Discriminant Analysis II

Mike Strube

November 8, 2018

1 Preliminaries

In this section, the RStudio workspace and console panes are cleared of old output, variables, and other miscellaneous debris. Packages are loaded and any required data files are retrieved.

```
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)
library(ggplot2)

## Warning:  package 'ggplot2' was built under R version 3.4.4
##
## Attaching package:  'ggplot2'
## The following objects are masked from 'package:psych':
##
##    %+%, alpha

library(MASS)
library(sciplot)
library(plyr)
library(dawai)

## Warning:  package 'dawai' was built under R version 3.4.4
## Loading required package:  mvtnorm
## Loading required package:  ibdreg
## Loading required package:  boot
##
## Attaching package:  'boot'
```

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

```
library(candisc)
```

```
## Loading required package: car
##
## Attaching package: 'car'
## The following object is masked from 'package:boot':
##
##     logit
## The following object is masked from 'package:psych':
##
##     logit
## Loading required package: heplots
##
## Attaching package: 'candisc'
## The following object is masked from 'package:stats':
##
##     cancor
```

```
library(biotools)
```

```
## Loading required package: rpanel
## Loading required package: tcltk
## Package 'rpanel', version 1.1-3: type help(rpanel) for summary information
##
## Attaching package: 'rpanel'
## The following object is masked from 'package:boot':
##
##     poisons
## Loading required package: tkrplot
## Warning: loading Rplot failed
## Loading required package: lattice
##
## Attaching package: 'lattice'
## The following object is masked from 'package:boot':
##
##     melanoma
## Loading required package: SpatialEpi
## Loading required package: sp
```

```
## ---
## biotools version 3.1
```

```
##
##
## Attaching package: 'biotools'
## The following object is masked from 'package:heplots':
##
##     boxM
```

```
library(DiscriMiner)
```

```
## Warning: package 'DiscriMiner' was built under R version 3.4.4
```

```
library(ade4)
```

```
## Warning:  package 'ade4' was built under R version 3.4.4
```

```
library(MVN)
```

```
## Warning:  package 'MVN' was built under R version 3.4.4
## sROC 0.1-2 loaded
```

```
library(biotools)
library(klaR)
```

```
## Warning:  package 'klaR' was built under R version 3.4.4
```

```
library(GGally)
```

```
## Warning:  package 'GGally' was built under R version 3.4.4
```

```
library(reshape2)
library(MVN)
library(qqplotr)
```

```
## Warning:  package 'qqplotr' was built under R version 3.4.4
##
## Attaching package:  'qqplotr'
## The following objects are masked from 'package:ggplot2':
##
##    stat_qq_line, StatQqLine
```

```
library(gridExtra)
library(caret)
```

```
## Error in library(caret):  there is no package called 'caret'
```

1.1 Data

In this hypothetical example, data from 500 graduate students seeking jobs were examined. Available for each student were three predictors: GRE(V+Q), Years to Finish the Degree, and Number of Publications. The outcome measure was categorical: "Got a job" versus "Did not get a job."

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

Job <- read.table("get_a_job.csv", sep = ",", header = TRUE)
Job <- as.data.frame(Job)
Job$job_num <- Job$job
Job$job[Job$job == "1"] <- "No Job"
Job$job[Job$job == "2"] <- "Job"
# Residuals
Job$gre_R <- lm(gre ~ as.factor(job), data = Job)$residuals
Job$pubs_R <- lm(pubs ~ as.factor(job), data = Job)$residuals
Job$years_R <- lm(years ~ as.factor(job), data = Job)$residuals
```

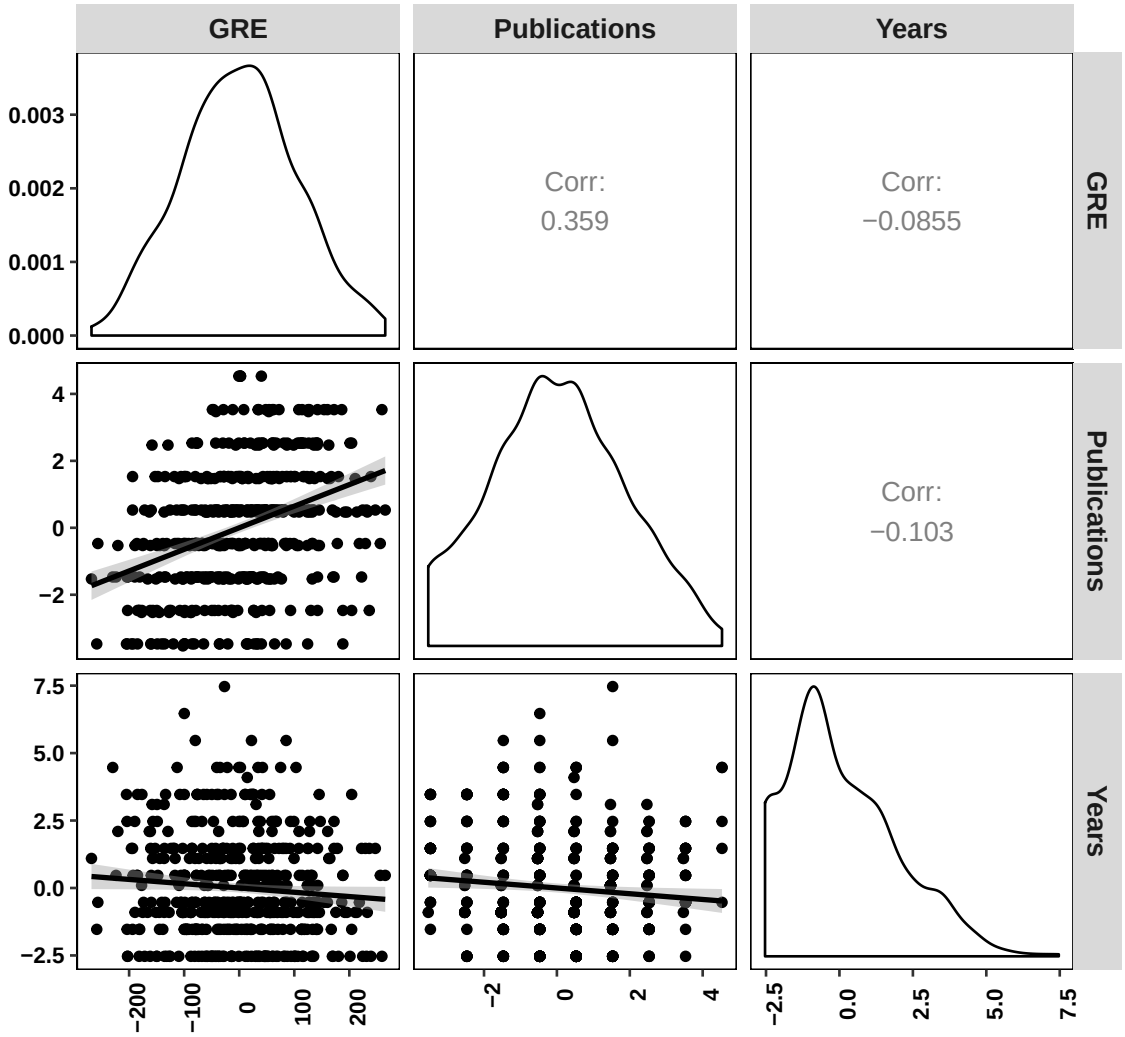
2 Job Search Data

These hypothetical data simulate the factors that might contribute to successfully getting an academic job.

2.1 Basic Visualization

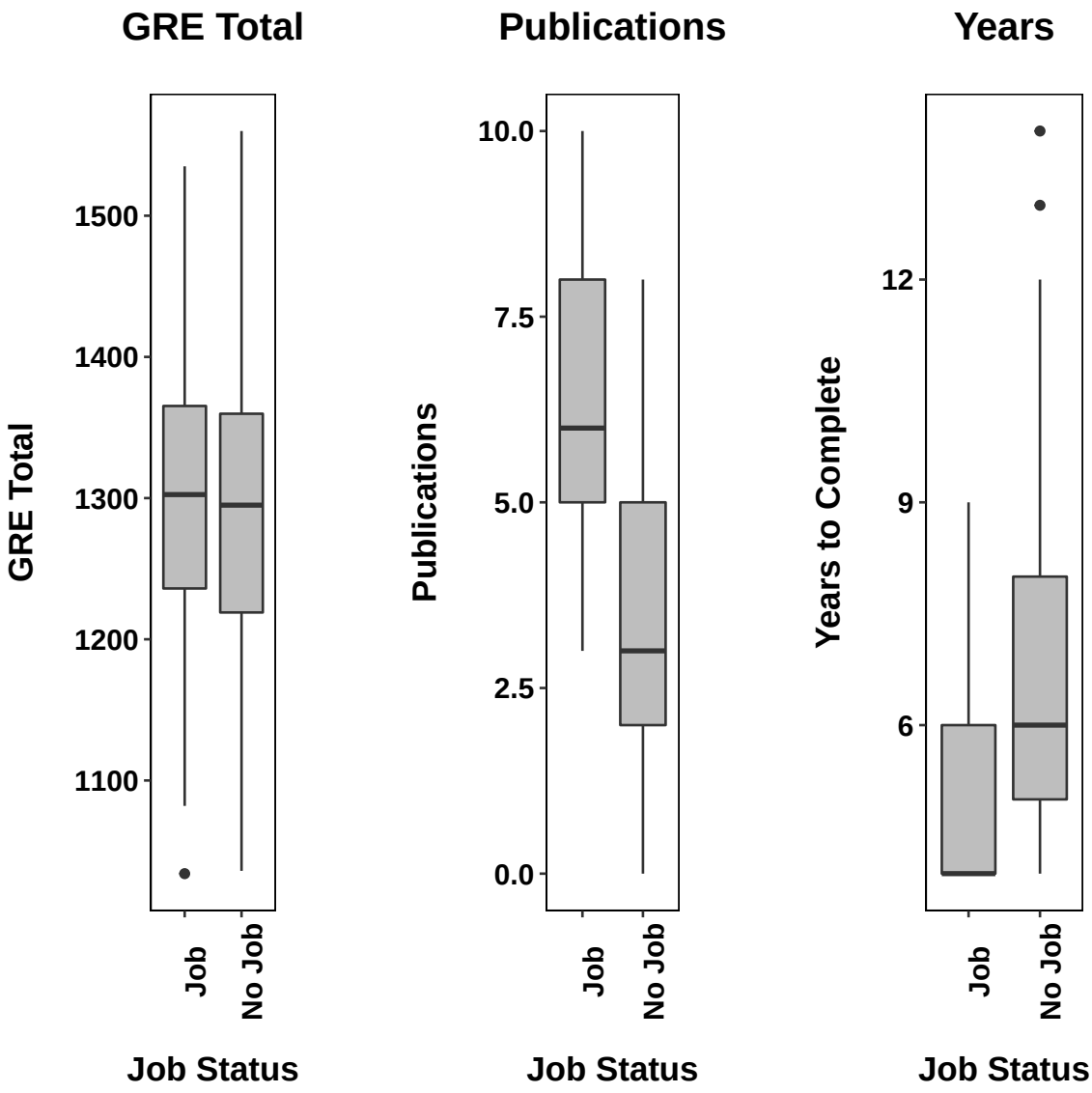
The basic nature of the data is easily viewed with some simple graphics.

```
ggpairs(Job[7:9], lower = list(continuous = "smooth"), upper = list(continuous = "cor"),
    columnLabels = c("GRE", "Publications", "Years")) + 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("Correlations Among Job Search Features (Residuals)")
```




```
p1 <- ggplot(Job, aes(x = job, y = gre)) + geom_boxplot(fill = "gray") +
    ylab("GRE Total") + xlab("Job Status") + 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, face = "bold", angle = 90), axis.title.x = element_text(margin = margin(15,
    0, 0, 0), size = 14), axis.title.y = element_text(margin = margin(0,
    15, 0, 0), size = 14), 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("GRE Total")
```

```
p2 <- ggplot(Job, aes(x = job, y = pubs)) + geom_boxplot(fill = "gray") +
    ylab("Publications") + xlab("Job Status") + 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, face = "bold", angle = 90), axis.title.x = element_text(margin = margin(15,
    0, 0, 0), size = 14), axis.title.y = element_text(margin = margin(0,
    15, 0, 0), size = 14), 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("Publications")

p3 <- ggplot(Job, aes(x = job, y = years)) + geom_boxplot(fill = "gray") +
    ylab("Years to Complete") + xlab("Job Status") + 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, face = "bold", angle = 90), axis.title.x = element_text(margin = margin(15,
    0, 0, 0), size = 14), axis.title.y = element_text(margin = margin(0,
    15, 0, 0), size = 14), 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("Years")
grid.arrange(p1, p2, p3, nrow = 1)
```



2.2 Group Differences

A univariate look at the data will provide some clues about likely variables of influence in the discriminant analysis.

```
Job_MANOVA <- manova(as.matrix(Job[, 1:3]) ~ as.factor(Job$job))
Job_Wilks <- summary(Job_MANOVA, test = c("Wilks"))
summary(Job_MANOVA)

##                      Df Pillai approx F num Df den Df Pr(>F)
## as.factor(Job$job)    1   0.41      115      3    496 <2e-16
## Residuals           498

summary.aov(Job_MANOVA)
```

```
##  Response gre :
##                      Df  Sum Sq Mean Sq F value Pr(>F)
## as.factor(Job$job)    1    5693    5693    0.53   0.47
## Residuals           498 5362834   10769
##
##  Response pubs :
##                      Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(Job$job)    1    927     927     266 <2e-16
## Residuals           498   1733       3
##
##  Response years :
##                      Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(Job$job)    1    263   262.6    70.9  4e-16
## Residuals           498   1844     3.7
```

2.3 Basic Function

There are several packages that provide discriminant analysis, but they vary in the specific results they can produce. The two most common functions are lda( ) from the MASS package and candisc( ) from the candisc package. They are illustrated here; some others that provide a few additional useful results are shown later.

```
Job_LDA <- lda(Job$job_num ~ gre + pubs + years, data = Job)
```

Note that candisc( ) uses a multivariate linear model object from the lm( ) function.

```
Job_MLM <- lm(cbind(gre, pubs, years) ~ as.factor(job), data = Job)
Job_CDA <- candisc(Job_MLM, data = Job)
```

2.4 Standardized and Unstandardized Discriminant Function

The nature of the functions produced by lda( ) is a source of considerable confusion. The documentation describes them as standardized coefficients that would be applied to the centered but not standardized variables. That would usually be the definition for unstandardized weights and the call for raw coefficients from candisc( ) confirms that interpretation. The candisc( ) also produces true standardized coefficients that allow determining the relative contribution of the variables to group discrimination, even if the variables are on different scales. The candisc( ) function produces the structure matrix as well representing the correlations between the original discriminating variables and the discriminant functions. Note a potential problem if you use several packages on the same data. They may use slightly different estimation algorithms and even if they do not, they may reflect some functions (reversing the signs).

```
Job_LDA$scaling

##             LD1
## gre   -0.003253
## pubs   0.513837
## years -0.198684

Job_CDA$coeffs.raw
```

```
##            Can1
## gre   -0.003253
## pubs   0.513837
## years -0.198684

Job_CDA$coeffs.std

##          Can1
## gre   -0.3375
## pubs   0.9584
## years -0.3824

Job_CDA$structure

##           Can1
## gre    0.05083
## pubs   0.92155
## years -0.55108
```

2.5 lda( ) method options

The lda( ) function has a method option that provides several varieties of discriminant analysis. The "moment" option provides the standard analysis. The "mle" option provides maximum likelihood estimates. The "mve" option provides a robust solution that estimates the means and covariance matrices based on a subsample of the data that is relatively free of extreme data points. The "t" option also provides robust estimates that are based on a t distribution. These are illustrated here.

```
Job_LDA_1 <- lda(Job$job_num ~ gre + pubs + years, method = "mle",
    data = Job)
Job_LDA_1$scaling

##             LD1
## gre   -0.003259
## pubs   0.514867
## years -0.199082

Job_LDA_2 <- lda(Job$job_num ~ gre + pubs + years, method = "mve",
    data = Job)
Job_LDA_2$scaling

##             LD1
## gre   -0.004191
## pubs   0.541802
## years -0.199068

Job_LDA_3 <- lda(Job$job_num ~ gre + pubs + years, method = "t", data = Job)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.323   0.662   0.765   0.753   0.857   0.989
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.229   0.558   0.686   0.679   0.806   0.982
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.199   0.518   0.655   0.649   0.783   0.982
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.187   0.499   0.642   0.636   0.775   0.983
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.182   0.490   0.637   0.630   0.771   0.983
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.179   0.487   0.635   0.627   0.769   0.984
```

```
Job_LDA_3$scaling
```

```
##             LD1
## gre   -0.002951
## pubs   0.453050
## years -0.159205
```

2.6 Predicted Group Membership

Originally, Fisher proposed the construction of as many classification functions as there were groups, with classification determined by the group with the highest classification score. The Fisher classification functions are available in the DiscriMiner package as a value that can be requested from the linDA( ) function.

```
Job_Fisher <- linDA(Job[, 1:3], Job$job, prior = NULL, validation = NULL,
    learn = NULL, test = NULL, prob = FALSE)
Job_Fisher$functions
```

```
##               Job   No Job
## constant -86.9739 -91.2117
## gre        0.1268   0.1329
## pubs      -0.4572  -1.4186
## years      1.8635   2.2352
```

```
head(Job_Fisher$scores)
```

```
##     Job No Job
## 1 99.80 102.09
## 2 92.43  98.03
## 3 83.31  83.30
## 4 75.05  75.95
## 5 87.35  88.85
## 6 89.53  93.01
```

```
head(Job_Fisher$classification)
```

```
## [1] No Job No Job Job    No Job No Job No Job
## Levels: Job No Job
```

```
Job_Fisher$confusion
```

```
##          predicted
## original Job No Job
##   Job     95     41
##   No Job  20    344
```

```
1 - Job_Fisher$error_rate
```

```
## [1] 0.878
```

Another approach available in the biotools package calculates the Mahalanobis distances of an object from the group centroids for each group and classifies it in the nearest.

```
Job_Mahal <- D2.disc(Job[, 1:3], Job[, 4])
head(Job_Mahal$D2)

##       Job No.Job grouping   pred misclass
## 1  4.2886  1.687   No Job No Job
## 2 14.9213  5.696   No Job No Job
## 3  0.3935  2.382   No Job    Job        *
## 4  2.2663  2.420   No Job    Job        *
## 5  2.8168  1.790   No Job No Job
## 6  8.0757  3.081   No Job No Job

Job_Mahal$confusion.matrix

##        new Job new No Job
## Job        131          5
## No Job      85        279

Proportion_of_Correct_Classification <- sum(diag(Job_Mahal$confusion.matrix))/sum(Job_Mahal$confusion.ma
Proportion_of_Correct_Classification

## [1] 0.82
```

The most common approach uses a Bayesian model, takes prior probabilities into account, and calculates the posterior probabilities for group membership. Cases are classified into the group for which they have the highest posterior probability. This information can be provided by both the lda( ) and candisc( ) functions. The lda( ) function provides the posterior probabilities. The candisc ( ) function shows the classification group and can also provide function means for the groups. The linDA( ) function from the DiscriMiner package directly gives the misclassification rate and a confusion table.

```
Job_Predicted <- predict(Job_LDA)
Job_LDA$svd

## [1] 18.62

Job_LDA$counts

##   1   2
## 364 136

Job_LDA$prior

##     1     2
## 0.728 0.272

Job_Predicted$class

##   [1] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [31] 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [61] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
##  [91] 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## [121] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [151] 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## [241] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1
## [271] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [301] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [331] 1 1 1 2 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [361] 1 1 1 1 1 2 2 2 2 2 1 1 2 2 2 1 2 2 2 2 2 1 2 1 2 2 1 1 2 2
## [391] 2 2 2 1 1 2 1 2 2 1 1 2 1 2 2 1 2 1 1 1 2 1 2 2 1 2 2 2 2 2
## [421] 2 2 1 1 2 2 2 1 1 2 2 1 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 1 2 1
## [451] 2 2 2 2 2 2 2 1 2 1 1 2 2 1 2 1 2 2 2 2 2 2 1 2 1 2 2 2 2 2
## [481] 2 2 1 2 2 2 1 2 1 2 1 2 2 2 2 2 1 2 2 2
## Levels: 1 2

head(Job_Predicted$posterior)

##        1        2
## 1 0.9077 0.092347
## 2 0.9963 0.003695
## 3 0.4975 0.502488
## 4 0.7125 0.287474
## 5 0.8173 0.182719
## 6 0.9702 0.029837

Job_P <- cbind(Job_Predicted$x, Job_Predicted$posterior, Job_Predicted$class)
head(Job_P)

##       LD1      1        2
## 1 -0.2686 0.9077 0.092347 1
## 2 -2.0386 0.9963 0.003695 1
## 3  0.9581 0.4975 0.502488 2
## 4  0.4677 0.7125 0.287474 1
## 5  0.1521 0.8173 0.182719 1
## 6 -0.9081 0.9702 0.029837 1

table(Original = Job$job_num, Predicted = predict(Job_LDA)$class)

##          Predicted
## Original   1   2
##        1 344  20
##        2  41  95

Proportion_of_Correct_Classification <- sum(diag(table(Original = Job$job_num,
    Predicted = predict(Job_LDA)$class)))/sum(table(Original = Job$job_num,
    Predicted = predict(Job_LDA)$class))
Proportion_of_Correct_Classification

## [1] 0.878

Job_CDA$eigenvalues

## [1]  6.960e-01  6.452e-18 -6.594e-18

head(Job_CDA$scores)
```

```
##   as.factor(job)    Can1
## 1         No Job -0.2686
## 2         No Job -2.0386
## 3         No Job  0.9581
## 4         No Job  0.4677
## 5         No Job  0.1521
## 6         No Job -0.9081

Job_CDA$means

## [1]  1.3621 -0.5089

Job_linDA <- linDA(Job[, 1:3], Job[, 4], prior = NULL, validation = NULL,
    learn = NULL, test = NULL, prob = FALSE)
Job_linDA$confusion

##          predicted
## original Job No Job
##   Job     95     41
##   No Job  20    344

1 - Job_linDA$error_rate

## [1] 0.878
```

2.7 Eigenvalues, Lambda, and Canonical Correlations

Classic discriminant analysis is a special case of canonical correlation analysis and within that context it can be useful to examine the canonical correlations and eigenvalues. These are most useful in estimating the magnitude of discrimination of the functions by calculating the percentage of total discrimination that is due to each function. A closely related issue is the statistical significant of the functions.

The candisc( ) function readily provides the eigenvalues and squared canonical correlations. They are related to each other. The ratio of the squared canonical correlation to (1-squared canonical correlation) is the eigenvalue for that function :

$$\lambda_j = \frac{r_j^2}{1 - r_j^2}$$

The eigenvalues can be used to determine the proportion of group separation provided by each discriminant function:

$$proportion_j = \frac{\lambda_j}{\sum_{d=1}^{D} \lambda_d}$$

Wilks' lambda, used in tests of significance, is the product of (1-squared canonical correlation) for a set of functions:

$$\Lambda_j = \prod_{d=j}^{D}(1 - r_d^2)$$

Tests of significance in discriminant analysis are made in a step-wise fashion. First, the entire set is tested for significance by calculating Wilks' lambda (the produce of all 1-squared canonical correlations) and estimating an F ratio (or sometimes a chi-square). If this test is significant, then we can conclude that there is significant discrimination possible. Then the first (and most important) function is excluded and the remainer are tested. If this is not significant, then the first function was the only source of discrimination. If this is significant, then the first and at least the second are significant sources of discrimination. The second is next excluded and the inferences follow in the same fashion. If the remainder are not significant, the first and second were the only sources of significance.

```
Job_CDA$rank

## [1] 1

Job_CDA$eigenvalues

## [1]  6.960e-01  6.452e-18 -6.594e-18

Job_CDA$canrsq

## [1] 0.4104

Job_CDA$pct

## [1]  1.000e+02  9.270e-16 -9.474e-16

Job_CDA

##
## Canonical Discriminant Analysis for as.factor(job):
##
##   CanRsq Eigenvalue Difference Percent Cumulative
## 1   0.41      0.696                100        100
##
## Test of H0: The canonical correlations in the
## current row and all that follow are zero
##
##   LR test stat approx F numDF denDF Pr(> F)
## 1         0.59      115     3   496  <2e-16
```

2.8 Homogeneity Assumption

We assume in discriminant analysis that the separate group variance-covariance matrices are homogeneous. This assumption underlies tests used to determine the number of significant functions. Box's test can be used to test this assumption.

```
boxM(Job[, 1:3], Job$job)

##
##  Box's M-test for Homogeneity of Covariance Matrices
```

```
##
## data:  Job[, 1:3]
## Chi-Sq (approx.) = 110, df = 6, p-value <2e-16
```

```
boxM(Job[, 1:3], Job$job)$cov
```

```
## $Job
##            gre    pubs    years
## gre   10455.78 56.3826 -25.2563
## pubs     56.38  2.2065   0.5251
## years   -25.26  0.5251   1.4056
##
## $`No Job`
##            gre    pubs    years
## gre   10885.13 74.3135 -14.0374
## pubs     74.31  3.9523  -0.7029
## years   -14.04 -0.7029   4.5581
```

```
boxM(Job[, 1:3], Job$job)$pooled
```

```
##            gre   pubs   years
## gre   10768.74 69.453 -17.079
## pubs     69.45  3.479  -0.370
## years   -17.08 -0.370   3.704
```

2.9 Multivariate Normality Assumption

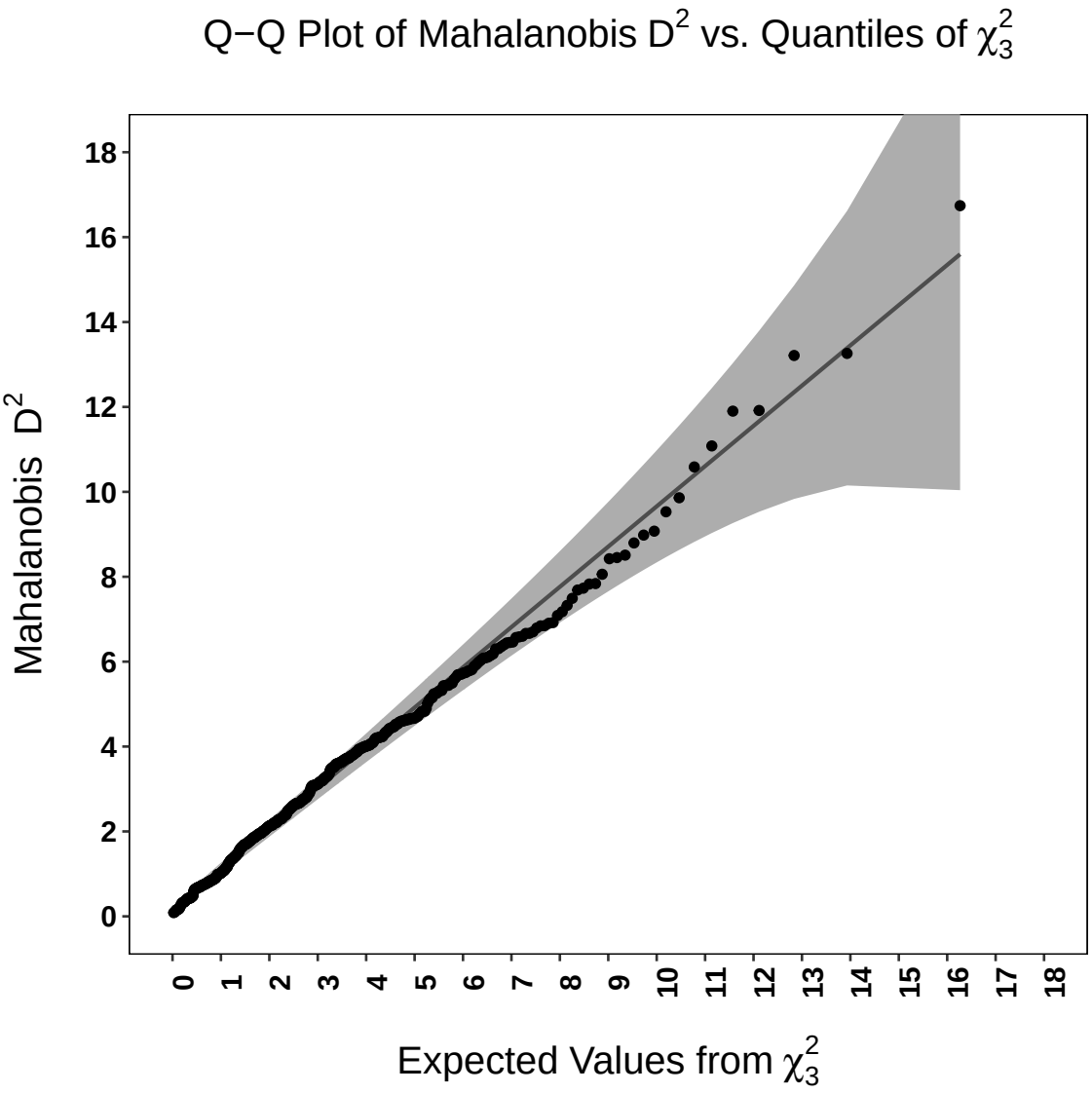
The classification part of discriminant analysis (as well as any significance tests for the discriminant functions) rely on the multivariate normality assumption. The tests are performed on the residualized data so that species differences do not affect the results. Note that a violation of multivariate normality will also affect the test of homogeneity of covariance matrices.

```
mvn(Job[, 7:9], mvnTest = "mardia")
```

```
## $multivariateNormality
##              Test        Statistic              p value Result
## 1 Mardia Skewness  86.730301008051 2.37724296170123e-14     NO
## 2 Mardia Kurtosis -1.6503616863684   0.0988689824798465    YES
## 3             MVN             <NA>                 <NA>     NO
##
## $univariateNormality
##           Test  Variable Statistic   p value Normality
## 1 Shapiro-Wilk   gre_R      0.9962  0.2749      YES
## 2 Shapiro-Wilk  pubs_R      0.9683  <0.001      NO
## 3 Shapiro-Wilk  years_R     0.9349  <0.001      NO
##
## $Descriptives
##           n      Mean Std.Dev  Median      Min     Max    25th
## gre_R   500 1.121e-15 103.669  0.1618 -268.338 265.245 -75.005
## pubs_R  500 2.414e-17   1.863 -0.4698   -3.529   4.530  -1.470
## years_R 500 6.825e-17   1.923 -0.5330   -2.533   7.467  -1.533
##           75th    Skew Kurtosis
## gre_R   64.245 0.05913  -0.3313
```

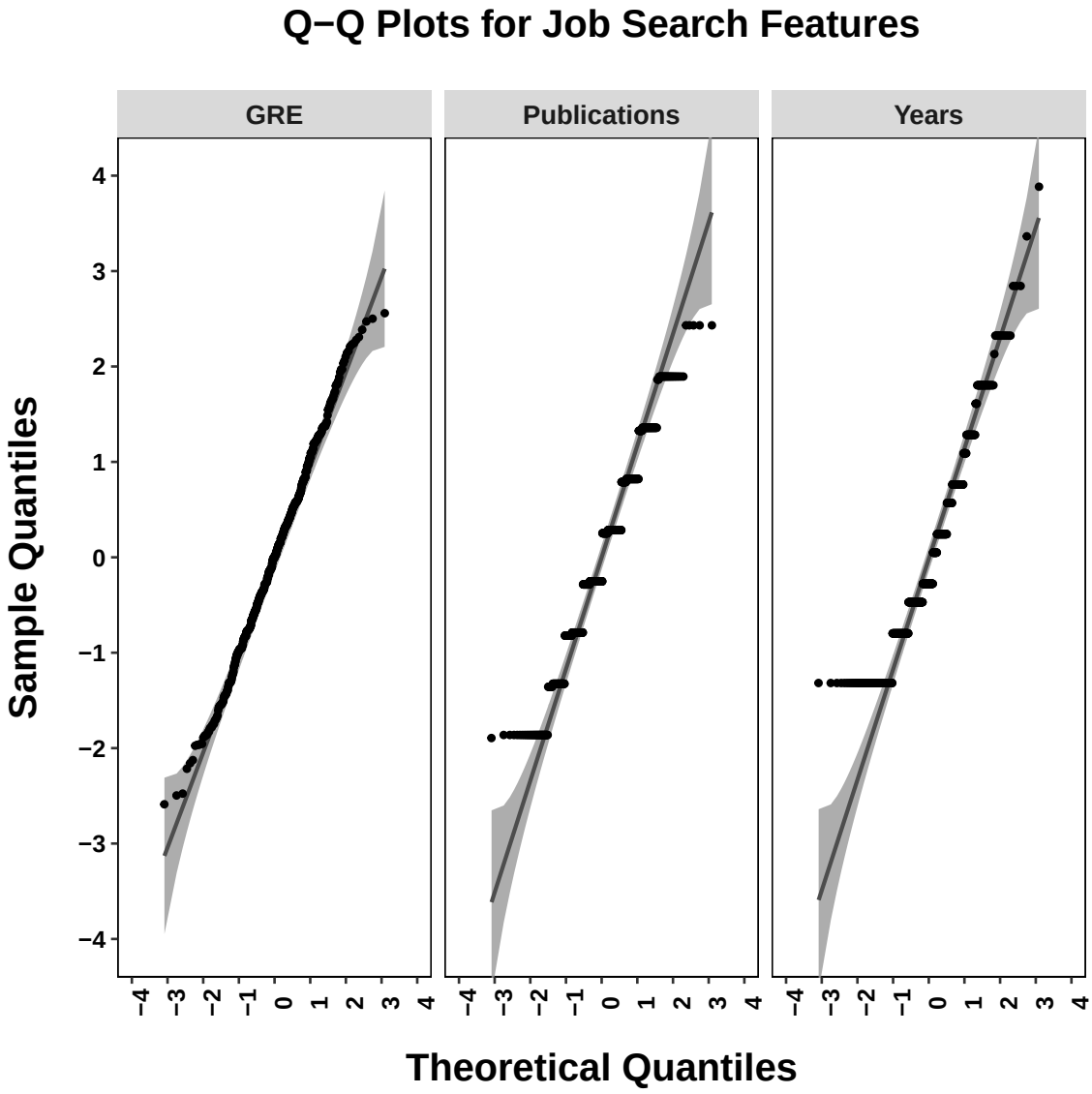
```
## pubs_R   1.471 0.07606  -0.5278
## years_R  1.467 0.72303   0.1353
```

```
CV <- cov(Job[, 7:9])
D2_1 <- mahalanobis(Job[, 7:9], center = colMeans(Job[, 7:9]), cov = CV)
D2_1 <- as.data.frame(D2_1)
ggplot(D2_1, aes(sample = D2_1)) + stat_qq_band(distribution = "chisq",
    dparams = list(df = 3)) + stat_qq_line(distribution = "chisq",
    dparams = list(df = 3)) + stat_qq(distribution = "qchisq", dparams = list(df = 3)) +
    scale_y_continuous(breaks = seq(0, 18, 2)) + scale_x_continuous(breaks = seq(0,
    18, 1)) + coord_cartesian(xlim = c(0, 18), ylim = c(0, 18)) +
    xlab(expression("Expected Values from" * ~chi[3]^2)) + ylab(expression("Mahalanobis " *
    ~D^2)) + 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, 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(expression("Q-Q Plot of Mahalanobis" *
    ~D^2 * " vs. Quantiles of" * ~chi[3]^2))
```




```
Job_QQ <- scale(Job[, 7:9])
Data_long <- melt(Job_QQ)
Data_long <- as.data.frame(Data_long)
names(Data_long) <- c("Index", "feature", "value")
Data_long$feature_F <- factor(Data_long$feature, levels = c("gre_R",
    "pubs_R", "years_R"), labels = c("GRE", "Publications", "Years"))
p <- ggplot(Data_long, aes(sample = value)) + stat_qq_band() + stat_qq_line() +
    stat_qq(distribution = qnorm, size = 1) + scale_y_continuous(breaks = seq(-4,
    4, 1)) + scale_x_continuous(breaks = seq(-4, 4, 1)) + coord_cartesian(xlim = c(-4,
    4), ylim = c(-4, 4)) + xlab("Theoretical Quantiles") + ylab("Sample Quantiles") +
    theme(text = element_text(size = 14, family = "sans", color = "black",
        face = "bold"), axis.text.y = element_text(colour = "black",
        size = 10, face = "bold"), axis.text.x = element_text(colour = "black",
        size = 10, 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("Q-Q Plots for Job Search Features")
p + facet_wrap(~feature_F)
```



2.10 Quadratic Model

When the homogeneity of covariance matrices assumption is not met, an alternative quadratic model can be fit instead using the qda( ) function. This method produces

matrices that transform the observations for each group so that the within-groups covariance matrix is spherical (proportional to an identity matrix). An alternative is bootstrapping with the standard analysis. This approach is described later.

```
# By specifying the jackknife procedure, the classification
# results can be obtained directly.
Job_QDA <- qda(Job$job_num ~ gre + pubs + years, data = Job, CV = TRUE)
head(Job_QDA$posterior)
```

```
##        1         2
## 1 0.9993 6.563e-04
## 2 1.0000 8.685e-13
## 3 0.4031 5.969e-01
## 4 0.6188 3.812e-01
## 5 0.7609 2.391e-01
## 6 0.9918 8.196e-03
```

```
Job_QDA$class
```

```
##   [1] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [31] 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
##  [61] 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1
##  [91] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1
## [121] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1
## [151] 1 1 1 1 1 1 2 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
## [181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [211] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## [241] 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1
## [271] 2 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
## [301] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [331] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [361] 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 1 2 2
## [391] 2 2 2 2 1 2 2 2 2 2 1 2 2 1 2 2 2 2 1 1 2 2 1 2 2 1 2 2 2 2 2
## [421] 2 2 2 1 2 2 2 1 1 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1
## [451] 1 2 2 2 2 2 2 2 1 2 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2
## [481] 1 2 1 2 2 2 2 1 1 2 2 1 2 2 2 2 2 2 2 2 2
## Levels: 1 2
```

```
table(Original = Job$job_num, Predicted = Job_QDA$class)
```

```
##          Predicted
## Original   1   2
##        1 332  32
##        2  28 108
```

```
Proportion_of_Correct_Classification <- sum(diag(table(Original = Job$job_num,
    Predicted = Job_QDA$class)))/sum(table(Original = Job$job_num,
    Predicted = Job_QDA$class))
Proportion_of_Correct_Classification
```

```
## [1] 0.88
```

```
# With CV=FALSE, the function gives the scaling matrices and the
# predict function is used to get the classifications.
Job_QDA <- qda(Job$job_num ~ gre + pubs + years, data = Job, CV = FALSE)
```

```
Job_QDA_P <- predict(Job_QDA)
head(Job_QDA_P$posterior)
```

```
##        1         2
## 1 0.9993 6.515e-04
## 2 1.0000 8.404e-13
## 3 0.4056 5.944e-01
## 4 0.6211 3.789e-01
## 5 0.7622 2.378e-01
## 6 0.9919 8.071e-03
```

```
Job_QDA_P$class
```

```
##   [1] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [31] 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
##  [61] 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1
##  [91] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [121] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1
## [151] 1 1 1 1 1 1 2 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
## [181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [211] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## [241] 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1
## [271] 2 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [301] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [331] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [361] 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 1 2 2
## [391] 2 2 2 2 1 2 2 2 2 1 2 2 1 2 2 2 2 1 1 2 2 1 2 2 1 2 2 2 2 2
## [421] 2 2 2 1 2 2 2 1 1 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1
## [451] 1 2 2 2 2 2 2 1 2 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2
## [481] 1 2 1 2 2 2 1 1 2 2 1 2 2 2 2 2 2 2 2 2
## Levels: 1 2
```

```
table(Original = Job$job_num, Predicted = Job_QDA_P$class)
```

```
##          Predicted
## Original   1   2
##        1 333  31
##        2  28 108
```

```
Proportion_of_Correct_Classification <- sum(diag(table(Original = Job$job_num,
    Predicted = Job_QDA_P$class)))/sum(table(Original = Job$job_num,
    Predicted = Job_QDA_P$class))
Proportion_of_Correct_Classification
```

```
## [1] 0.882
```

2.11 Cross-Validation

The basic predict( ) function when applied to the entire data set will simply provide the classification from the model and the ability to determine the percentage correct classification. The jackknife procedure [use CV=TRUE in lda()] will leave each case out in turn, estimate the discriminant analysis with the remaining cases, and then use that information to classify the left out case. This approach insures that each case is classified with information it did not contribute to in the estimation. A traditional cross-validation

uses part of the sample (or a separate sample) to estimate the discriminant functions and then applies that solution to the remaining cases. Each is illustrated here.

2.11.1 Simple Prediction

```
table(Original = Job$job_num, Predicted = predict(Job_LDA)$class)
```

```
##          Predicted
## Original   1   2
##        1 344  20
##        2  41  95
```

```
Proportion_of_Correct_Classification <- sum(diag(table(Original = Job$job_num,
    Predicted = predict(Job_LDA)$class)))/sum(table(Original = Job$job_num,
    Predicted = predict(Job_LDA)$class))
Proportion_of_Correct_Classification
```

```
## [1] 0.878
```

2.11.2 Leave-One-Out

The jackknife procedure will leave each case out in turn, estimate the discriminant analysis with the remaining cases, and then use that information to classify the left-out case. This approach insures that each case is classified with information it did not contribute to in the estimation.

```
Job_Jack <- lda(job_num ~ gre + pubs + years, data = Job, CV = TRUE)
Jack_T <- table(Original = Job$job_num, Predicted = Job_Jack$class)
Proportion_of_Correct_Classification <- sum(diag(Jack_T))/sum(Jack_T)
Proportion_of_Correct_Classification
```

```
## [1] 0.866
```

2.11.3 Separate-Samples Cross-Validation

The most convincing cross-validation uses part of the sample (or a separate sample) to estimate the discriminant functions and then applies that solution to the remaining cases. Here the sample is split into random halves. The first sample (called the training set) is used to derive the discriminant functions. Those functions are then used on the second set to classify cases.

A single cross-validation is open to variability, especially as the sample sizes decrease. This variability can be made apparent with the bootstrapping procedure described later. If sample sizes are very large, the entire sample can be split into multiple subsamples, called folds, and a K-fold cross-validation conducted. This uses, in turn, one of the subsamples as the validation group with the remaining subsamples used as the training sample. The mean classification can be taken across the multiple validations and variability can be directly estimated.

```
training_sample <- sample(1:500, 250)
Job_Train <- lda(job_num ~ gre + pubs + years, data = Job, CV = FALSE,
    subset = training_sample)
```

```
Job_Predict <- predict(Job_Train, newdata = Job[-training_sample,
    ])
Job_Original <- as.data.frame(Job[-training_sample, 6])
Job_Cross <- cbind(Job_Original, Job_Predict$class)
names(Job_Cross) <- c("Original", "Predicted")
table(Original = Job_Cross$Original, Predicted = Job_Cross$Predicted)
```

```
##          Predicted
## Original   1   2
##        1 173  14
##        2  18  45
```

```
Proportion_of_Correct_Classification <- sum(diag(table(Original = Job_Cross$Original,
    Predicted = Job_Cross$Predicted)))/sum(table(Original = Job_Cross$Original,
    Predicted = Job_Cross$Predicted))
Proportion_of_Correct_Classification
```

```
## [1] 0.872
```

2.12 Classification Quality

In addition to the proportion correct classification, there are several other common ways to assess classification quality. One

2.12.1 Klecka's Tau

Klecka's tau (τ) is sometimes calculated for classification results:

$$\tau = \frac{N_o - \sum_{i=1}^{G} p_i n_i}{N - \sum_{i=1}^{G} p_i n_i}$$

N_o is the observed number of correct classifications, n_i is the number of cases in group i, p_i is the proportion of the total sample expected to be in group i, G is the number of groups, and N is the total sample size.

```
Class_T <- table(Original = Job$job_num, Predicted = predict(Job_LDA)$class)
Class_T
```

```
##          Predicted
## Original   1   2
##        1 344  20
##        2  41  95
```

```
Proportion_of_Correct_Classification <- sum(diag(Class_T))/sum(Class_T)
Proportion_of_Correct_Classification
```

```
## [1] 0.878
```

```
# Marginals (O = Observed, P = Predicted)
MO1 <- sum(Class_T[1, ])
MO2 <- sum(Class_T[2, ])
```

```
MP1 <- MO1
MP2 <- MO2
# Total observations
N <- sum(Class_T)
# Observed agreement
O <- sum(diag(Class_T))
# Expected agreement
E <- (MO1 * MP1/N) + (MO2 * MP2/N)
# Klecka's tau
Tau <- (O - E)/(N - E)
Tau

## [1] 0.6919
```

2.12.2 Classification Significance

The total number of correct classifications that would occur by chance (302, 60.4%) can be tested against the actual number of correct classifications given the discriminant analysis model (439, 87.8%, for the simple prediction of all cases). A t-test can be calculated:

$$t = \frac{439 - 302}{\sqrt{500(.604)(1 - .604)}} = 12.53$$

Where the denominator is the standard error of the number of correct classifications by chance (the null hypothesis).

```
t <- (O - E)/sqrt(N * (E/N) * (1 - E/N))
```

The difference between chance expected and actual classification can be tested with a chi-square as well. :

$$\chi^2 = \sum_{i=1}^{C} \frac{(f_{o_i} - f_{e_i})^2}{f_{e_i}}$$

$$\chi^2 = \frac{(439 - 302)^2}{302} + \frac{(61 - 198)^2}{198} = 156.94$$

Because this is a single degree of freedom test, t^2 = χ^2.

```
chi_squared <- (((O - E)^2)/E) + ((((500 - O) - (500 - E))^2)/(500 -
    E))
chi_squared

## [1] 157
```

For comparison, here is the classic χ^2, which does not assume that the predicted marginals are the same as the obtained marginals.

```
O1 <- Class_T[1, 1]
O2 <- Class_T[1, 2]
O3 <- Class_T[2, 1]
O4 <- Class_T[2, 2]
E1 <- (sum(Class_T[1, ]) * sum(Class_T[, 1]))/N
E2 <- (sum(Class_T[2, ]) * sum(Class_T[, 1]))/N
E3 <- (sum(Class_T[2, ]) * sum(Class_T[, 1]))/N
E4 <- (sum(Class_T[2, ]) * sum(Class_T[, 2]))/N
chi_squared <- (((O1 - E1)^2)/E1) + (((O2 - E2)^2)/E2) + (((O3 - E3)^2)/E3) +
    (((O4 - E4)^2)/E4)
```

2.12.3 Other Indices

The confusionMatrix( ) function from the caret package provides quite a number of other indices. Of note, it provides Cohen's kappa, a chance-corrected agreement statistic. If the data are arranged in a confusion table as follows:

		Actual		
		Absent	*Present*	*Marginal*
Prediction	Absent	d	c	Row 1 = d+c
	Present	b	a	Row 2 = b+a
	Marginal	Column 1 = d+b	Column 2 = c+a	N=a+b+c+d

Cohen's κ is defined as:

$$\kappa = \frac{N_o - N_e}{N - N_e} = \frac{p_o - p_e}{1 - p_e}$$

N_o is the number of correct classifications (the sum of the main diagonal: d+a). N_e is the number of correct classifications expected by chance. This is calculated using the marginals: [(Row 1 x Column 1)+(Row 2 x Column 2)]/N. N is the total sample size. Or, κ can be estimated with proportions, p_o and p_e.

The other indices reported are likewise a function of elements in the table. They are commonly used when there are only two groups (as here). When there are more than two groups, then these are also reported, but for all combinations of each group versus the combination of the remaining groups. Some of the more useful are precision, recall, and F1.

Precision is defined as:

$$Precision = \frac{a}{a + b}$$

This index answers the question, "What percentage of predicted events are correct?"

Recall is defined as:

$$Recall = \frac{a}{a + c}$$

This index answers the question, "What percentage of events were correctly predicted?"

Precision and recall are negatively related; as one increases, the other decreases. An index that combines them is F1, defined as:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

This is the harmonic mean of precision and recall.

Some of the remaining indices are often useful. These are the most common.

Sensitivity is defined as:

$$Sensitivity = \frac{a}{a + c}$$

Sensitivity is the same as recall: "What percentage of events were correctly predicted?"

Specificity is defined as:

$$Specificity = \frac{d}{b + d}$$

This index answers the question: "What percentage of event absences were correctly predicted?"

Prevalence is defined as:

$$Prevalence = \frac{a + c}{a + b + c + d}$$

This index aswers the question: "What is the proportion of actual events in the sample?"

Detection Rate is defined as:

$$Detection\ \ Rate = \frac{a}{a + b + c + d}$$

This index answers the question: "What proportion of the entire sample are correctly predicted events?"

Detection Prevalence is defined as:

$$Detection\ \ Prevalence = \frac{a + b}{a + b + c + d}$$

This index answers the question: "What proportion of the entire sample are predicted events?"

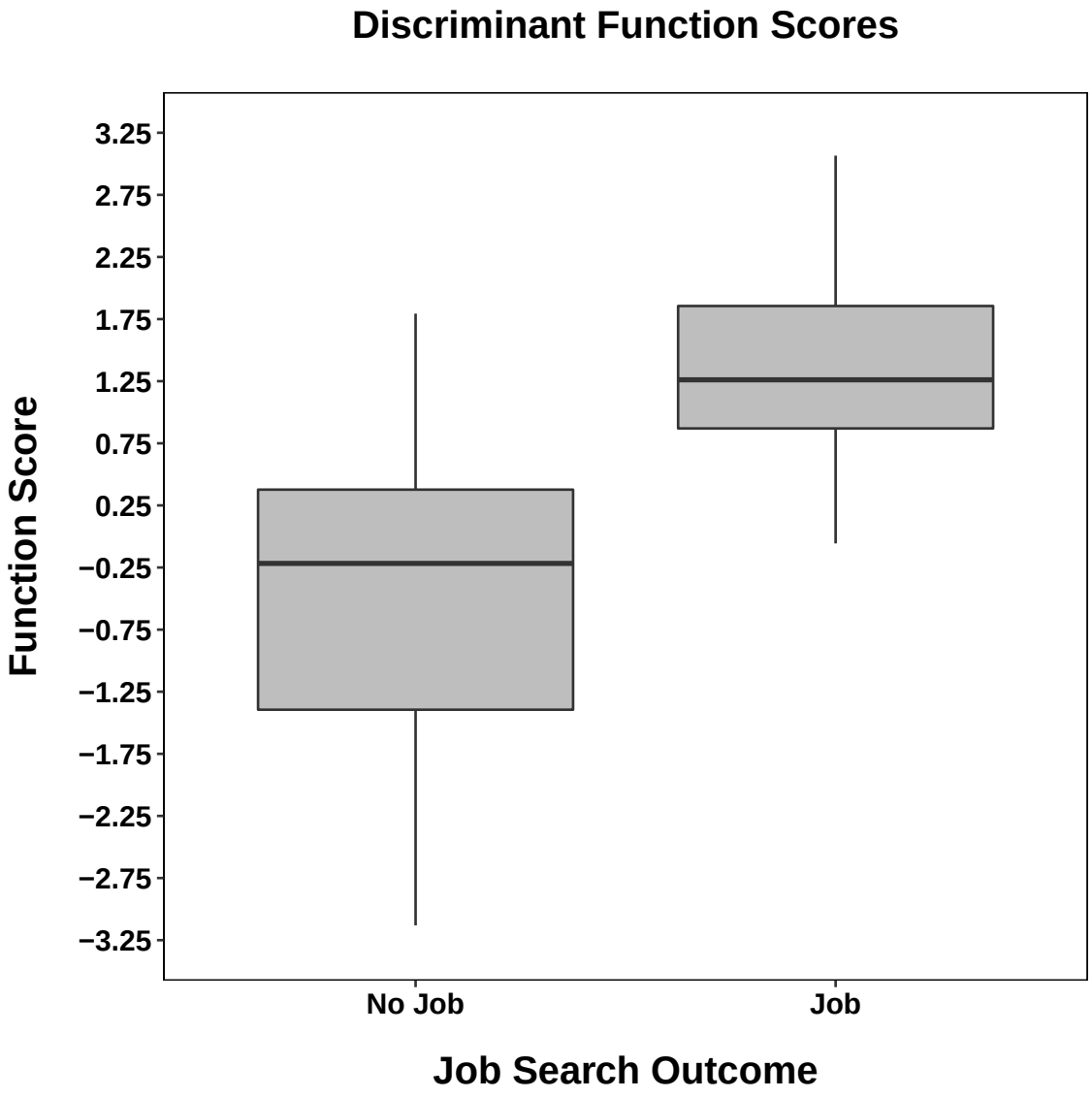
```
Job_Predicted <- factor(Job_Jack$class, levels = c(1, 2), labels = c("No Job",
    "Job"))
Job_Actual <- factor(Job$job_num, levels = c(1, 2), labels = c("No Job",
    "Job"))
confusionMatrix(Job_Predicted, Job_Actual, positive = "Job", mode = "everything")

## Error in confusionMatrix(Job_Predicted, Job_Actual, positive = "Job", :  could not find
function "confusionMatrix"
```

2.13 Visualization

There are a number of ways to visualize the analysis. A few examples are given here.

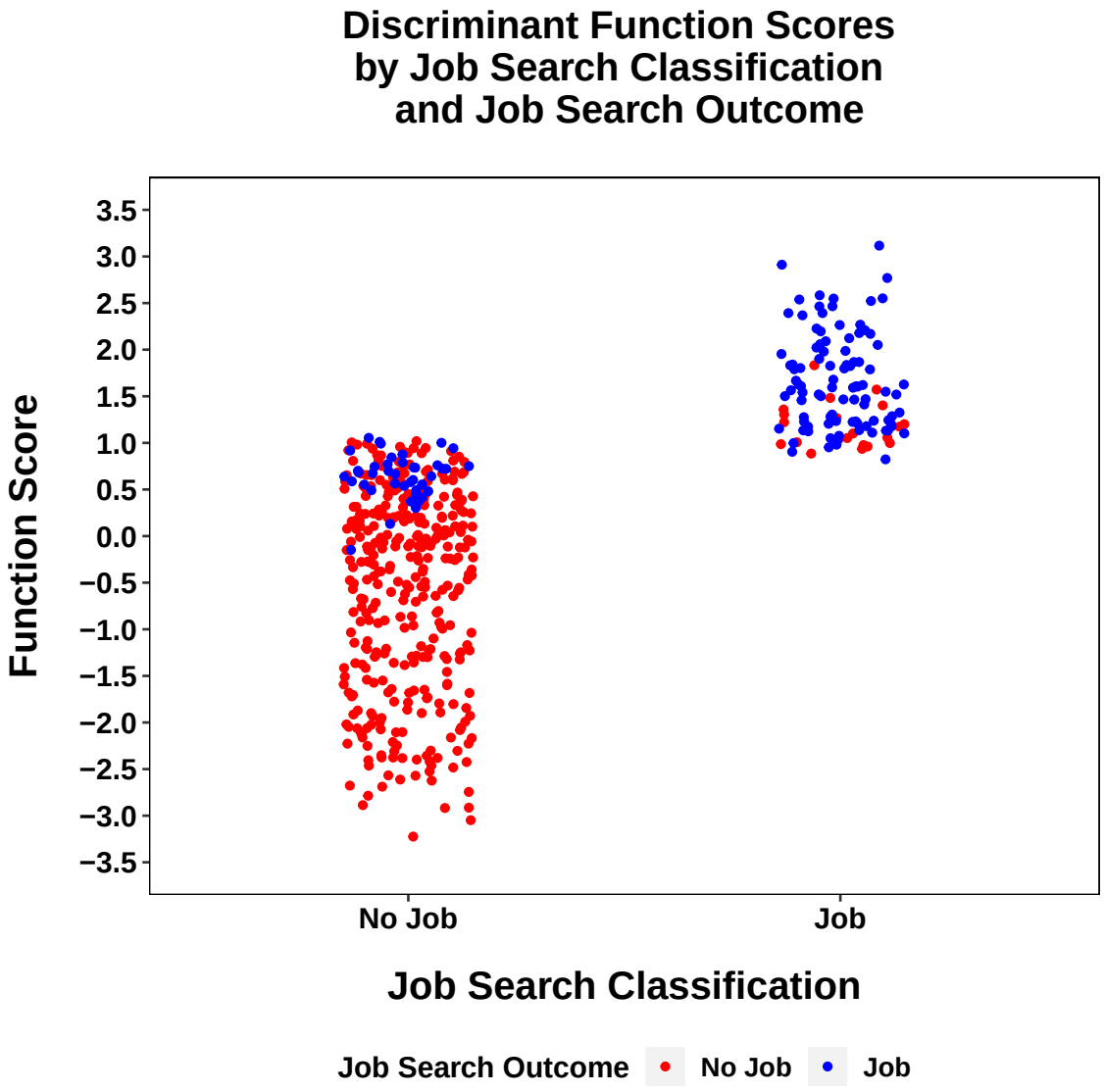
```
Job_LDA_Values <- predict(Job_LDA)
LDA_Means <- as.data.frame(aggregate(Job_LDA_Values$x, by = list(Job_LDA_Values$class),
    FUN = mean, na.rm = TRUE))
names(LDA_Means) <- c("Class", "Mean")
LDA_Means$Class_F <- factor(LDA_Means$Class, levels = c("1", "2"),
    labels = c("No Job", "Job"))
```

```
plot_data <- cbind(Job_LDA_Values$x[, 1], Job_LDA_Values$class)

plot_data <- as.data.frame(plot_data)
names(plot_data) <- c("Score", "Class")
plot_data$Class_F <- factor(plot_data$Class, levels = c(1, 2), labels = c("No Job",
    "Job"))
plot_data$Job_F <- factor(Job$job_num, levels = c(1, 2), labels = c("No Job",
    "Job"))

ggplot(plot_data, aes(x = Job_F, y = Score)) + geom_boxplot(fill = "gray") +
    scale_y_continuous(breaks = c(seq(-3.25, 3.25, 0.5))) + coord_cartesian(xlim = c(1,
    2), ylim = c(-3.25, 3.25)) + ylab("Function Score") + xlab("Job Search Outcome") +
    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, face = "bold", angle = 0), 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("Discriminant Function Scores")
```




```
ggplot(plot_data, aes(x = Class_F, y = Score, color = Job_F)) + geom_jitter(width = 0.15,
    height = 0.15, shape = 19, size = 1.25, na.rm = TRUE) + scale_color_manual("Job Search Outcome",
    values = c("red", "blue")) + scale_y_continuous(breaks = c(seq(-3.5,
    3.5, 0.5))) + coord_cartesian(xlim = c(1, 2), ylim = c(-3.5, 3.5)) +
    xlab("Job Search Classification") + ylab("Function Score") + 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, face = "bold", angle = 0), 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"),
```

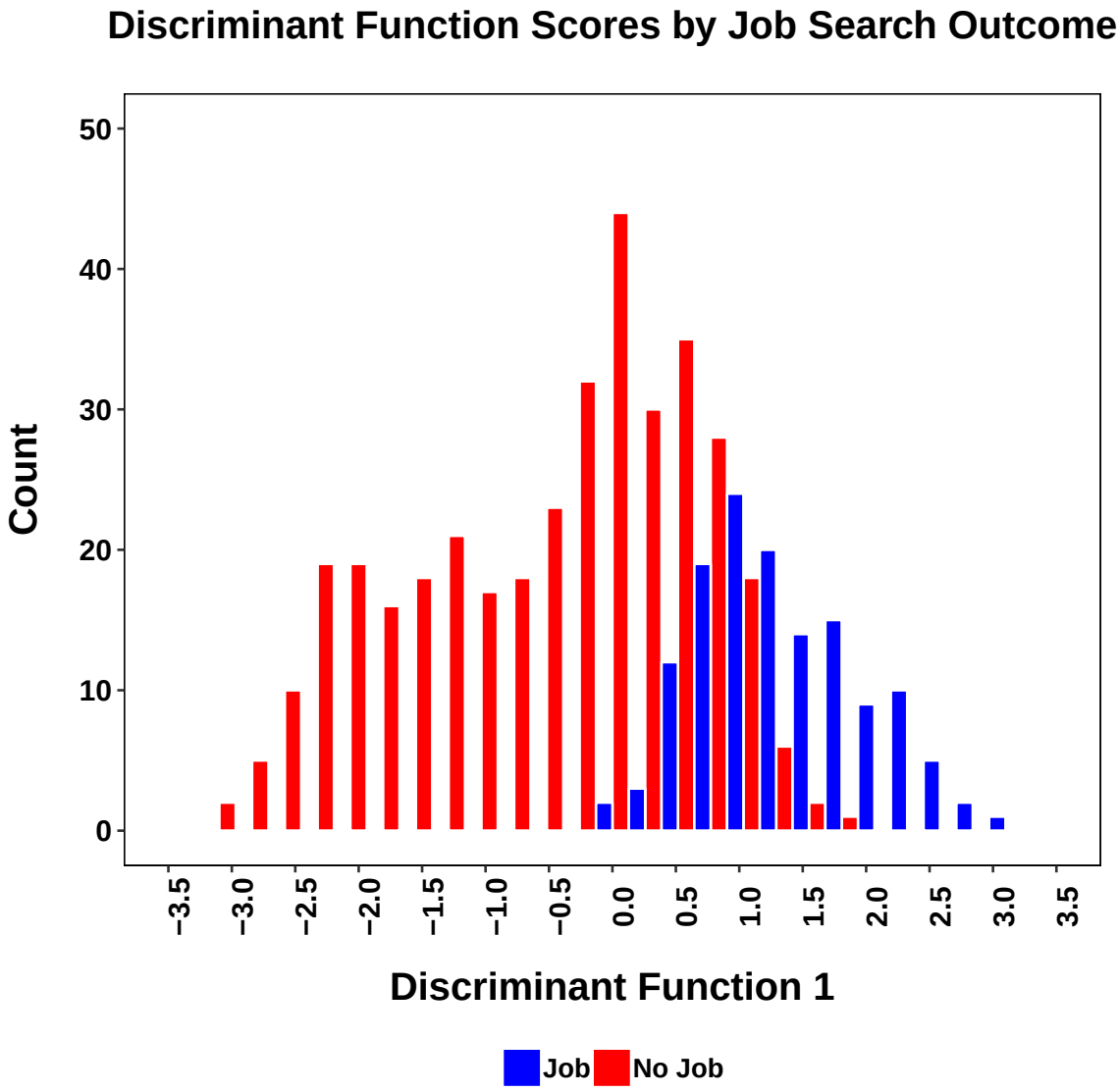
```
    plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
    legend.title = element_text(colour = "black", size = 12, face = "bold")) +
    ggtitle("Discriminant Function Scores \nby Job Search Classification \n and Job Search Outcome")
```



The different discrimination is evident when the discriminant function scores are examined in analyses of variance.

```
summary(aov(Job_LDA_Values$x[, 1] ~ as.factor(Job$job)))

##                      Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(Job$job)    1    347     347     347 <2e-16
## Residuals           498    498       1
```

```r
plot_data <- cbind(Job_LDA_Values$class, Job$job)
plot_data <- as.data.frame(plot_data)
names(plot_data) <- c("Class", "Job")
plot_data$Score <- as.numeric(Job_LDA_Values$x[, 1])
plot_data$Class_F <- factor(plot_data$Class, levels = c(1, 2), labels = c("No Job",
    "Job"))

ggplot(plot_data, aes(x = Score, fill = Job)) + geom_histogram(color = "white",
    alpha = 1, bins = 25, size = 0.5, position = "dodge") + scale_fill_manual(values = c("blue",
    "red")) + scale_x_continuous(breaks = seq(-3.5, 3.5, 0.5)) + coord_cartesian(xlim = c(-3.5,
    3.5), ylim = c(0, 50)) + xlab("Discriminant Function 1") + ylab("Count") +
    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, 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("Discriminant Function Scores by Job Search Outcome")
```



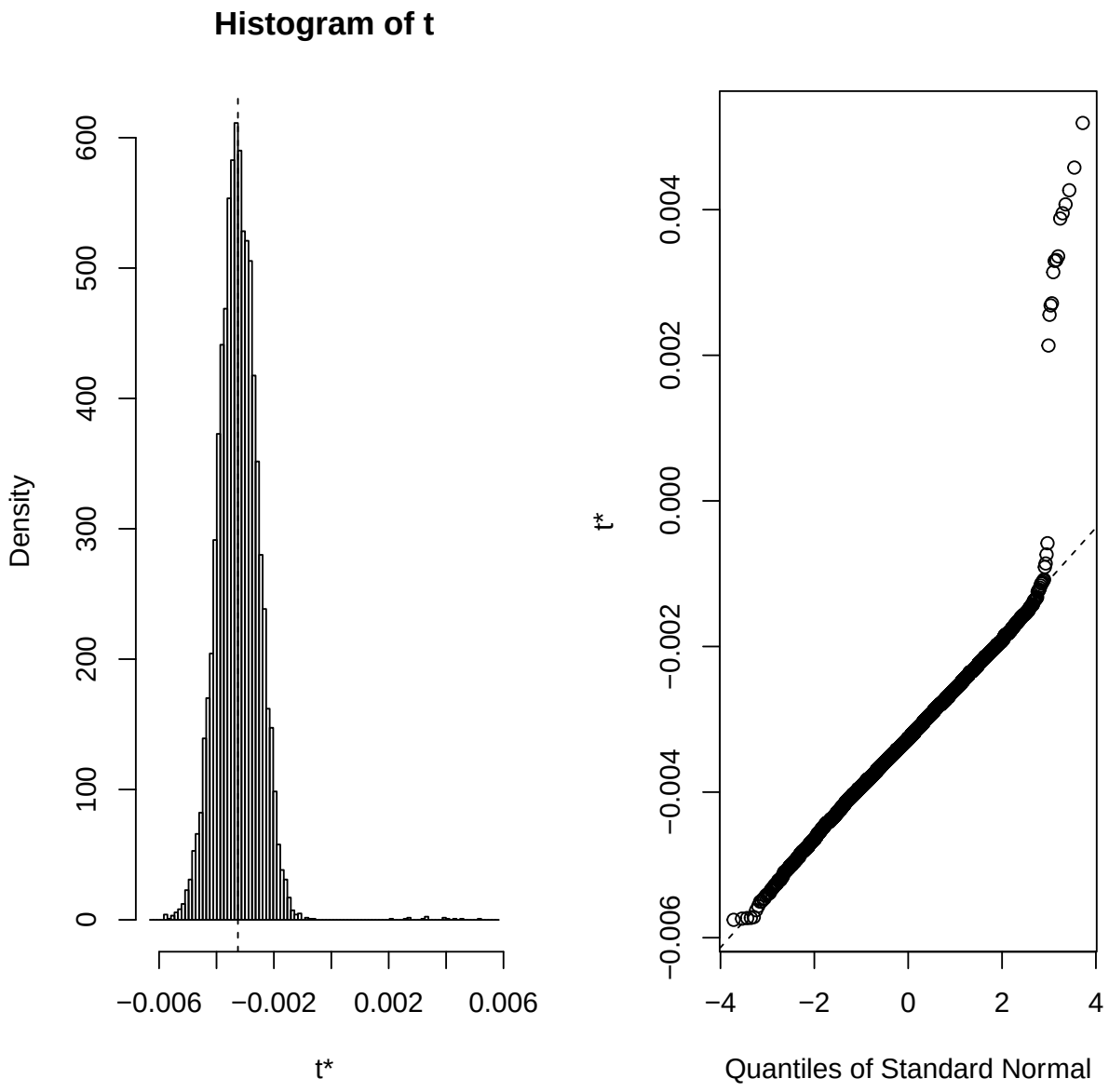
3 Bootstrap Analysis

If the assumptions underlying the discriminant analysis (homogeneous covariance matrices, multivariate normality) are not viable, an alternative approach can be taken that does not make these assumptions: bootstrapping. In the bootstrapping approach, we assume that whatever population the sample came from, it is representative of that population. Therefore we can sample randomly from it, with replacement, to get repeated samples of the same size on which we can repeat the analyses. The resulting empirical distributions of key indices (coefficients, proportion classified correctly, tau) can be examined and confidence intervals placed to assist inferences.

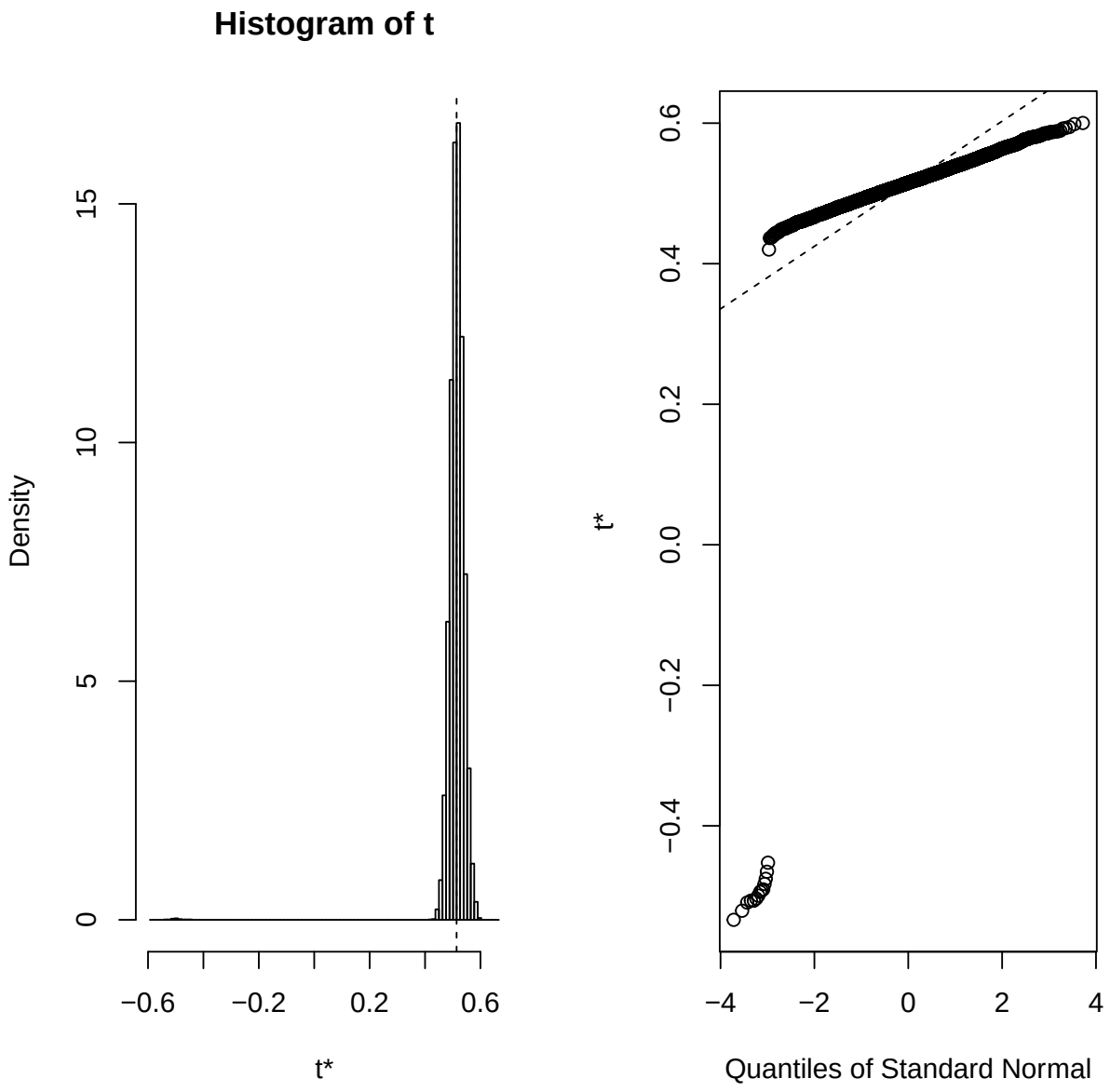
```
# A function that will conduct the linear discriminant analysis
# using resamples of the data.  The first formula is for the lda(
```

```
# ) model, using the jackknife procedure, in order to get
# classification results. The second forumula is for the candisc(
# ) function in order to get discriminant function coefficients
# and structure coefficients. A total of 11 values from each
# analysis are returned for illustration in bootstrap
# distributions: raw function coefficients for gre, pubs, and
# years; standardized coefficients for gre, pubs and years;
# structure coefficients for gre, pubs, and years; proportion of
# correct classification, ; and tau.
lda_boot <- function(formula_1, formula_2, data, indices) {
    boot_data <- data[indices, ]
    boot_fit_1 <- lda(formula_1, data = boot_data, CV = TRUE)
    boot_MLM <- lm(formula_2, data = boot_data)
    boot_fit_2 <- candisc(boot_MLM, data = boot_data)
    results <- rbind(boot_fit_2$coeffs.raw, boot_fit_2$coeffs.std,
        boot_fit_2$structure)
    T <- table(Original = boot_data$job_num, Predicted = boot_fit_1$class)
    PC <- sum(diag(T))/sum(T)
    MO1 <- sum(T[1, ])
    MO2 <- sum(T[2, ])
    MP1 <- MO1
    MP2 <- MO2
    N <- sum(T)
    O <- sum(diag(T))
    E <- (MO1 * MP1/N) + (MO2 * MP2/N)
    Tau <- (O - E)/(N - E)
    results <- rbind(results, PC, Tau)
    return(results)
}
```

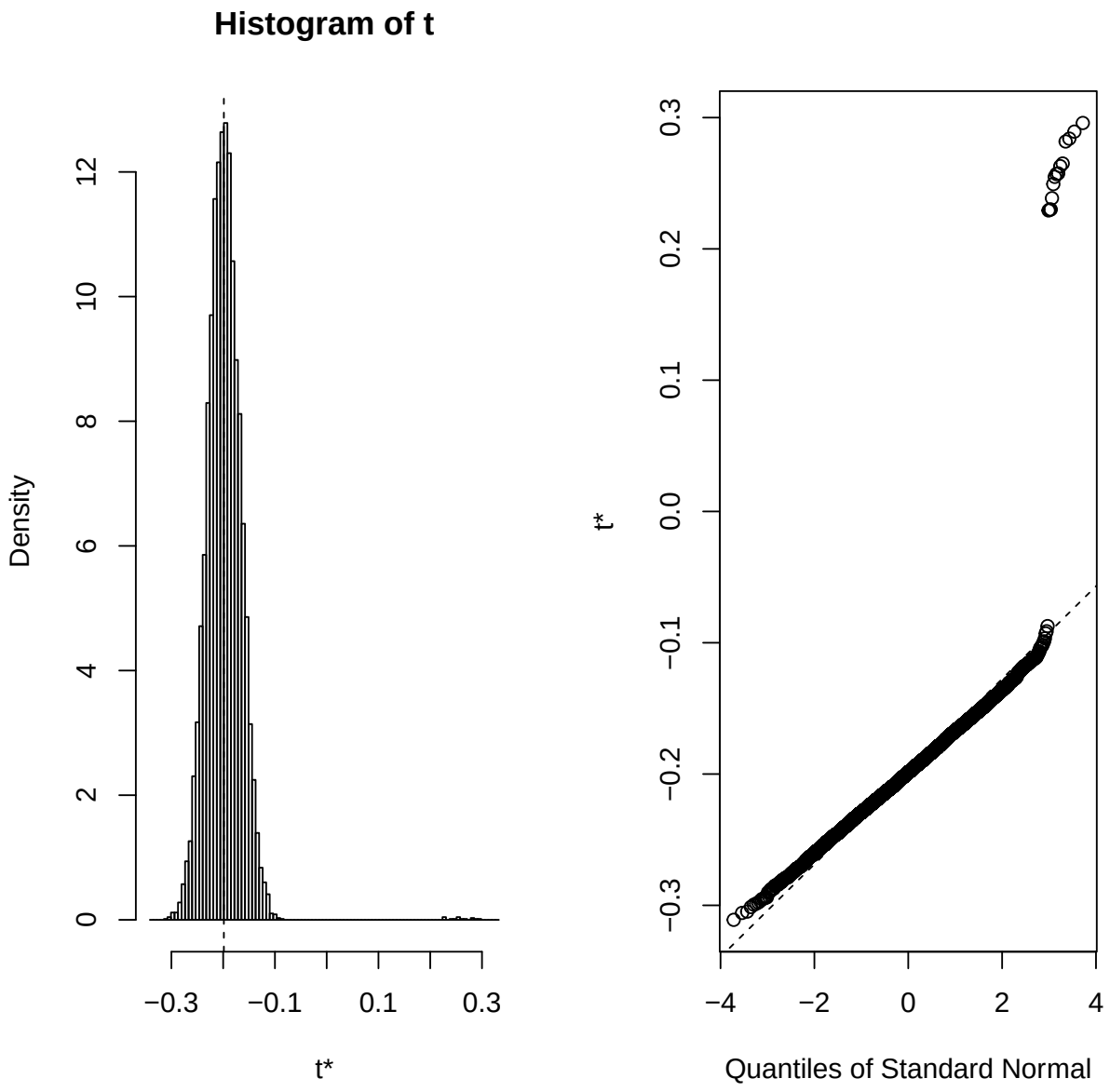
```
# Select the data from the original file that will be passed to
# the bootstrapping function.
boot_input <- Job[, c(1:3, 6)]

# Call the boot( ) function from the boot library and request
# 10000 resamples.
boot_results <- boot(data = boot_input, statistic = lda_boot, R = 10000,
    formula_1 = job_num ~ gre + pubs + years, formula_2 = cbind(gre,
        pubs, years) ~ as.factor(job_num))

# Plot the bootstrapping results using the default plot( )
# function.
plot(boot_results, index = 1)
```



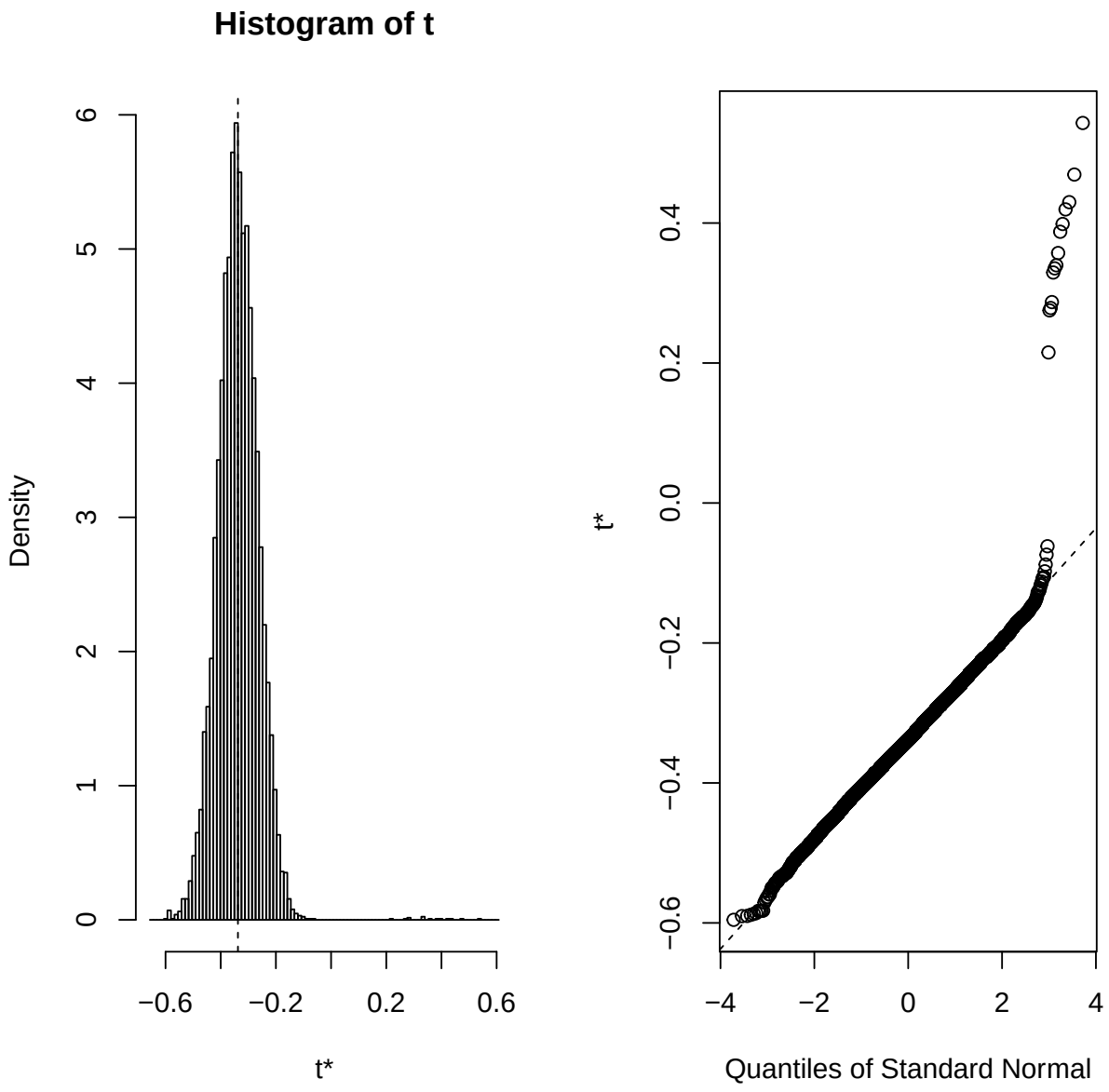

```
plot(boot_results, index = 2)
```



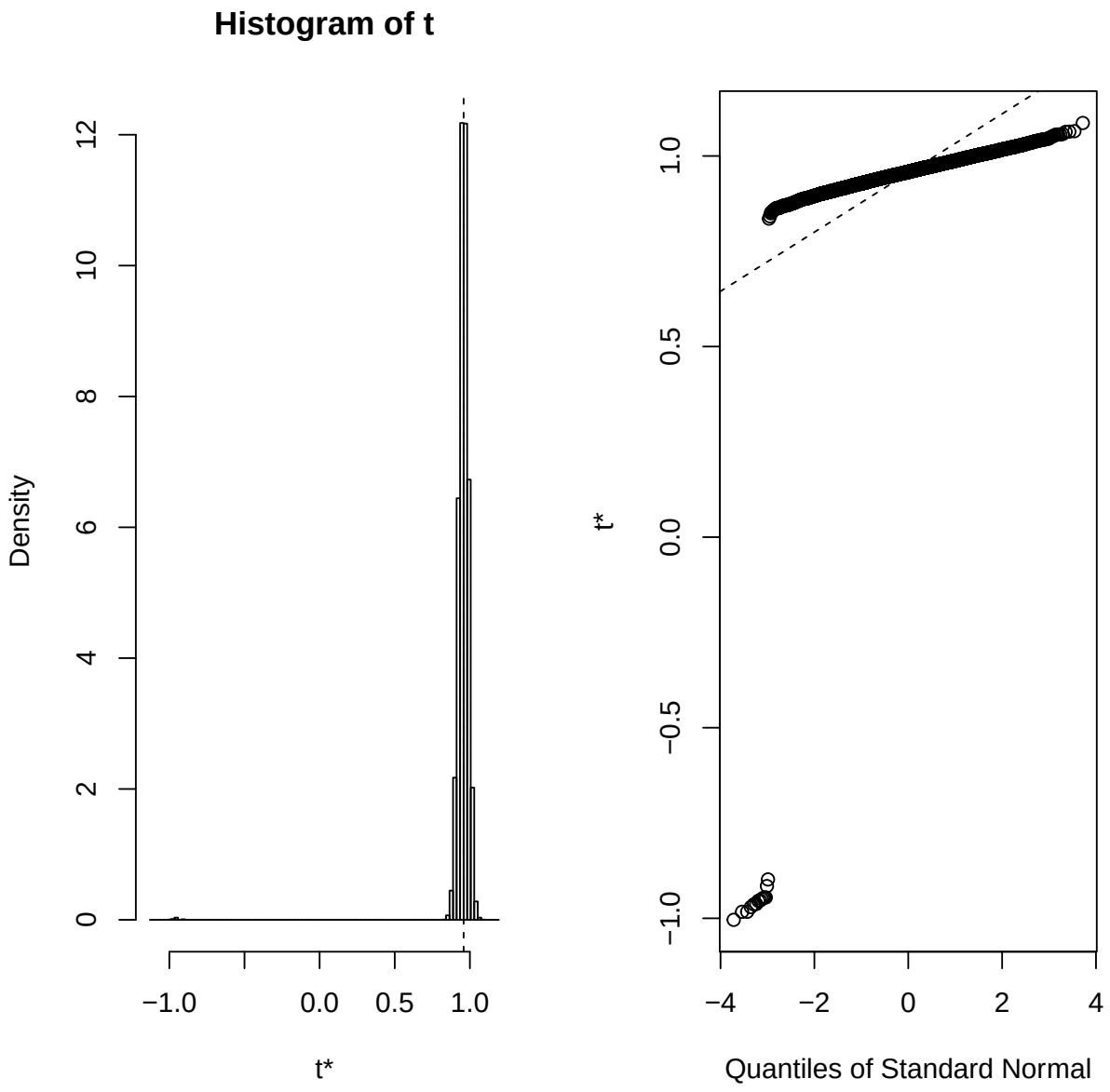

```
plot(boot_results, index = 3)
```



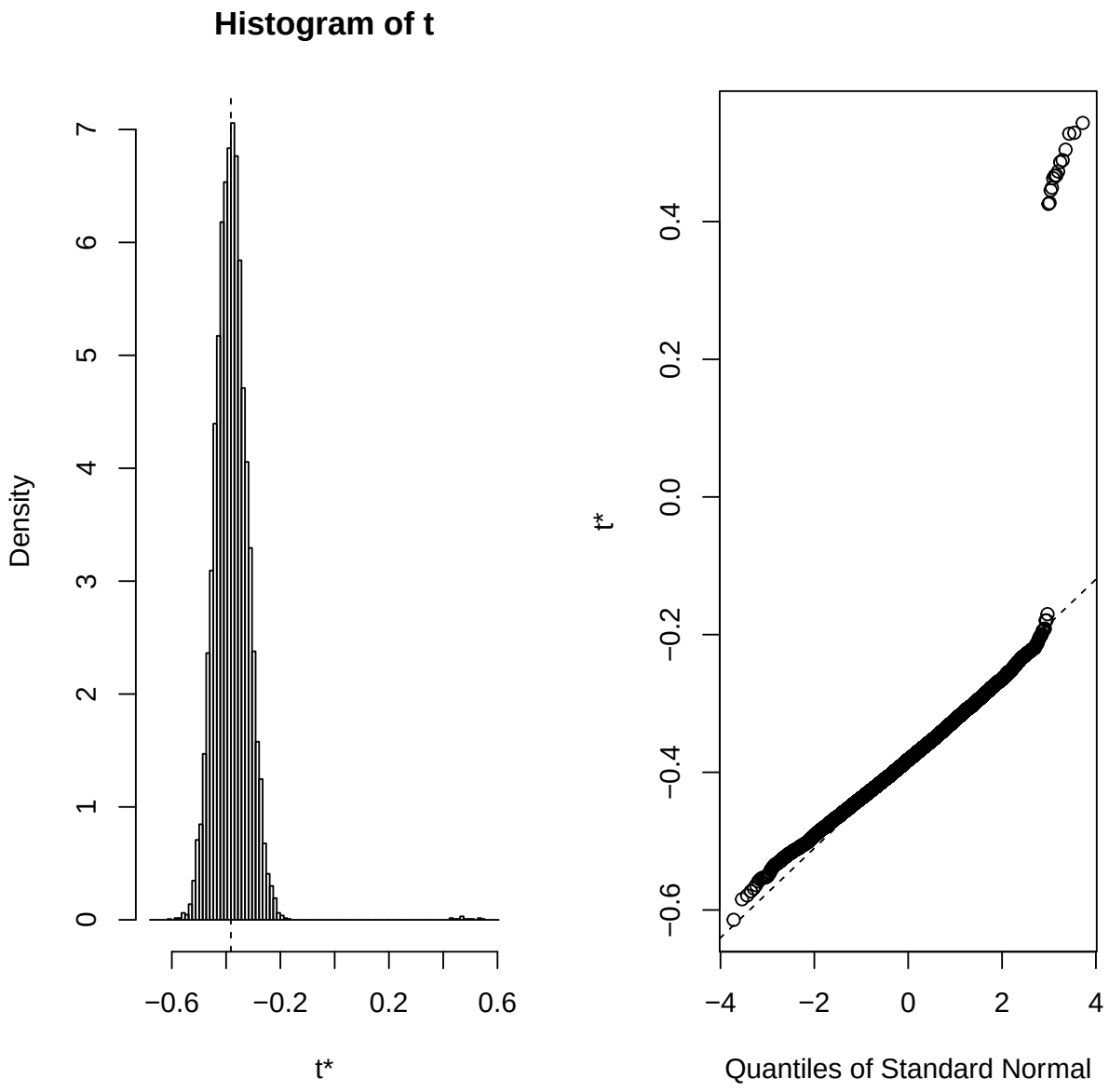

```
plot(boot_results, index = 4)
```



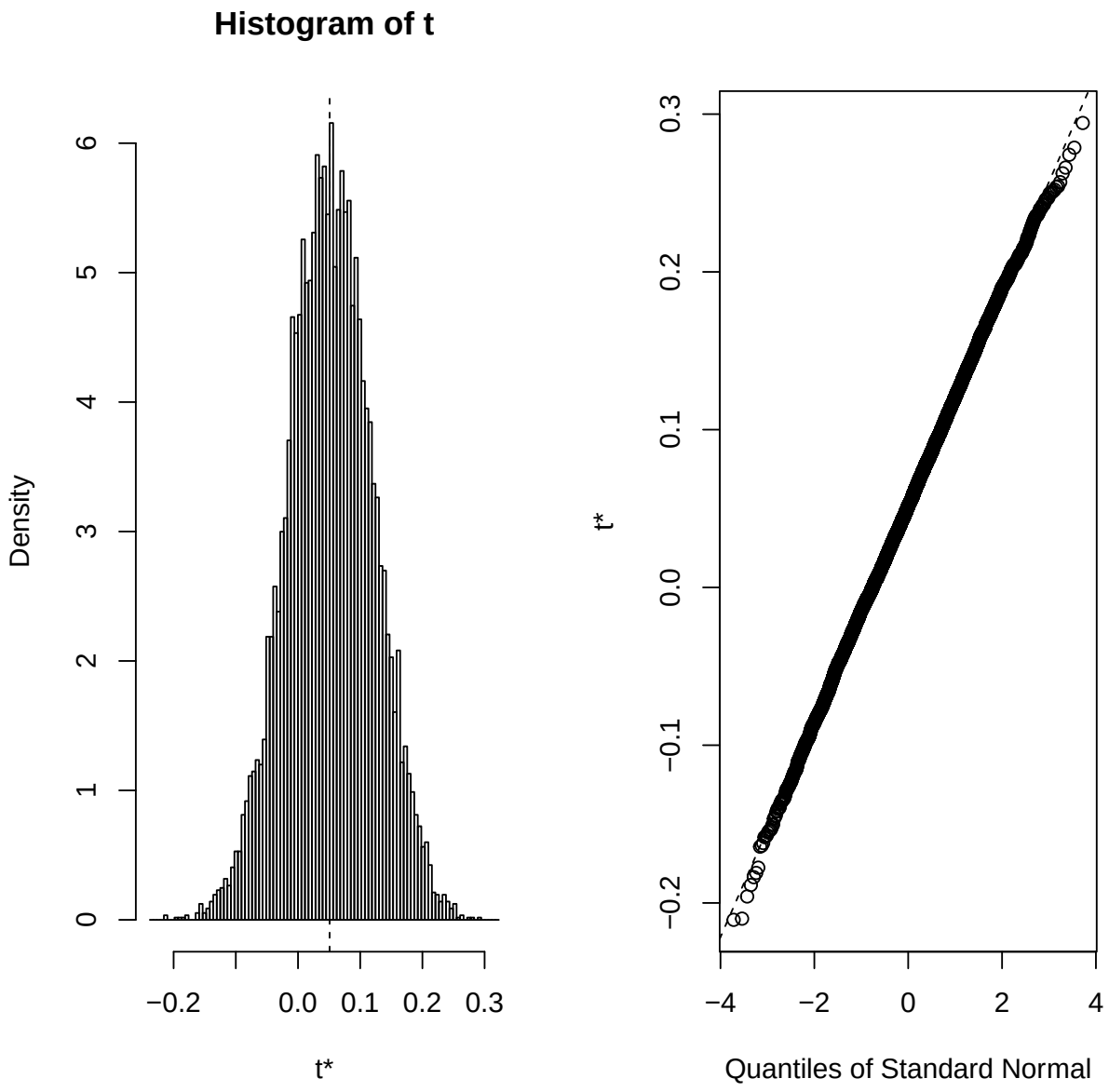

```
plot(boot_results, index = 5)
```



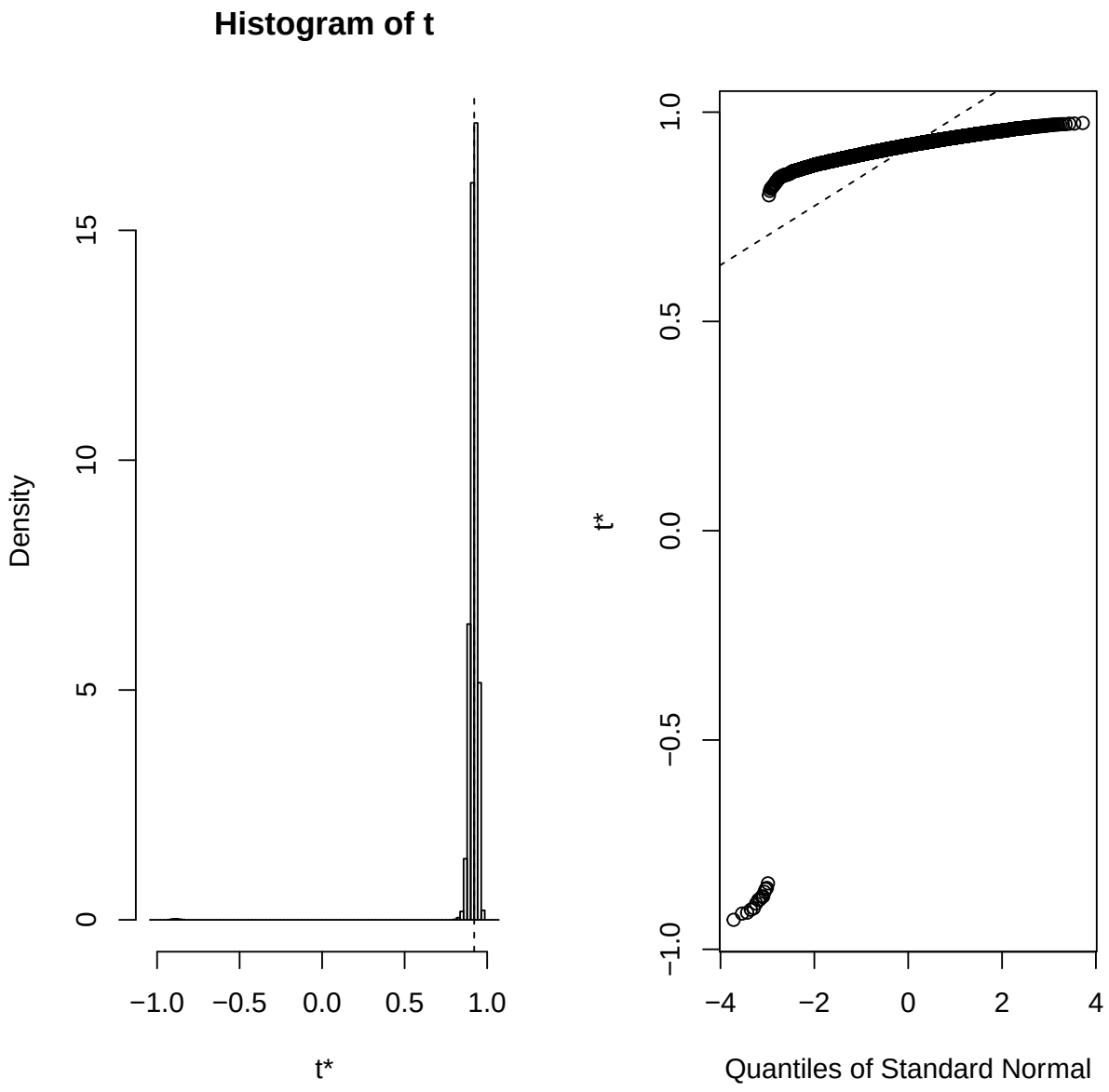

```
plot(boot_results, index = 6)
```



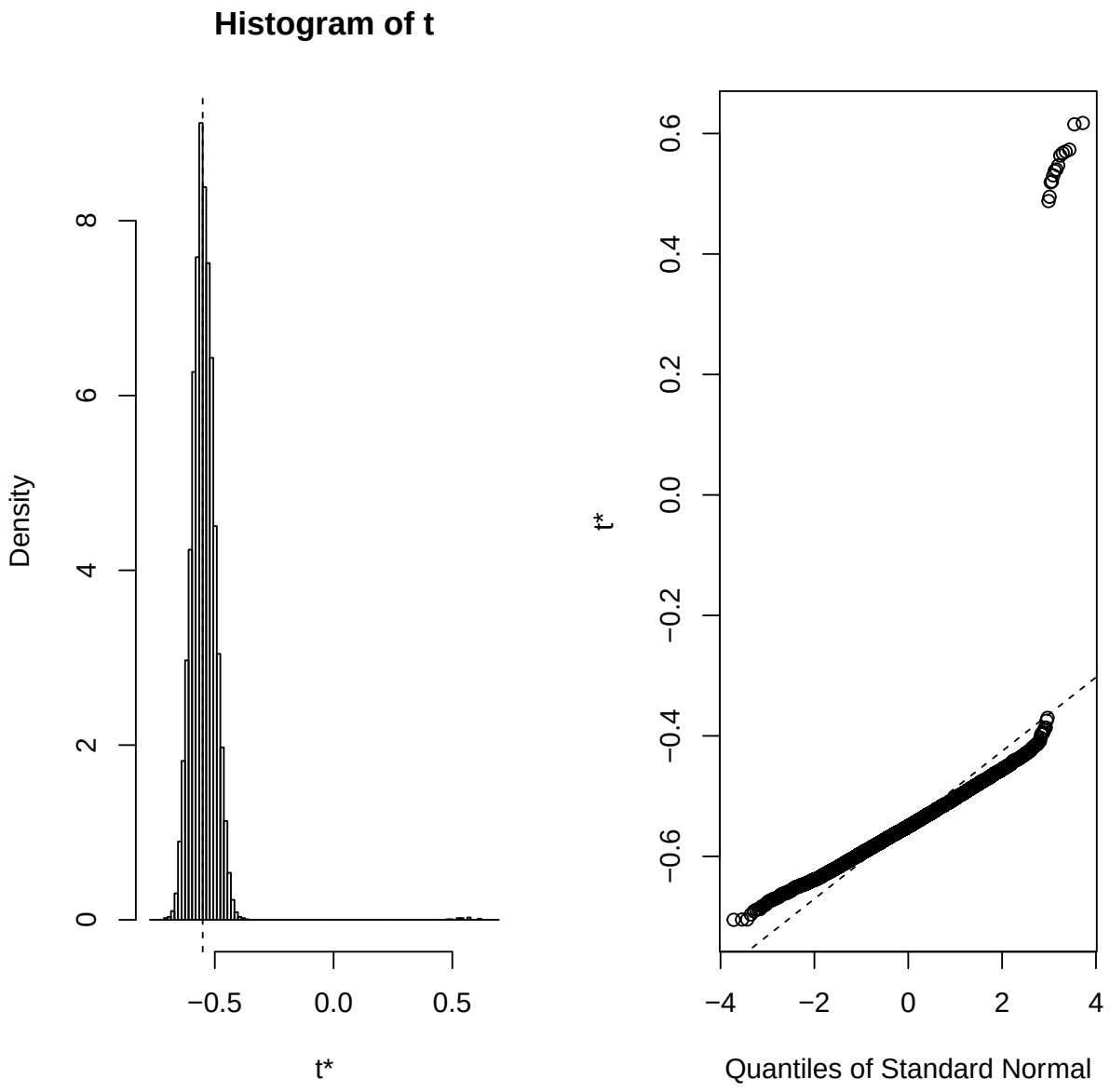

```
plot(boot_results, index = 7)
```



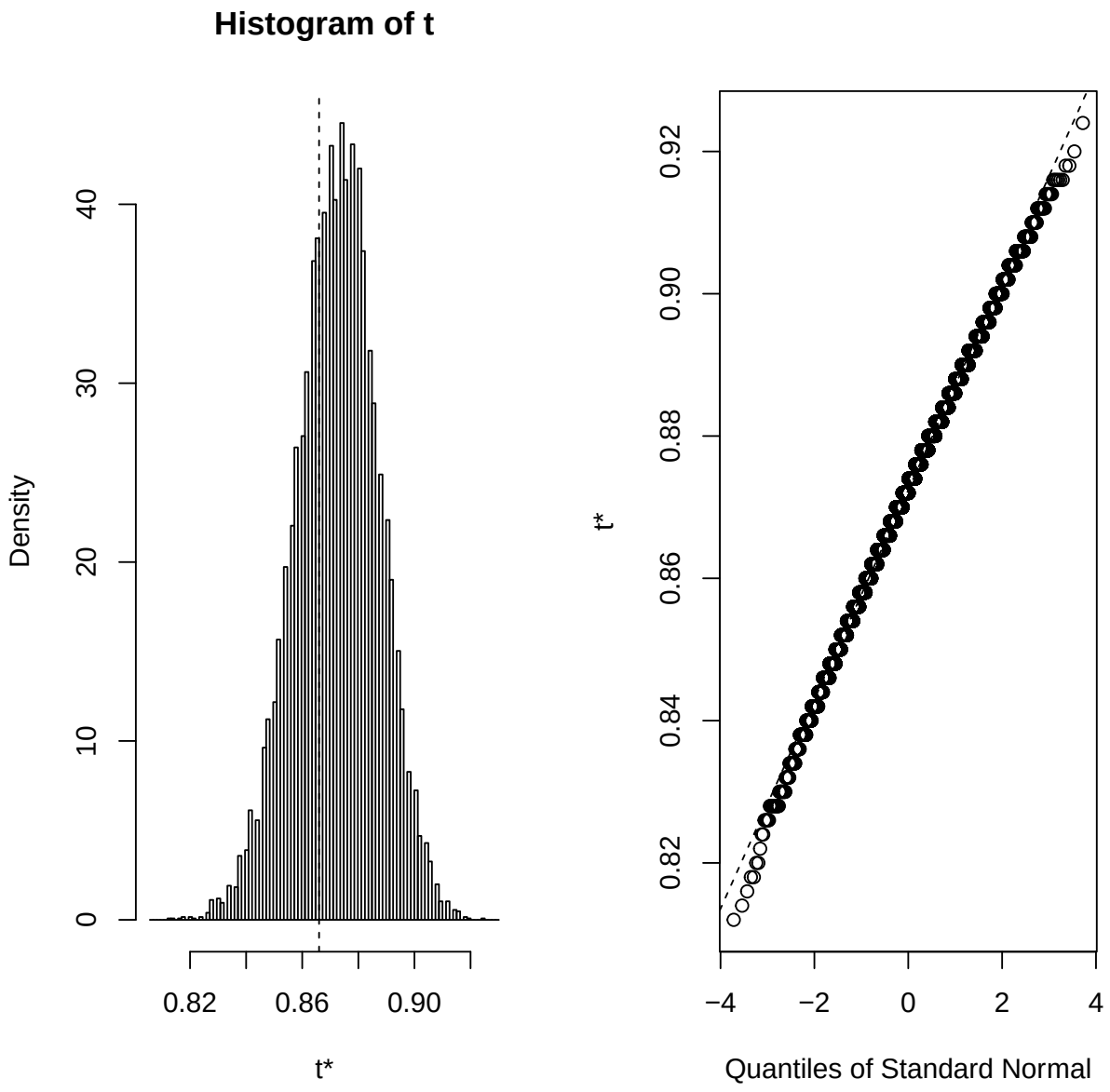

```
plot(boot_results, index = 8)
```



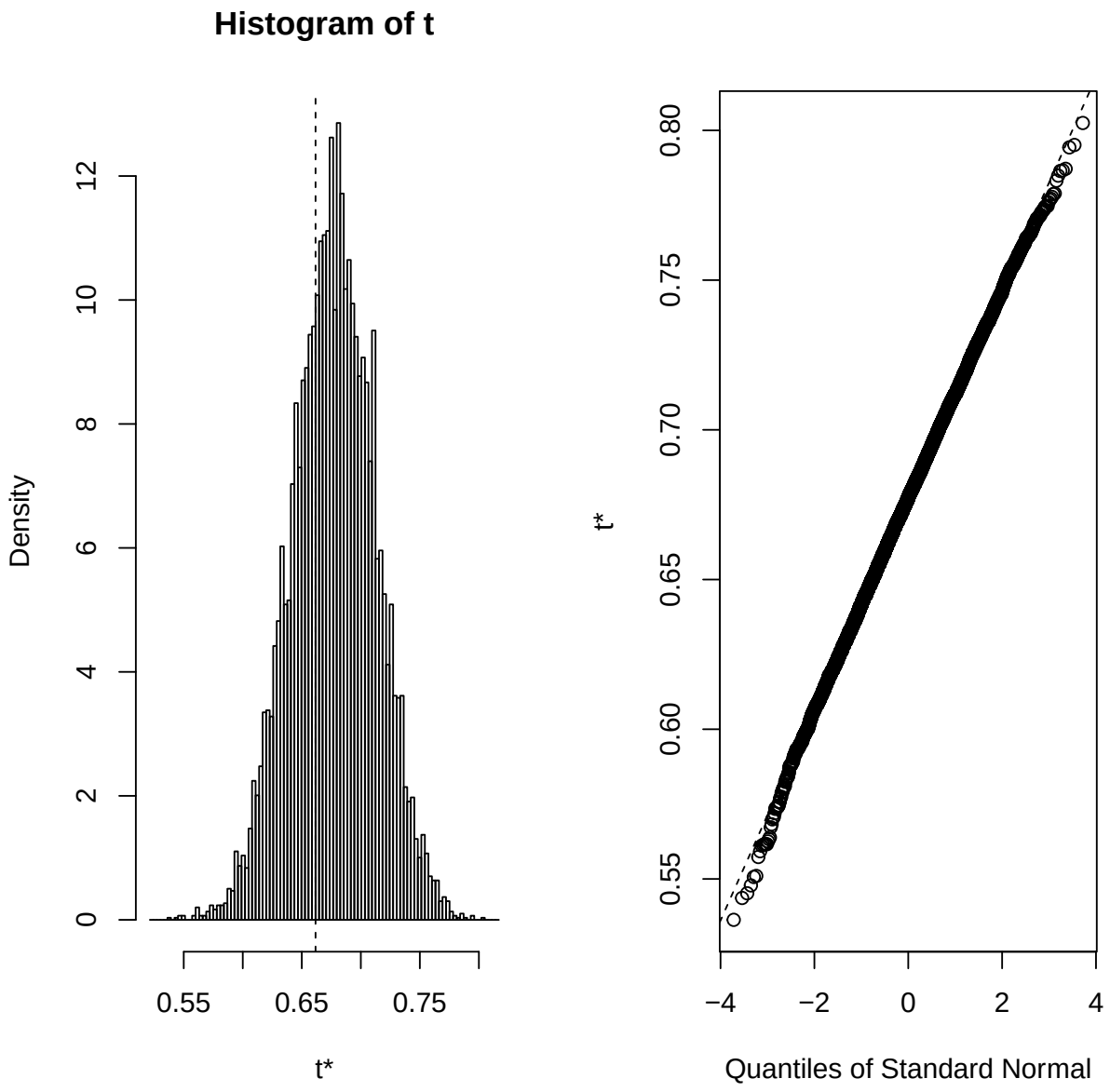

```
plot(boot_results, index = 9)
```




```
plot(boot_results, index = 10)
```




```
plot(boot_results, index = 11)
```



3.1 Confidence Intervals

Several different confidence intervals can be used in bootstrapping. They are extracted and assembled in one matrix here.

```
CI_Boot <- matrix(NA, nrow = 11, ncol = 8)
for (i in seq(1, 4, 1)) {
    for (j in seq(1, 11, 1)) {
        if (i == 1) {
            CI_Boot[j, (i * 2 - 1)] <- boot.ci(boot_results, type = "norm",
                index = j)$normal[2]
            CI_Boot[j, (i * 2)] <- boot.ci(boot_results, type = "norm",
                index = j)$normal[3]
        } else if (i == 2) {
```

```
            CI_Boot[j, (i * 2 - 1)] <- boot.ci(boot_results, type = "basic",
                index = j)$basic[4]
            CI_Boot[j, (i * 2)] <- boot.ci(boot_results, type = "basic",
                index = j)$basic[5]
        } else if (i == 3) {
            CI_Boot[j, (i * 2 - 1)] <- boot.ci(boot_results, type = "perc",
                index = j)$perc[4]
            CI_Boot[j, (i * 2)] <- boot.ci(boot_results, type = "perc",
                index = j)$perc[5]
        } else {
            CI_Boot[j, (i * 2 - 1)] <- boot.ci(boot_results, type = "bca",
                index = j)$bca[4]
            CI_Boot[j, (i * 2)] <- boot.ci(boot_results, type = "bca",
                index = j)$bca[5]
        }
    }
}
```

Bootstrap Confidence 95% Intervals								
	Normal		Basic		Percentile		BCA	
Effect	Lower CL	Upper CL	Lower CL	Upper CL	Lower CL	Upper CL	Lower CL	Upper CL
GRE B	-0.0047	-0.0018	-0.0046	-0.0019	-0.0046	-0.0019	-0.0046	-0.0019
Publications B	0.4266	0.6012	0.4648	0.5593	0.4684	0.5628	0.4643	0.5586
Years B	-0.2684	-0.1301	-0.26	-0.1384	-0.2589	-0.1373	-0.2592	-0.1375
GRE β	-0.4858	-0.1912	-0.4758	-0.1972	-0.4779	-0.1993	-0.4777	-0.1992
Publications β	0.8094	1.1135	0.9009	1.0201	0.8967	1.0159	0.8964	1.0154
Years β	-0.5123	-0.2571	-0.4977	-0.2749	-0.4898	-0.267	-0.4931	-0.2692
GRE Structure	-0.0843	0.1838	-0.0847	0.1847	-0.083	0.1864	-0.0847	0.185
Pub. Structure	0.7883	1.0646	0.8881	0.9682	0.8749	0.955	0.8763	0.9556
Years Structure	-0.6744	-0.434	-0.6444	-0.4644	-0.6378	-0.4577	-0.6378	-0.4577
Classification	0.8308	0.8884	0.832	0.89	0.842	0.9	0.824	0.886
Tau	0.5773	0.7153	0.5782	0.7153	0.608	0.7451	0.5679	0.7132

3.2 Outlier Elimination Function

Occasionally the bootstrapping process will produce extreme outliers due to the vagaries of random sampling. Those outliers can create problems for histograms. This function identifies and eliminates outliers from the data frame to be used in a histogram. It tallies the number of eliminated outliers so that can be indicated on the histogram.

```
outlier_detect <- function(data, Z = 4) {
    data_2a <- matrix(NA, nrow = (length(data[1])))
    data_2b <- matrix(NA, nrow = (length(data[1])))
    data_3 <- scale(data)
    counter_a <- 0
    counter_b <- 0
    for (i in seq(1, length(data), 1)) {
        if (abs(data_3[i]) <= Z) {
            counter_a <- counter_a + 1
            data_2a[counter_a] <- data[i]
        } else {
            counter_b <- counter_b + 1
            data_2b[counter_b] <- data[i]
        }
```

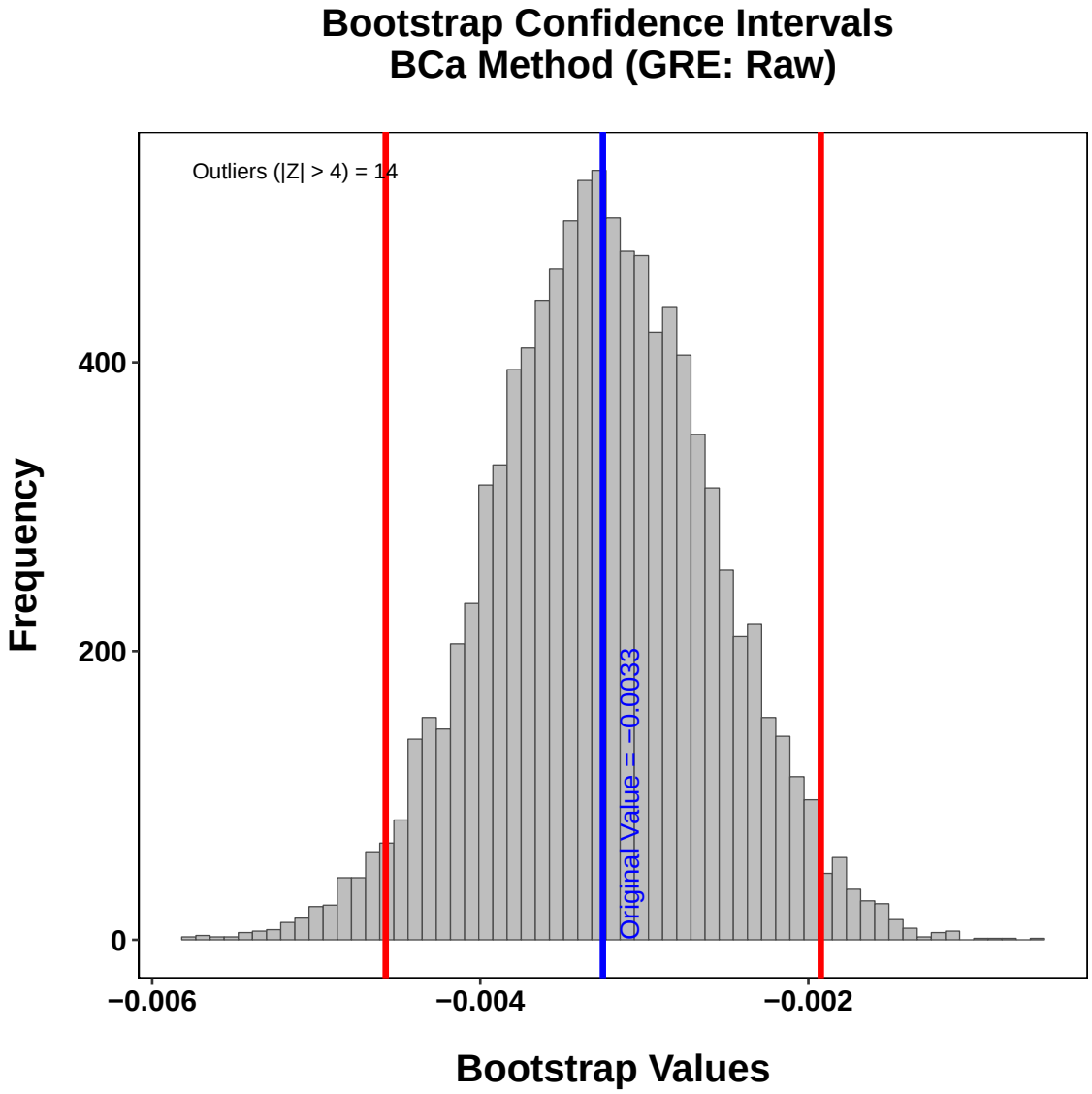
```
    }
    data_2a <- na.omit(data_2a)
    data_2b <- na.omit(data_2b)
    results <- list(data_2a, data_2b, counter_a, counter_b)
    return(results)
}
```

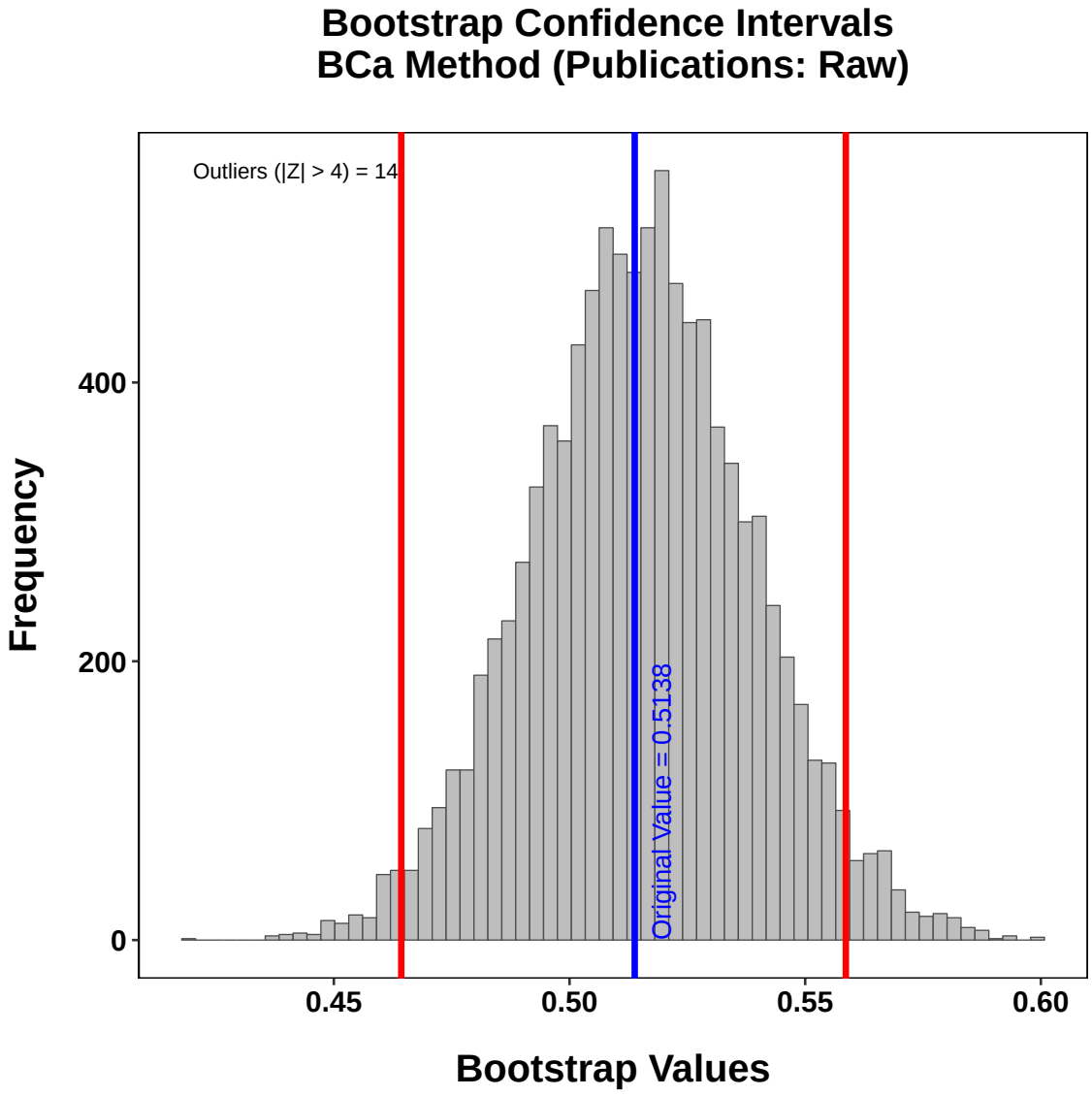
3.3 Simple Bootstrapping with Bias-Corrected and Accelerated Confidence Intervals

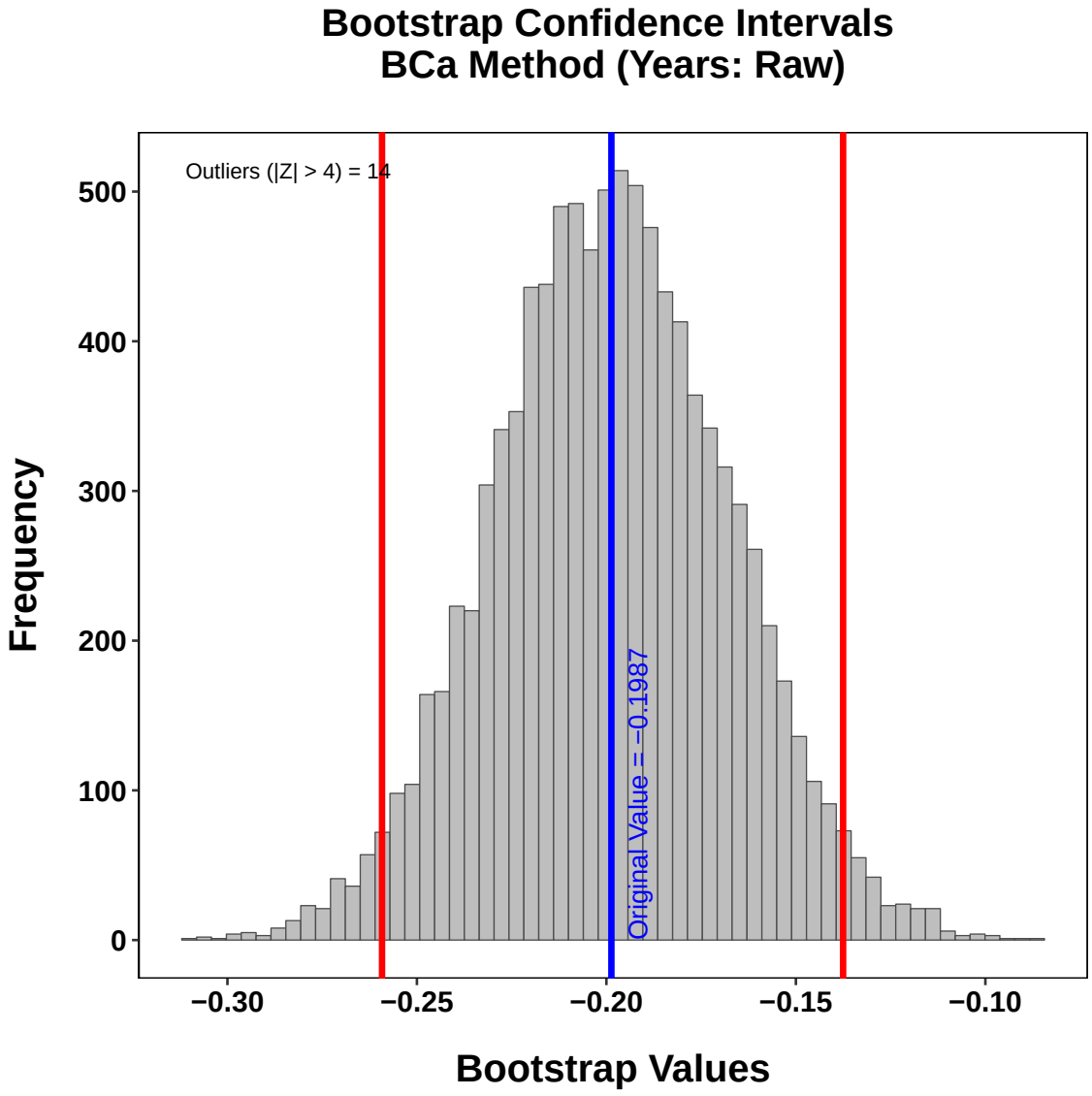
Of the several ways to calculate bootstrap confidence intervals, the bias corrected and accelerated is the most commonly recommended.

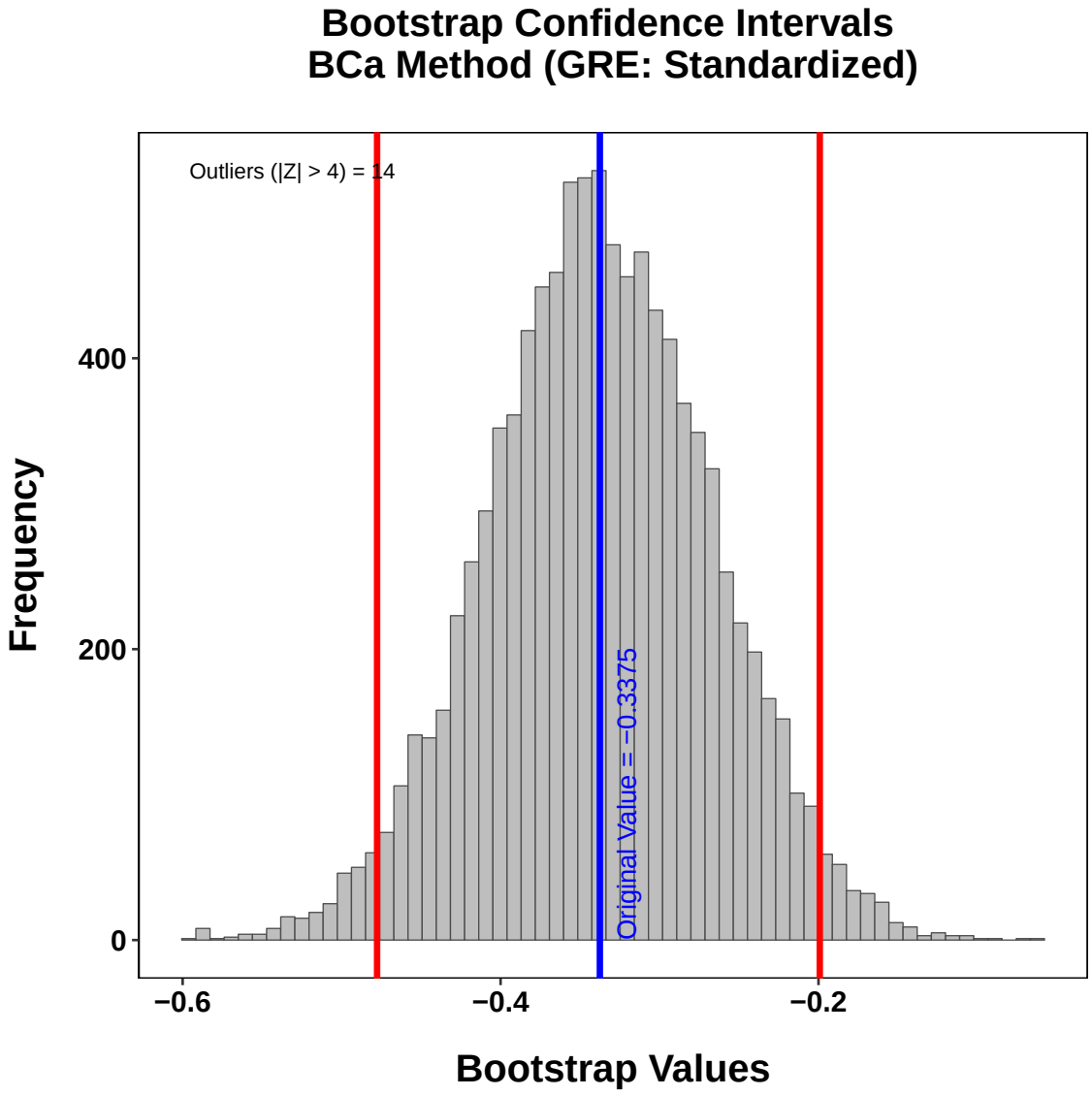
```
Effects <- c("GRE: Raw", "Publications: Raw", "Years: Raw", "GRE: Standardized",
    "Publications: Standardized", "Years: Standardized", "GRE: Structure",
    "Publications: Structure", "Years: Structure", "Prop. Correct",
    "Tau")

Outlier_Z <- 4

# Number of bins specified using the Friedman-Diaconis rule.
for (j in seq(1, 11, 1)) {
    trimmed_data <- outlier_detect(boot_results$t[, j], Outlier_Z)
    plot_data <- as.data.frame(trimmed_data[[1]])
    names(plot_data) <- c("t")
    plot <- ggplot(plot_data, aes(x = t)) + geom_histogram(bins = round((max(plot_data$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("Bootstrap Values") + 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 = boot.ci(boot_results,
        type = "bca", index = j)$bca[4], size = 1.25, color = "red") +
        geom_vline(xintercept = boot.ci(boot_results, type = "bca",
            index = j)$bca[5], size = 1.25, color = "red") + geom_vline(xintercept = boot_results$t0[j],
        size = 1.25, color = "blue") + annotate("text", x = boot_results$t0[j] +
        (max(plot_data$t) - min(plot_data$t))/32, y = 0, label = paste("Original Value = ",
        toString(round(10 * boot_results$t0[j], 3)/10), sep = ""),
        color = "blue", angle = 90, hjust = 0) + annotate("text",
        x = min(plot_data$t), y = max(ggplot_build(plot)$data[[1]]$count),
        label = paste("Outliers (|Z| > ", toString(round(Outlier_Z,
            2)), ") = ", toString(trimmed_data[[4]]), sep = ""), color = "black",
```

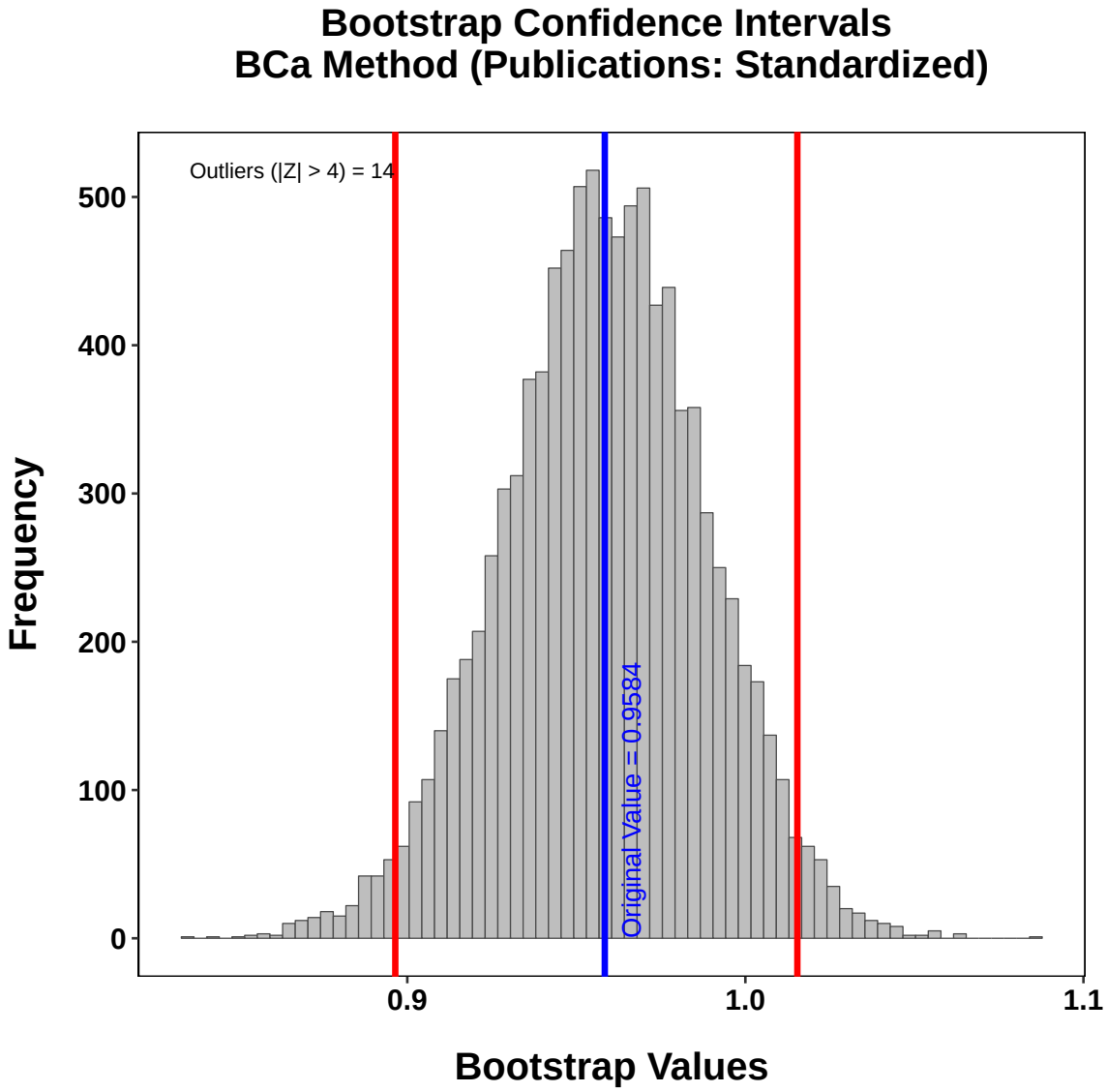
```
        angle = 0, hjust = 0, size = 3) + ggtitle(paste("Bootstrap Confidence Intervals \nBCa Method (",
        toString(Effects[j]), ")", sep = ""))
    print(p)
}
```

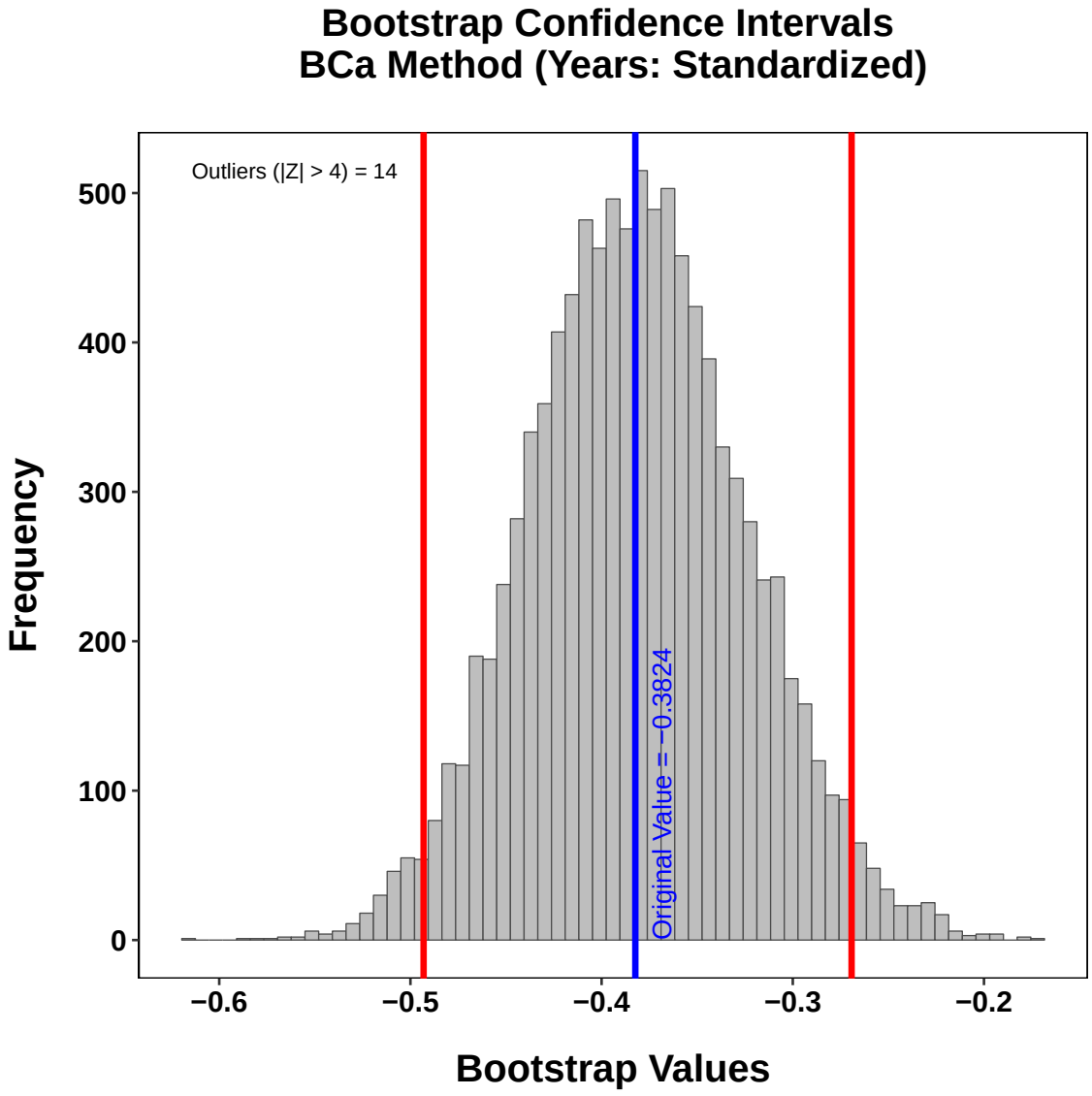


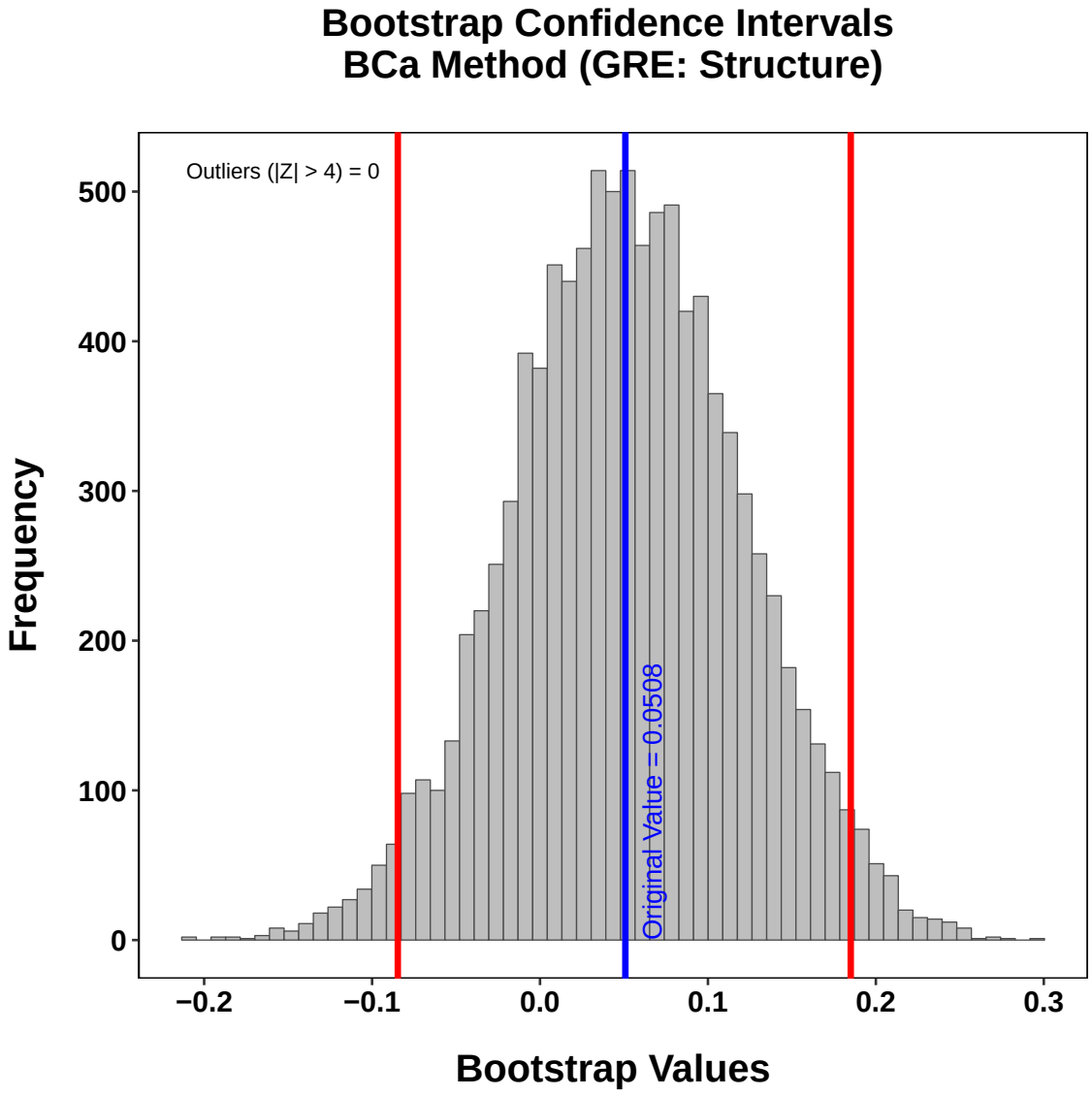

Bootstrap Confidence Intervals
BCa Method (Publications: Raw)
Outliers (|Z| > 4) = 14
Original Value = 0.5138
Frequency
400
200
0
0.45
0.50
0.55
0.60
Bootstrap Values

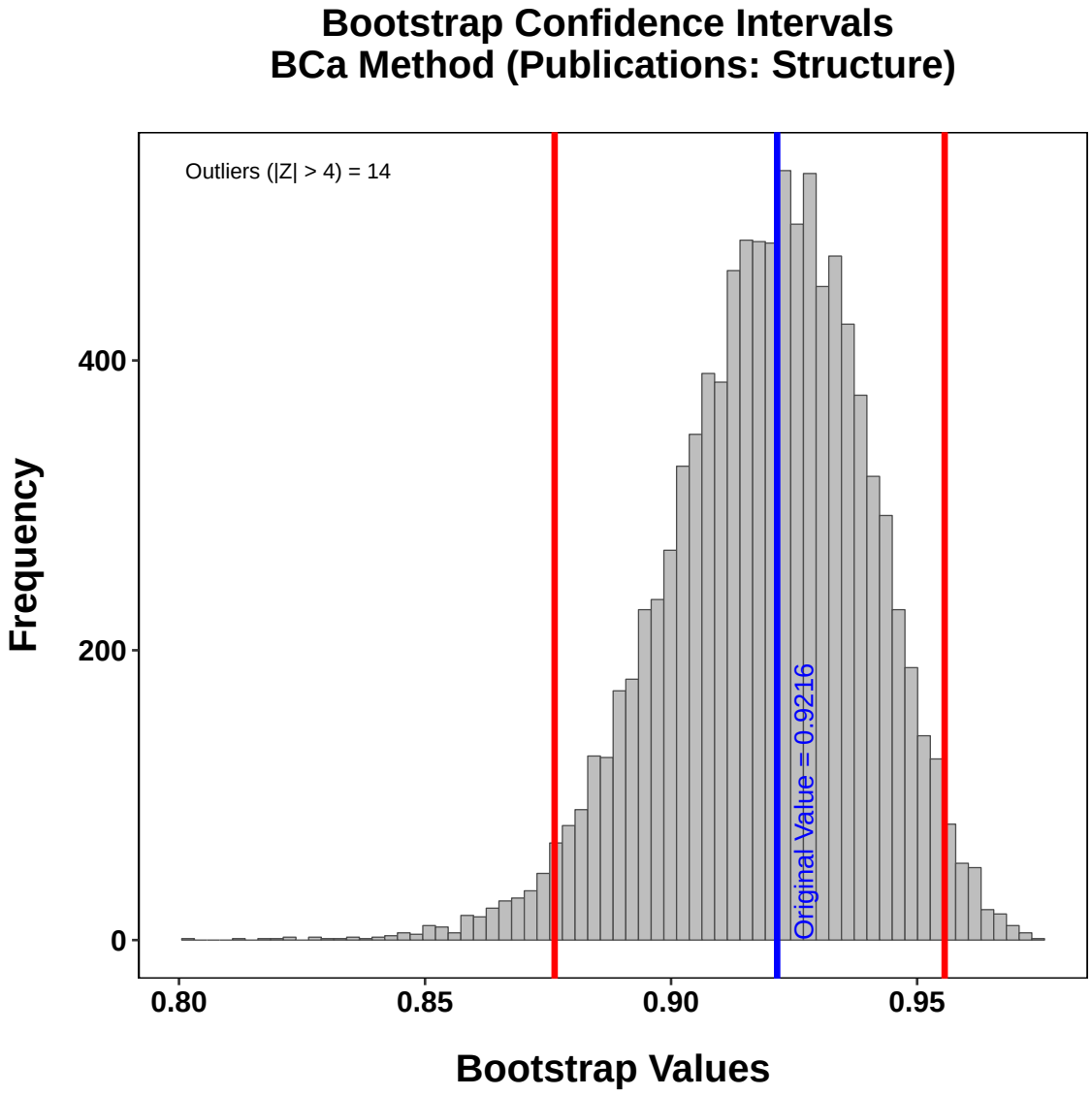

Bootstrap Confidence Intervals
BCa Method (Years: Raw)
Outliers (|Z| > 4) = 14
Original Value = −0.1987
Frequency
0
100
200
300
400
500
−0.30
−0.25
−0.20
−0.15
−0.10
Bootstrap Values

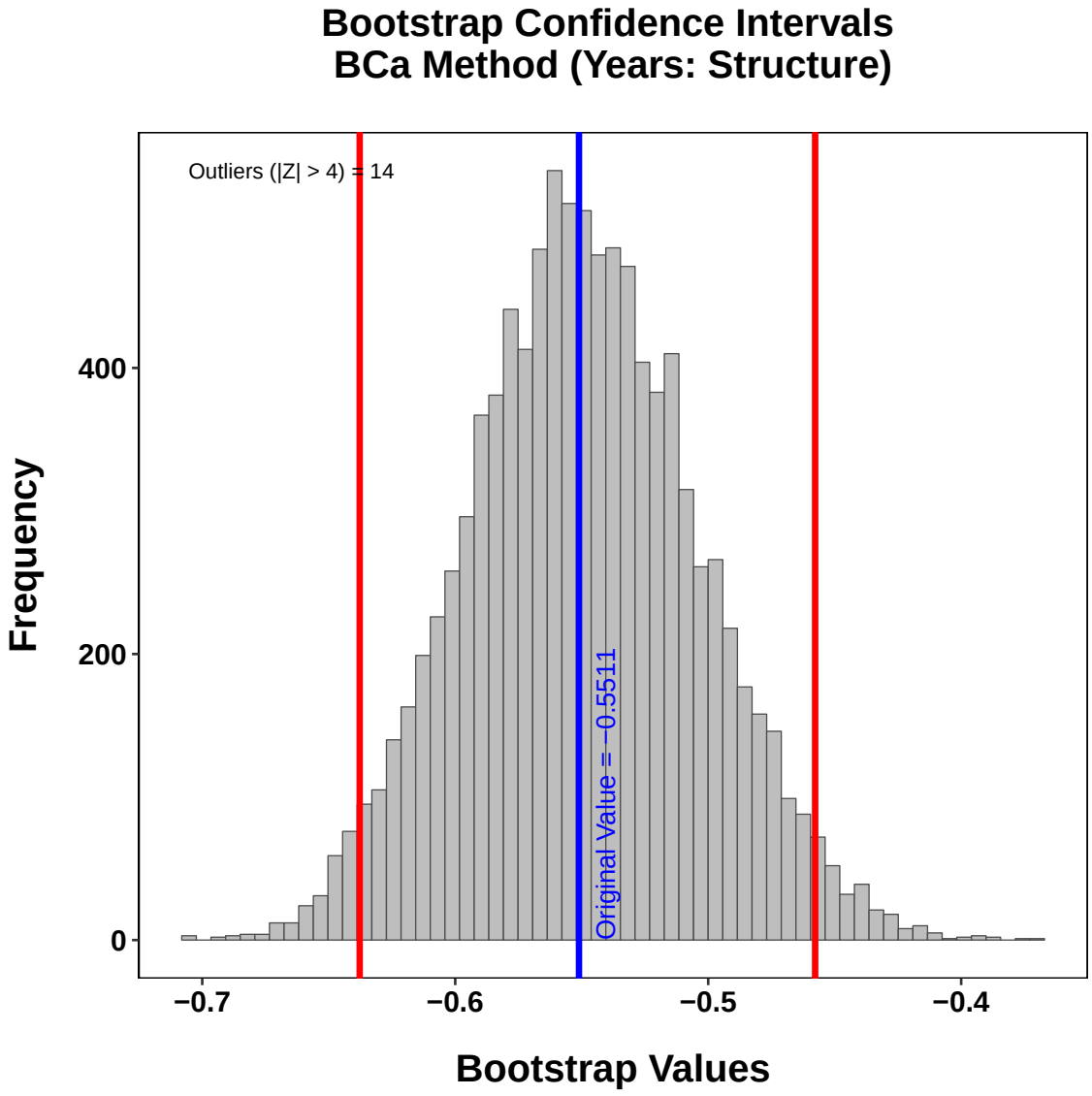

Bootstrap Confidence Intervals
BCa Method (GRE: Standardized)
Outliers (|Z| > 4) = 14
Original Value = −0.3375
Frequency
400
200
0
−0.6
−0.4
−0.2
Bootstrap Values

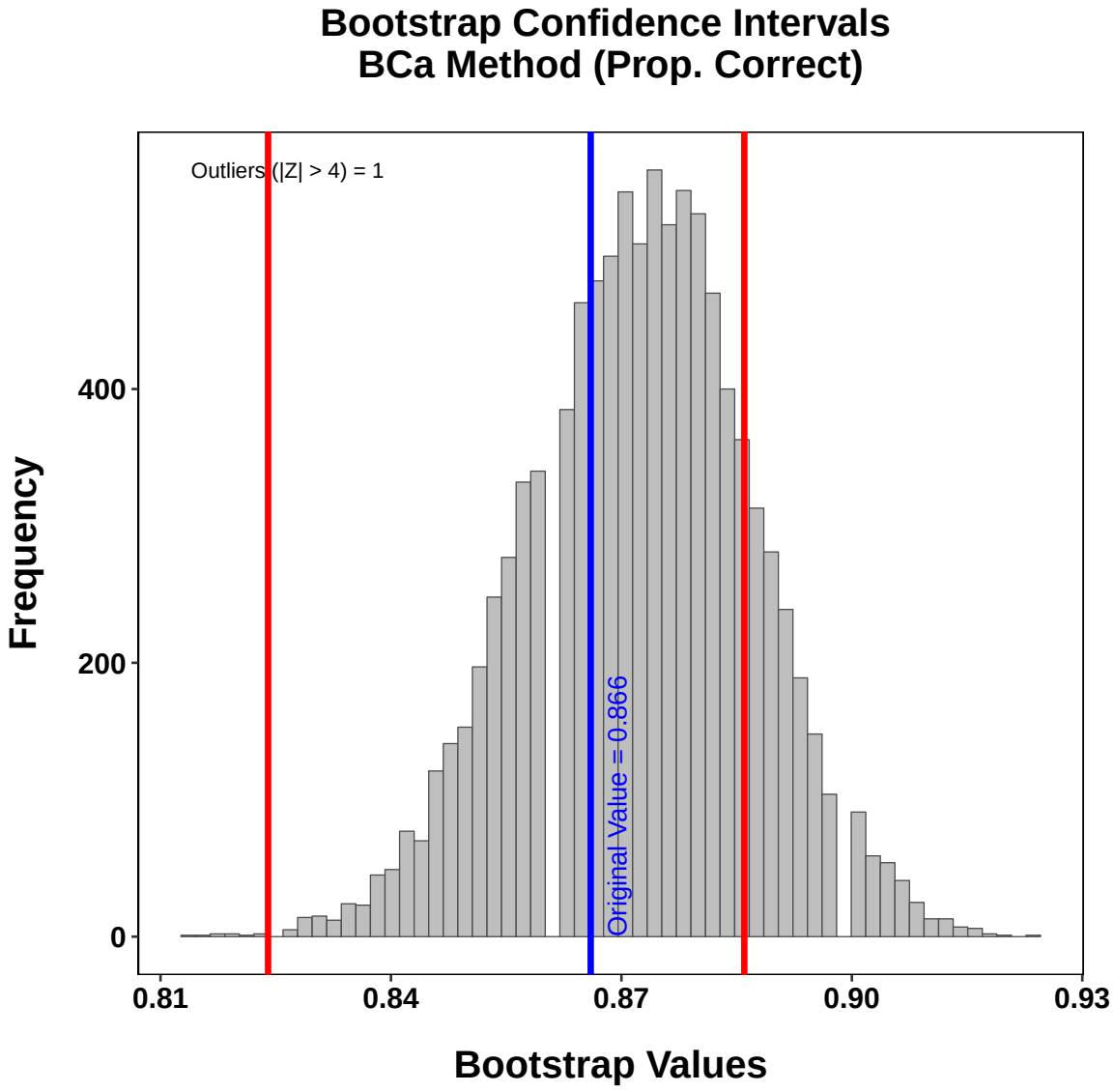

Bootstrap Confidence Intervals
BCa Method (Publications: Standardized)
Outliers (|Z| > 4) = 14
Original Value = 0.9584
Frequency
0
100
200
300
400
500
0.9
1.0
1.1
Bootstrap Values

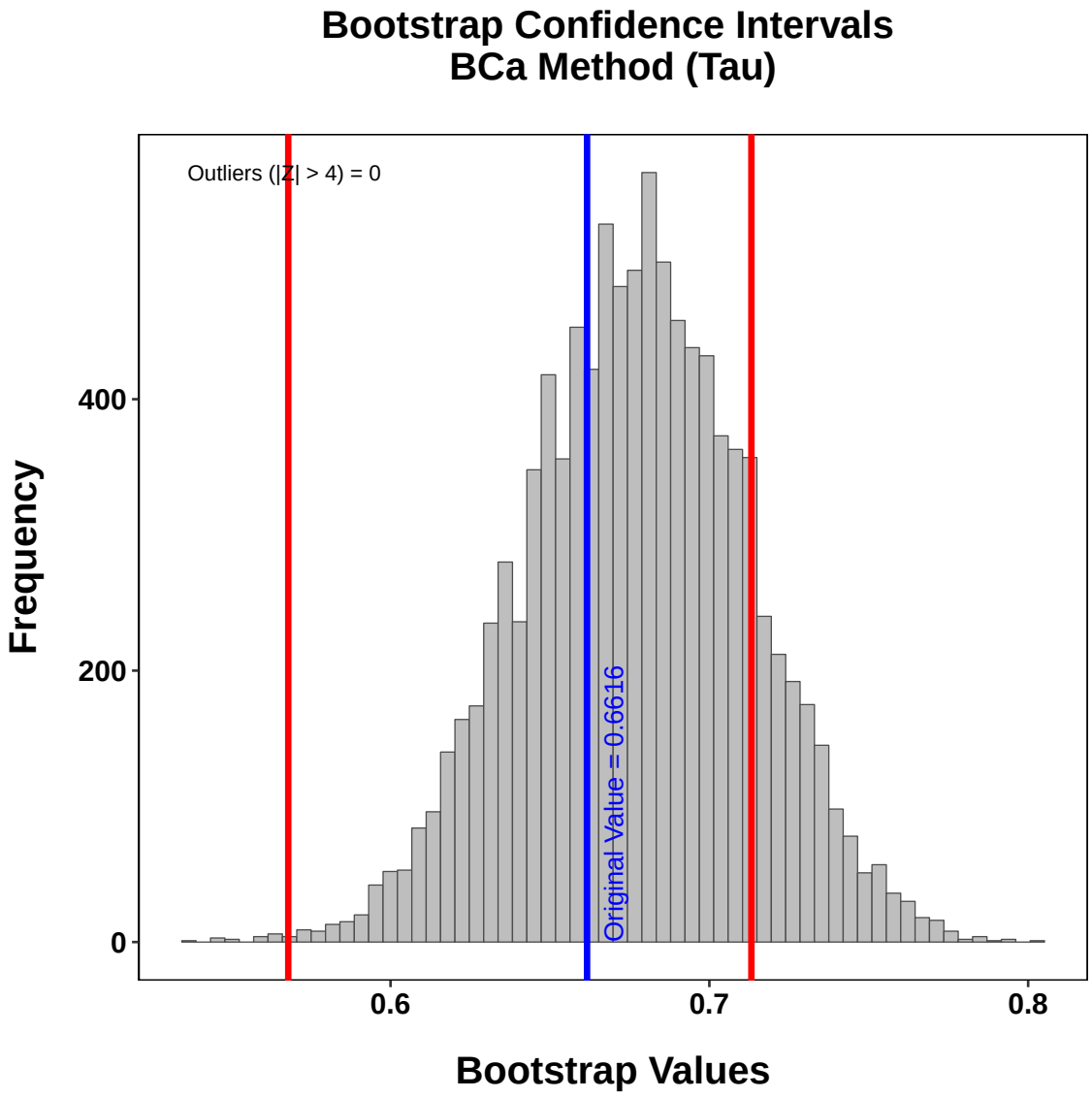

Bootstrap Confidence Intervals
BCa Method (Years: Standardized)
Outliers (|Z| > 4) = 14
Original Value = −0.3824
Frequency
0
100
200
300
400
500
−0.6
−0.5
−0.4
−0.3
−0.2
Bootstrap Values


Bootstrap Confidence Intervals
BCa Method (GRE: Structure)
Outliers (|Z| > 4) = 0
Original Value = 0.0508
Frequency
0
100
200
300
400
500
−0.2
−0.1
0.0
0.1
0.2
0.3
Bootstrap Values


Bootstrap Confidence Intervals
BCa Method (Publications: Structure)
Outliers (|Z| > 4) = 14
Original Value = 0.9216
Frequency
400
200
0
0.80
0.85
0.90
0.95
Bootstrap Values


Bootstrap Confidence Intervals
BCa Method (Years: Structure)
Outliers (|Z| > 4) = 14
Original Value = −0.5511
Frequency
0
200
400
−0.7
−0.6
−0.5
−0.4
Bootstrap Values


Bootstrap Confidence Intervals
BCa Method (Prop. Correct)
Outliers (|Z| > 4) = 1
Original Value = 0.866
Frequency
400
200
0
0.81
0.84
0.87
0.90
0.93
Bootstrap Values




```
Sys.time() - how_long

## Time difference of 7.833 mins
```