200
0
0.00
0.01
0.02
0.03
Simulation Estimate



Tally false rejections for canonical correlations. Because the tests are sequential, the tally from the previous step becomes the denominator for the current step when calculating the proportion of rejections.

```
k1 <- sum(CCA_Sim_Results$p1 < 0.05)
k2 <- sum(CCA_Sim_Results$p2 < 0.05)
k3 <- sum(CCA_Sim_Results$p3 < 0.05)
k4 <- sum(CCA_Sim_Results$p4 < 0.05)
k5 <- sum(CCA_Sim_Results$p5 < 0.05)
prop_1 <- prop.test(k1, 10000, correct = FALSE)
prop_2 <- prop.test(k2, k1, correct = FALSE)
prop_3 <- prop.test(k3, k2, correct = FALSE)
prop_4 <- prop.test(k4, k3, correct = FALSE)

## Warning in prop.test(k4, k3, correct = FALSE): Chi-squared approximation may be incorrect
```

```r
prop_5 <- prop.test(k5, k4, correct = FALSE)
```

```
## Error in prop.test(k5, k4, correct = FALSE): elements of 'n' must be positive
```

Null Rejection Summary			
Canonical Correlation	p	Lower 95% CL	Upper 95% CL
1	0.0468	0.0428	0.0511
2	0.0256	0.0147	0.0443
3	0.0833	0.0149	0.3539
4	0	0	0.7935
5	-	-	-

3 Cross-Validation

The most convincing approach to cross-validation requires a calibration sample and a separate hold-out sample. The calibration sample is analyzed and the canonical coefficients derived. Those coefficients are then applied to the hold-out sample. A standard canonical correlation analysis is also conducted on the hold-out sample. The correlations among the actual and estimated canonical variates are computed.

A sample of 400 participants completed the NEO (The Big Five, Set 1) and measures of self-esteem, optimism, life satisfaction, and happiness (Set 2). The sample was split randomly to produce a calibration sample and a hold-out sample. A canonical correlation analysis was conducted on the calibration sample to determine the nature of the relations between sets and to obtain standardized canonical coefficients to use in the hold-out sample.

3.1 Calibration Sample

```
CCA_Calibration <- cancor(cbind(SE, Opt, Life_Sat, Happy) ~ NEO_N +
    NEO_E + NEO_O + NEO_A + NEO_C, data = Calibration, prefix = c("P_",
    "WB_"), set.names = c("Personality", "Well-Being"))
Calibration_Coef_Personality <- as.matrix(coef(CCA_Calibration, standardization = TRUE,
    type = "both")[[1]])
Calibration_Coef_Well_Being <- as.matrix(coef(CCA_Calibration, standardization = TRUE,
    type = "both")[[2]])

CCA_Calibration

##
## Canonical correlation analysis of:
##   5   Personality  variables:  NEO_N, NEO_E, NEO_O, NEO_A, NEO_C
##   with  4   Well-Being  variables:  SE, Opt, Life_Sat, Happy
##
##     CanR  CanRSQ   Eigen percent    cum
## 1 0.8768 0.76882 3.32566 68.9370  68.94
## 2 0.7190 0.51692 1.07003 22.1806  91.12
## 3 0.5347 0.28585 0.40027  8.2972  99.41
## 4 0.1657 0.02746 0.02823  0.5852 100.00
##                       scree
## 1 ************************
## 2 ********
## 3 ***
## 4
##
## Test of H0: The canonical correlations in the
## current row and all that follow are zero
##
##    CanR LR test stat approx F numDF denDF Pr(> F)
## 1 0.877        0.078     36.8    20   634 < 2e-16
## 2 0.719        0.336     21.6    12   508 < 2e-16
## 3 0.535        0.695     12.9     6   386 2.9e-13
## 4 0.166        0.973              2

CCA_Calibration$coef
```

```
## $X
##            P_1      P_2     P_3       P_4
## NEO_N -0.22308 -0.80648  0.8251 -0.393287
## NEO_E  0.17813 -0.49624 -0.1667 -0.007479
## NEO_O  0.63299  0.30450  0.3715  0.714660
## NEO_A  0.04131 -0.91893 -0.1351  0.041384
## NEO_C  0.73568  0.02147  0.3811 -0.689296
##
## $Y
##             WB_1    WB_2    WB_3    WB_4
## SE        0.1773  0.5026 -0.6913 -0.5772
## Opt       0.6268  0.5786  0.3356  0.7041
## Life_Sat  0.5725 -0.5111  0.4903 -0.8200
## Happy    -0.1364 -0.6474 -0.8329  0.6122
```

```
CCA_Calibration$structure
```

```
## $X.xscores
##           P_1      P_2      P_3       P_4
## NEO_N -0.4230 -0.23239  0.86784  0.005039
## NEO_E  0.3182 -0.46109 -0.39102  0.009468
## NEO_O  0.4602 -0.01180  0.47155  0.752108
## NEO_A  0.3692 -0.63714 -0.39483  0.224249
## NEO_C  0.7372  0.08711 -0.02572 -0.660487
##
## $Y.xscores
##             P_1     P_2       P_3      P_4
## SE       0.3649  0.3114 -0.371655 -0.06546
## Opt      0.7516  0.1707  0.005828  0.07571
## Life_Sat 0.6930 -0.3821  0.034670 -0.04933
## Happy    0.4079 -0.4146 -0.310679  0.05580
##
## $X.yscores
##          WB_1      WB_2     WB_3      WB_4
## NEO_N -0.3709 -0.167085  0.46399  0.000835
## NEO_E  0.2790 -0.331511 -0.20906  0.001569
## NEO_O  0.4036 -0.008485  0.25211  0.124627
## NEO_A  0.3237 -0.458086 -0.21110  0.037159
## NEO_C  0.6464  0.062629 -0.01375 -0.109445
##
## $Y.yscores
##            WB_1    WB_2     WB_3    WB_4
## SE       0.4162  0.4331 -0.69513 -0.3950
## Opt      0.8572  0.2375  0.01090  0.4569
## Life_Sat 0.7904 -0.5315  0.06485 -0.2977
## Happy    0.4652 -0.5766 -0.58109  0.3367
```

```
coef(CCA_Calibration, standardize = TRUE, type = "both")
```

```
## [[1]]
##            P_1      P_2     P_3       P_4
## NEO_N -0.22308 -0.80648  0.8251 -0.393287
## NEO_E  0.17813 -0.49624 -0.1667 -0.007479
## NEO_O  0.63299  0.30450  0.3715  0.714660
## NEO_A  0.04131 -0.91893 -0.1351  0.041384
```

```
## NEO_C  0.73568  0.02147  0.3811 -0.689296
##
## [[2]]
##             WB_1    WB_2    WB_3    WB_4
## SE        0.1773  0.5026 -0.6913 -0.5772
## Opt       0.6268  0.5786  0.3356  0.7041
## Life_Sat  0.5725 -0.5111  0.4903 -0.8200
## Happy    -0.1364 -0.6474 -0.8329  0.6122
```

```
p.asym(CCA_Calibration$cancor, 400, 5, 4, tstat = "Wilks")
```

```
## Wilks' Lambda, using F-approximation (Rao's F):
##             stat approx df1  df2  p.value
## 1 to 4:  0.07757 75.374  20 1298 0.000000
## 2 to 4:  0.33552 44.176  12 1037 0.000000
## 3 to 4:  0.69454 26.189   6  786 0.000000
## 4 to 4:  0.97254  5.562   2  394 0.004149
```

3.2 Hold-Out Sample

```
CCA_Hold_Out <- cancor(cbind(SE, Opt, Life_Sat, Happy) ~ NEO_N + NEO_E +
    NEO_O + NEO_A + NEO_C, data = Hold_Out, prefix = c("P_", "WB_"),
    set.names = c("Personality", "Well-Being"))
Hold_Out_Coef_Personality <- as.matrix(coef(CCA_Hold_Out, standardization = TRUE,
    type = "both")[[1]])
Hold_Out_Coef_Well_Being <- as.matrix(coef(CCA_Hold_Out, standardization = TRUE,
    type = "both")[[2]])
```

```
CCA_Hold_Out
```

```
##
## Canonical correlation analysis of:
##   5   Personality  variables:  NEO_N, NEO_E, NEO_O, NEO_A, NEO_C
##   with  4   Well-Being  variables:  SE, Opt, Life_Sat, Happy
##
##     CanR  CanRSQ  Eigen percent    cum                     scree
## 1 0.8798 0.77400 3.4248 77.2217  77.22 ************************
## 2 0.6639 0.44080 0.7883 17.7736  95.00 ******
## 3 0.4033 0.16266 0.1943  4.3801  99.38 *
## 4 0.1642 0.02696 0.0277  0.6247 100.00
##
## Test of H0: The canonical correlations in the
## current row and all that follow are zero
##
##    CanR LR test stat approx F numDF denDF    Pr(> F)
## 1 0.880        0.103    31.23    20   634    < 2e-16
## 2 0.664        0.456    14.65    12   508    < 2e-16
## 3 0.403        0.815     6.94     6   386 0.00000051
## 4 0.164        0.973                2
```

```
CCA_Hold_Out$coef
```

```
## $X
```

```
##            P_1     P_2     P_3      P_4
## NEO_N -0.30058 -0.9409 -0.5520  0.04721
## NEO_E  0.03961 -0.6364  0.3559  0.09831
## NEO_O  0.68886  0.1738 -0.4431 -0.65566
## NEO_A -0.02042 -0.4893  0.4146 -0.46377
## NEO_C  0.75365 -0.2973 -0.3323  0.57991
##
## $Y
##             WB_1    WB_2    WB_3    WB_4
## SE        0.1348  0.4767  0.7092  0.6649
## Opt       0.7368  0.4725 -0.5487 -0.5461
## Life_Sat  0.5687 -0.8216 -0.2701  0.5572
## Happy    -0.2684 -0.2184  0.9092 -0.7245
```

```
CCA_Hold_Out$structure
```

```
## $X.xscores
##           P_1      P_2      P_3      P_4
## NEO_N -0.4496 -0.48512 -0.74310 -0.09950
## NEO_E  0.3164 -0.48548  0.48060 -0.09891
## NEO_O  0.4845 -0.00105 -0.38737 -0.74646
## NEO_A  0.2036 -0.37225  0.63599 -0.33328
## NEO_C  0.6936 -0.17710  0.04967  0.63877
##
## $Y.xscores
##             P_1     P_2      P_3      P_4
## SE       0.4342  0.2940 0.243684  0.07256
## Opt      0.7542  0.2099 0.002149 -0.06670
## Life_Sat 0.6068 -0.4537 0.069398  0.02734
## Happy    0.2967 -0.2374 0.275414 -0.08874
##
## $X.yscores
##          WB_1       WB_2     WB_3     WB_4
## NEO_N -0.3955 -0.3220865 -0.29970 -0.01634
## NEO_E  0.2784 -0.3223218  0.19383 -0.01624
## NEO_O  0.4263 -0.0006974 -0.15623 -0.12256
## NEO_A  0.1791 -0.2471452  0.25650 -0.05472
## NEO_C  0.6102 -0.1175795  0.02003  0.10488
##
## $Y.yscores
##            WB_1    WB_2     WB_3    WB_4
## SE       0.4936  0.4428 0.604207  0.4419
## Opt      0.8573  0.3162 0.005328 -0.4062
## Life_Sat 0.6897 -0.6834 0.172070  0.1665
## Happy    0.3373 -0.3575 0.682882 -0.5405
```

```
coef(CCA_Hold_Out, standardize = TRUE, type = "both")
```

```
## [[1]]
##            P_1     P_2     P_3      P_4
## NEO_N -0.30058 -0.9409 -0.5520  0.04721
## NEO_E  0.03961 -0.6364  0.3559  0.09831
## NEO_O  0.68886  0.1738 -0.4431 -0.65566
## NEO_A -0.02042 -0.4893  0.4146 -0.46377
## NEO_C  0.75365 -0.2973 -0.3323  0.57991
```

```
##
## [[2]]
##             WB_1    WB_2    WB_3    WB_4
## SE        0.1348  0.4767  0.7092  0.6649
## Opt       0.7368  0.4725 -0.5487 -0.5461
## Life_Sat  0.5687 -0.8216 -0.2701  0.5572
## Happy    -0.2684 -0.2184  0.9092 -0.7245
```

3.3 Calculate Predicted Scores Across Samples

3.3.1 Predict Scores in the Hold-Out Sample

```
Est_Personality_Hold_Out <- as.matrix(Hold_Out[, 3:7]) %*% Calibration_Coef_Personality
Est_Well_Being_Hold_Out <- as.matrix(Hold_Out[, 8:11]) %*% Calibration_Coef_Well_Being
Act_Personality_Hold_Out <- CCA_Hold_Out$scores$X
Act_Well_Being_Hold_Out <- CCA_Hold_Out$scores$Y
Hold_Out_Scores <- as.data.frame(cbind(Act_Personality_Hold_Out, Act_Well_Being_Hold_Out,
    Est_Personality_Hold_Out, Est_Well_Being_Hold_Out))
names(Hold_Out_Scores) <- c("A_P_1", "A_P_2", "A_P_3", "A_P_4", "A_WB_1",
    "A_WB_2", "A_WB_3", "A_WB_4", "E_P_1", "E_P_2", "E_P_3", "E_P_4",
    "E_WB_1", "E_WB_2", "E_WB_3", "E_WB_4")
```

3.3.2 Predict Scores in the Calibration Sample

```
Est_Personality_Calibration <- as.matrix(Calibration[, 3:7]) %*% Hold_Out_Coef_Personality
Est_Well_Being_Calibration <- as.matrix(Calibration[, 8:11]) %*% Hold_Out_Coef_Well_Being
Act_Personality_Calibration <- CCA_Calibration$scores$X
Act_Well_Being_Calibration <- CCA_Calibration$scores$Y
Calibration_Scores <- as.data.frame(cbind(Act_Personality_Calibration,
    Act_Well_Being_Calibration, Est_Personality_Calibration, Est_Well_Being_Calibration))
names(Calibration_Scores) <- c("A_P_1", "A_P_2", "A_P_3", "A_P_4",
    "A_WB_1", "A_WB_2", "A_WB_3", "A_WB_4", "E_P_1", "E_P_2", "E_P_3",
    "E_P_4", "E_WB_1", "E_WB_2", "E_WB_3", "E_WB_4")
```

3.4 Correlations Within Samples

3.4.1 Calibration Sample

```
round(cor(Calibration_Scores[, c(1:4, 9:12)]), 3)
```

```
##        A_P_1  A_P_2  A_P_3  A_P_4  E_P_1  E_P_2  E_P_3  E_P_4
## A_P_1  1.000  0.000  0.000  0.000  0.989 -0.134  0.049 -0.032
## A_P_2  0.000  1.000  0.000  0.000  0.120  0.857 -0.312  0.280
## A_P_3  0.000  0.000  1.000  0.000  0.037 -0.307 -0.948 -0.130
## A_P_4  0.000  0.000  0.000  1.000  0.014  0.223 -0.019 -0.921
## E_P_1  0.989  0.120  0.037  0.014  1.000 -0.012 -0.026 -0.035
## E_P_2 -0.134  0.857 -0.307  0.223 -0.012  1.000  0.004  0.004
## E_P_3  0.049 -0.312 -0.948 -0.019 -0.026  0.004  1.000  0.058
## E_P_4 -0.032  0.280 -0.130 -0.921 -0.035  0.004  0.058  1.000
```

```
round(cor(Calibration_Scores[, c(5:8, 13:16)]), 3)
```

```
##        A_WB_1 A_WB_2 A_WB_3 A_WB_4 E_WB_1 E_WB_2 E_WB_3 E_WB_4
## A_WB_1  1.000  0.000  0.000  0.000  0.991 -0.155  0.032 -0.091
## A_WB_2  0.000  1.000  0.000  0.000  0.084  0.928 -0.190  0.289
## A_WB_3  0.000  0.000  1.000  0.000  0.105 -0.266 -0.972 -0.011
## A_WB_4  0.000  0.000  0.000  1.000  0.023  0.209 -0.134 -0.953
## E_WB_1  0.991  0.084  0.105  0.023  1.000 -0.099 -0.089 -0.089
## E_WB_2 -0.155  0.928 -0.266  0.209 -0.099  1.000  0.049  0.086
## E_WB_3  0.032 -0.190 -0.972 -0.134 -0.089  0.049  1.000  0.081
## E_WB_4 -0.091  0.289 -0.011 -0.953 -0.089  0.086  0.081  1.000
```

```
round(cor(Calibration_Scores[, c(1:8)]), 3)
```

```
##        A_P_1 A_P_2 A_P_3 A_P_4 A_WB_1 A_WB_2 A_WB_3 A_WB_4
## A_P_1  1.000 0.000 0.000 0.000  0.877  0.000  0.000  0.000
## A_P_2  0.000 1.000 0.000 0.000  0.000  0.719  0.000  0.000
## A_P_3  0.000 0.000 1.000 0.000  0.000  0.000  0.535  0.000
## A_P_4  0.000 0.000 0.000 1.000  0.000  0.000  0.000  0.166
## A_WB_1 0.877 0.000 0.000 0.000  1.000  0.000  0.000  0.000
## A_WB_2 0.000 0.719 0.000 0.000  0.000  1.000  0.000  0.000
## A_WB_3 0.000 0.000 0.535 0.000  0.000  0.000  1.000  0.000
## A_WB_4 0.000 0.000 0.000 0.166  0.000  0.000  0.000  1.000
```

```
round(cor(Calibration_Scores[, c(9:16)]), 3)
```

```
##         E_P_1  E_P_2  E_P_3  E_P_4 E_WB_1 E_WB_2 E_WB_3 E_WB_4
## E_P_1   1.000 -0.012 -0.026 -0.035  0.868 -0.059 -0.008 -0.056
## E_P_2  -0.012  1.000  0.004  0.004 -0.081  0.642  0.034  0.156
## E_P_3  -0.026  0.004  1.000  0.058 -0.029 -0.081  0.537 -0.060
## E_P_4  -0.035  0.004  0.058  1.000 -0.022  0.178  0.049  0.207
## E_WB_1  0.868 -0.081 -0.029 -0.022  1.000 -0.099 -0.089 -0.089
## E_WB_2 -0.059  0.642 -0.081  0.178 -0.099  1.000  0.049  0.086
## E_WB_3 -0.008  0.034  0.537  0.049 -0.089  0.049  1.000  0.081
## E_WB_4 -0.056  0.156 -0.060  0.207 -0.089  0.086  0.081  1.000
```

3.4.2 Hold-Out Sample

```
round(cor(Hold_Out_Scores[, c(1:4, 9:12)]), 3)
```

```
##        A_P_1  A_P_2  A_P_3  A_P_4  E_P_1  E_P_2  E_P_3  E_P_4
## A_P_1  1.000  0.000  0.000  0.000  0.989  0.164 -0.007  0.048
## A_P_2  0.000  1.000  0.000  0.000 -0.125  0.882 -0.348  0.285
## A_P_3  0.000  0.000  1.000  0.000  0.069 -0.309 -0.933  0.004
## A_P_4  0.000  0.000  0.000  1.000 -0.012  0.202 -0.056 -0.900
## E_P_1  0.989 -0.125  0.069 -0.012  1.000  0.039 -0.031  0.038
## E_P_2  0.164  0.882 -0.309  0.202  0.039  1.000 -0.047  0.156
## E_P_3 -0.007 -0.348 -0.933 -0.056 -0.031 -0.047  1.000 -0.075
## E_P_4  0.048  0.285  0.004 -0.900  0.038  0.156 -0.075  1.000
```

```
round(cor(Hold_Out_Scores[, c(5:8, 13:16)]), 3)
```

```
##        A_WB_1 A_WB_2 A_WB_3 A_WB_4 E_WB_1 E_WB_2 E_WB_3 E_WB_4
```

```
## A_WB_1  1.000  0.000  0.000  0.000  0.991  0.164  0.004 -0.038
## A_WB_2  0.000  1.000  0.000  0.000 -0.067  0.934 -0.254  0.292
## A_WB_3  0.000  0.000  1.000  0.000  0.118 -0.211 -0.963 -0.064
## A_WB_4  0.000  0.000  0.000  1.000 -0.007  0.238  0.096 -0.954
## E_WB_1  0.991 -0.067  0.118 -0.007  1.000  0.073 -0.093 -0.058
## E_WB_2  0.164  0.934 -0.211  0.238  0.073  1.000 -0.010  0.052
## E_WB_3  0.004 -0.254 -0.963  0.096 -0.093 -0.010  1.000 -0.104
## E_WB_4 -0.038  0.292 -0.064 -0.954 -0.058  0.052 -0.104  1.000
```

```
round(cor(Hold_Out_Scores[, c(1:8)]), 3)
```

```
##        A_P_1 A_P_2 A_P_3 A_P_4 A_WB_1 A_WB_2 A_WB_3 A_WB_4
## A_P_1   1.00 0.000 0.000 0.000   0.88  0.000  0.000  0.000
## A_P_2   0.00 1.000 0.000 0.000   0.00  0.664  0.000  0.000
## A_P_3   0.00 0.000 1.000 0.000   0.00  0.000  0.403  0.000
## A_P_4   0.00 0.000 0.000 1.000   0.00  0.000  0.000  0.164
## A_WB_1  0.88 0.000 0.000 0.000   1.00  0.000  0.000  0.000
## A_WB_2  0.00 0.664 0.000 0.000   0.00  1.000  0.000  0.000
## A_WB_3  0.00 0.000 0.403 0.000   0.00  0.000  1.000  0.000
## A_WB_4  0.00 0.000 0.000 0.164   0.00  0.000  0.000  1.000
```

```
round(cor(Hold_Out_Scores[, c(9:16)]), 3)
```

```
##         E_P_1  E_P_2  E_P_3  E_P_4 E_WB_1 E_WB_2 E_WB_3 E_WB_4
## E_P_1   1.000  0.039 -0.031  0.038  0.871  0.059 -0.003 -0.057
## E_P_2   0.039  1.000 -0.047  0.156  0.089  0.604 -0.025  0.142
## E_P_3  -0.031 -0.047  1.000 -0.075 -0.035 -0.139  0.420 -0.034
## E_P_4   0.038  0.156 -0.075  1.000  0.031  0.148 -0.063  0.194
## E_WB_1  0.871  0.089 -0.035  0.031  1.000  0.073 -0.093 -0.058
## E_WB_2  0.059  0.604 -0.139  0.148  0.073  1.000 -0.010  0.052
## E_WB_3 -0.003 -0.025  0.420 -0.063 -0.093 -0.010  1.000 -0.104
## E_WB_4 -0.057  0.142 -0.034  0.194 -0.058  0.052 -0.104  1.000
```