Psychometrics

Part 3: Classical Test Theory

SALVADOR CASTRO

Contents

1 Classical Test Theory (CTT)

Classical (or weak, true-score) test theory is a body of related psychometric theory that predicts outcomes of psychological testing such as the difficulty of items or the ability of test-takers. CTT predicts outcomes of psychological testing such as the difficulty of items or the ability of test-takers in order to understand and improve the reliability of tests. CTT assumes that each person has a true score that would be obtained if there were no errors in measurement. A person's true score is defined as the expected number-correct score over an infinite number of independent administrations of the test. Test users never observe a person's true score, but only an observed score. Classical true-score theory describes how errors of measurement can influence observed scores under certain assumptions:

1. $X = \tau + \varepsilon$ (an observed test-score of a person is the sum of that persons true score and an error of measurement)
2. $E[X] = \tau$ (the expected value of observed scores is the true score)
3. $\rho_{\varepsilon\tau} = 0$ (the error of measurement on a test and the true scores on that test are uncorrelated)
4. $\rho_{\varepsilon_1\varepsilon_2} = 0$ (the error of measurement on a test and the true scores onare uncorrelated)
5. $\rho_{\varepsilon_1\tau_2} = 0$ (the error of measurement on a test and the true scores on all other tests are uncorrelated)
6. If two tests have observed scores X and X' that satisfy assumptions 1-5, and if, for every population of examinees, $\tau = \tau'$ and $\sigma^2_{\varepsilon} = \sigma^2_{\varepsilon'}$, then the tests are called **parallel tests**. In other words, parallel tests have the same true scores and eror variances.
7. If two tests have observed scores $\mathbf{X_1}$ and $\mathbf{X_2}$ that satisfy assumptions $1 - 5$, and if, for every population of examinees, $\tau_1 = \tau_2 + c$, where c is a constant, then the tests are called **essentially τ - equivalent tests**. To put it differently, essentially τ - equivalent tests have true scores that differ by a constant.

According to Fan (1998) some of the advantages of CTT are: Its relatively weak theoretical assumptions, which make CTT easy to apply in many testing situations (Hambleton & Jones, 1993 as in Fan, 1998). Although CTTs major focus is on test-level information, item statistics, such as item difficulty and item discrimination are also an important part of the CTT model (Fan 1998). The CTT model is relatively simple compared to IRT. CTT does not require a complex theoretical model to relate an examinees ability to success on a particular item (Fan 1998).

On the other hand CTT has some shortcomings, which can be summarized as circular dependency: The person statistic such as observed score and the item statistics such as item difficulty and item discrimi- nation are sample dependent (Fan 1998). This circular dependency poses some theoretical difficulties in some measurement situations such as test equating, computerized adaptive testing (Fan, 1998). Despite its theoretical weaknesses, measurement experts have worked out practical solutions within the framework of CTT (Fan 1998). For instance, test equating can be accomplished empirically within the CTT framework by equipercentile equating (Fan 1998). Similarly, item-invariant measurement can be accomplished by Thurstone absolute scaling (Englehard, 1990 as in Fan, 1998).

Preamble

```
# clear memory
rm(list = ls())
ls() # check memory
```

```
## character(0)
```

```
# set the default working directory
setwd("/Users/salvadorcastro/Desktop/R Files/CTT")

# the number of digits to print
options(digits = 5)

# turn off warning messages
options(warn = -1)
```

```
# Import Data
ctt.data <- read.csv("test2.csv", header = TRUE)
str(ctt.data)
```

```
## 'data.frame':    1720 obs. of  26 variables:
##  $ Gender: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V1    : int  1 1 1 0 1 0 0 1 0 1 ...
##  $ V2    : int  1 1 1 1 1 1 0 1 1 1 ...
##  $ V3    : int  1 1 0 0 1 1 1 1 1 1 ...
##  $ V4    : int  1 1 0 1 1 0 1 1 0 1 ...
##  $ V5   : int  0 0 1 1 1 0 0 1 0 1 ...
##  $ V6   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V7   : int  1 1 1 0 0 0 1 1 1 1 ...
##  $ V8   : int  0 1 0 0 1 0 0 0 0 0 ...
##  $ V9   : int  0 1 0 0 1 0 1 1 0 0 ...
##  $ V10  : int  1 1 1 0 1 0 0 1 1 1 ...
##  $ V11  : int  1 1 1 0 1 1 1 1 1 0 ...
##  $ V12  : int  0 1 0 0 1 1 1 1 0 1 ...
##  $ V13  : int  0 0 1 0 1 1 0 1 0 1 ...
##  $ V14  : int  0 1 1 0 1 0 1 1 0 1 ...
##  $ V15  : int  1 1 0 0 1 0 1 1 1 0 ...
##  $ V16  : int  1 0 0 1 1 0 0 0 0 0 ...
##  $ V17  : int  1 0 0 0 1 0 0 1 1 0 ...
##  $ V18  : int  1 1 0 0 1 0 0 1 0 0 ...
##  $ V19  : int  1 1 1 0 1 1 0 1 1 1 ...
##  $ V20  : int  1 1 0 0 1 0 1 1 0 1 ...
##  $ V21  : int  1 1 1 0 1 0 0 1 0 0 ...
##  $ V22  : int  1 1 1 1 1 1 1 1 0 1 ...
##  $ V23  : int  1 0 0 1 1 0 0 0 0 1 ...
##  $ V24  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V25  : int  1 1 0 0 1 0 0 1 1 1 ...
```

```
# responses only
responses <- as.matrix(ctt.data[,-1]) # drop column 1 (gender)
dimnames(responses) <- NULL
(N <- dim(responses)[2]) # number of items
```

```
## [1] 25
```

```
(K <- dim(responses)[1]) # number of test-takers
```

```
## [1] 1720
```

1.1 Reliability Analysis

Item analysis within the classical approach often relies on two statistics for evaluating single items: the P-value and the point-biserial correlation coefficient. The P-value represents the proportion of examinees responding in the keyed direction, and is typically referred to as item difficulty. The point-biserial correlation coefficient is a particular item's correlation with all of the other items and it provides an index of differentiating power of the item, which is typically referred to as item discrimination.

1.1.1 Internal Consistency of Tests

Internal consistency is an estimation of test-retest reliability based on the correlation among the variables (items).

1.1.1.1 Cronbach's α **Cronbach's α** (Cronbach 1951) is a coefficient of **internal consistency**. It is commonly used as an estimate of the **reliability** of a psychometric test for a sample of examinees, which provides a convenient index of overall test quality in a single number.

$$\alpha = \left[\frac{N}{N-1}\right]\left[\frac{\sigma_X^2 - \sum_{i=1}^{N}\sigma_{Y_i}^2}{\sigma_X^2}\right]$$

Cronbach's α	Internal consistency
$\alpha \geq 0.9$	Excellent (High-Stakes testing)
$0.7 \leq \alpha < 0.9$	Good (Low-Stakes testing)
$0.6 \leq \alpha < 0.7$	Acceptable
$0.5 \leq \alpha < 0.6$	Poor
$\alpha < 0.5$	Unacceptable

Compute means and variances of test items

```r
meansandvars <- rbind(colMeans(responses), apply(responses, 2, var))
rownames(meansandvars) <- c("means", "vars")
t(meansandvars)
```

```
##         means     vars
##  [1,] 0.76337 0.180740
##  [2,] 0.69651 0.211506
##  [3,] 0.60988 0.238064
##  [4,] 0.53895 0.248627
##  [5,] 0.49535 0.250124
##  [6,] 0.98721 0.012634
##  [7,] 0.57674 0.244252
##  [8,] 0.37500 0.234511
##  [9,] 0.36105 0.230826
## [10,] 0.63256 0.232564
## [11,] 0.59826 0.240486
## [12,] 0.56802 0.245516
## [13,] 0.35291 0.228496
## [14,] 0.67035 0.221110
## [15,] 0.38547 0.237020
## [16,] 0.35930 0.230338
## [17,] 0.39012 0.238064
## [18,] 0.52616 0.249461
## [19,] 0.67558 0.219299
## [20,] 0.56163 0.246345
## [21,] 0.52442 0.249549
## [22,] 0.94477 0.052212
## [23,] 0.63547 0.231784
## [24,] 0.93198 0.063433
## [25,] 0.63430 0.232098
```

Function: Computes Cronbach's alpha

```
cronbachs.alpha <-
  function(X){

    X <- data.matrix(X)
    n <- ncol(X)

    # Cronbachs alpha
    return((n/(n-1))*(1 - sum(apply(X, 2, var))/var(rowSums(X))))
  }

dump("cronbachs.alpha", file = "cronbachs.alpha.R")

# compute cronbachs alpha
```

```r
cronbachs.alpha(responses)
```

```
## [1] 0.82384
```

```r
# compate
CTT::reliability(responses)
```

```
##
##  Number of Items
##  25
##
##  Number of Examinees
##  1720
##
##  Coefficient Alpha
##  0.824
```

1.1.1.2 Kuder-Richardson formula 20 (KR20)

$$KR20 = \left[\frac{k}{k-1}\right]\left[\frac{\sigma_X^2 - \sum_{i=1}^{k} p_i(1-p_i)}{\sigma_X^2}\right]$$

Function: Computes Kuder-Richardson formula 20

```r
# formula 20
KR20 <-
  function(X){
    X <- data.matrix(X)
    k <- ncol(X)

    # Person total score variances
    SX <- var(rowSums(X))

    # item means
    IM <- colMeans(X)
    return(((k/(k - 1))*((SX - sum(IM*(1 - IM)))/SX)))
  }
```

```
KR20(responses)
```

```
## [1] 0.82397
```

1.1.1.3 Kuder-Richardson formula 21 (KR21)

$$KR21 = \left[\frac{N}{N-1}\right]\left[\frac{\sigma_X^2 - N\bar{p}(1-\bar{p})}{\sigma_X^2}\right]$$

Function: Computes Kuder-Richardson formula 21

```
# Kuder-Richardson formula 21
KR21 <-
  function(X){
    X <- data.matrix(X)
    n <- ncol(X)

    return((n/(n-1))*((var(rowSums(X)) - n*(sum(colMeans(X))/n) *
                         (1-(sum(colMeans(X))/n))))/var(rowSums(X)))
  }
```

```
KR21(responses)
```

```
## [1] 0.792
```

1.1.2 Split-half (Test-Retest) Reliability

The test-retest reliability is an estimation of reliability based on the correlation of two equivalent forms of tests. Correlation between the split-halves is a reasonable measure of the reliability of one half of the test. Reliability of the entire test would be greater than the reliability of either half taken alone.

Function: Split data (cases) into two about equal-sized random sets

```r
SPLIT.CASES <-
  function(X, seed = NULL){
    # optional fixed seed
    if (!is.null(seed)) {set.seed(seed)}

    X <- as.matrix(X)

    # if k = 2x, then lengths Y1 = Y2
    # if k = 2x+1, then lenths Y1 = Y2+1
    k <- nrow(X)
    index <- sample(1:k, ceiling(k/2))
    Y1 <- X[index, ]
    Y2 <- X[-index, ]
    return(list(Y1, Y2))
  }

dump("SPLIT.CASES", file = "SPLIT.CASES.R")
```

Function: Split data (variables) into two about equal-sized random sets

```r
SPLIT.ITEMS <-
  function(X, seed = NULL){
    # optional fixed seed
    if (!is.null(seed)) {set.seed(seed)}

    X <- as.matrix(X)

    # if n = 2x, then lengths Y1 = Y2
    # if n = 2x+1, then lenths Y1 = Y2+1
    n <- ncol(X)
    index <- sample(1:n, ceiling(n/2))
    Y1 <- X[, index ]
    Y2 <- X[, -index]
    return(list(Y1, Y2))
  }

dump("SPLIT.ITEMS", file = "SPLIT.ITEMS.R")
```

1.1.2.1 Spearman-Brown formula

The Spearman-Brown (Spearman 1910; Brown 1910) formula is used to predict the reliability of a test after changing the test length.

$$\rho_{XX'} = \frac{N\rho_{YY'}}{1 + (N-1)\rho_{YY'}}$$

where

N is the factor by which the length of the test is changed,

$\rho_{XX'}$ is the predicted reliability coefficient, and

$\rho_{YY'}$ is reliability of the original test

```r
# Spearman-Brown formula
SpearmanBrown <-
  function(X, N){
    source("cronbachs.alpha.R")

    X <- as.matrix(X)

    # cronbach's alpha for the original test
    alpha <- cronbachs.alpha(X)
    predicted.alpha <- N * alpha / (1 + (N - 1) * alpha)
    return(list(original.reliability = alpha,
                predicted.reliability = predicted.alpha))
  }

dump("SpearmanBrown", file = "SpearmanBrown.R")

# predict reliability by Spearman-Brown formula
# if the number of items is reduced from 25 to 15
SpearmanBrown(responses, N = 15/25)
```

```
## $original.reliability
## [1] 0.82384
##
```

```
## $predicted.reliability
## [1] 0.73726
```

```
# predict reliability by Spearman-Brown formula
# if the number of items is increased from 25 to 35
SpearmanBrown(responses, N = 35/25)
```

```
## $original.reliability
## [1] 0.82384
##
## $predicted.reliability
## [1] 0.8675
```

```
# predict reliability by Spearman-Brown formula
# if the number of items is doubled
SpearmanBrown(responses, N = 2)
```

```
## $original.reliability
## [1] 0.82384
##
## $predicted.reliability
## [1] 0.90341
```

1.1.2.2 Guttman's Lambda (Guttman 1945) L1: An intermediate coefficient used in computing the other lambdas.

L2: More complex than Cronbach's alpha and preferred by some researchers, though less common.

L3: Equivalent to Cronbach's alpha.

L4: Guttman split-half reliability.

L5: Recommended when a single item highly covaries with other items, which themselves lack high covariances with each other.

L6: Recommended when inter-item correlations are low in relation to squared multiple correlations

Guttman's λ_3 is the same as Cronbach's α (Cronbach 1951) and may be computed by

$$\lambda_3 = \alpha = \frac{K}{K-1}\left(1 - \frac{\sum_{i=1}^{K} \sigma^2_{Y_i}}{\sigma^2_X}\right)$$

If the items are scored 0 and 1, a shortcut formula is

$$\lambda_3 = \alpha = \frac{k}{k-1}\left(1 - \frac{\sum_{i=1}^{k} P_i * Q_i}{\sigma_X^2}\right)$$

where

σ_X^2 is the variance of the observed total test scores, and

$\sigma_{Y_i}^2$ is the variance of component i for the current sample of persons.

P_i is the proportion scoring 1 on item i, and

$Q_i = 1 - P_i$.

Function: Computes Guttman's $lambda_3$

```
G3 <-
  function(responses){

    X <- as.matrix(responses)
    k <- ncol(X) # number of items

    SX <- var(rowSums(X)) # variance of total person scores
    SI <- apply(X, 2, var) # item variances
    L1 <- (1 - (sum(SI)/SX)) # Guttman's lamda 1

    return((k*L1)/(k - 1))
  }

dump("G3", file = "G3.R")

# compute G3
G3(responses)
```

```
## [1] 0.82384
```

```
# compare
psych::guttman(responses)
```

```
## Loading required namespace: GPArotation
## Call: psych::guttman(r = responses)
##
## Alternative estimates of reliability
##
## Guttman bounds
## L1 =  0.79
## L2 =  0.82
## L3 (alpha) =  0.82
## L4 (max) =  0.85
## L5 =  0.81
## L6 (smc) =  0.82
## TenBerge bounds
## mu0 =  0.82 mu1 =  0.82 mu2 =  0.82 mu3 =  0.82
##
## alpha of first PC =  0.83
## estimated greatest lower bound based upon communalities=  0.87
##
## beta found by splitHalf  =  0.76
```

1.1.2.3 Pearson product-moment correlation coefficient

$$\rho_{YY'} = cor(Y, Y')$$

$$= \frac{cov(Y, Y')}{\sigma_Y \sigma_{Y'}}$$

$$= \frac{E\Big[(Y - \mu_Y)(Y' - \mu_{Y'})\Big]}{\sigma_Y \sigma_{Y'}}$$

If we have a series of n measurements of *Y* and *Y'* written as y_i and y_i where *i = 1, 2, . . . , K*, then the sample correlation coefficient can be used to estimate the population Pearson correlation *r* between *Y* and *Y'*.

$$r_{YY'} = \frac{\sum_{i=1}^{K}(y_i - \bar{y}_i)(y_i' - \bar{y}_i')}{(K-1)S_Y S_{Y'}}$$

$$= \frac{\sum_{i=1}^{K}(y_i - \bar{y}_i)(y_i' - \bar{y}_i')}{\sqrt{\sum_{i=1}^{K}(y_i - \bar{y}_i)^2 \sum_{i=1}^{K}(y_i' - \bar{y}_i')^2}}$$

Function: Computes a bootstrapped (1,000 replicates) Pearson product-moment correlation coefficient between two random subsets of examinees.

```r
pearson <-
  function(X, seed = NULL, n = NULL){
    source("SPLIT.ITEMS.R")

    # optional fixed seed
    if (!is.null(seed)) {set.seed(seed)}

    # the number of bootstrap replicates
    if (is.null(n)) {n <- 1e3}

    X <- as.matrix(X)



    r <- rep(NA, n)

    for (i in 1:n) {
      # split items
      Y <- SPLIT.ITEMS(X)

      # total scores
      S1 <- as.matrix(rowSums(Y[[1]]))
      S2 <- as.matrix(rowSums(Y[[2]]))

      # residual scores
      R1 <- S1 - mean(S1)
      R2 <- S2 - mean(S2)
      # Pearson product-moment correlation coefficient
      r[i] <- (t(R1) %*% R2) / (sqrt((t(R1) %*% R1)) * sqrt((t(R2) %*% R2)))
    }

    return(mean(r))
  }

dump("pearson", file = "pearson.R")

# compute the Pearson product-moment correlation coefficient
pearson(responses, seed = 456, n = 1)
```

```
## [1] 0.68635
```

```r
# compare
# split items
set.seed(456)
Y <- SPLIT.ITEMS(responses)

# total scores
S1 <- as.matrix(rowSums(Y[[1]]))
S2 <- as.matrix(rowSums(Y[[2]]))

cor(S1, S2)
```

```
##         [,1]
## [1,] 0.68635
```

When scores are not tau-equivalent (for example when there is not homogeneous but rather examination items of increasing difficulty) then the **KR-20** is an indication of the lower bound of internal consistency (reliability).

KR21 typically underestimates the reliability of a test compared to **KR20**, when the item difficulties are not equal. The formulas will be equal only if the item difficulties are equal.

Cronbach's α and **Kuder-Richardson** (Kuder and Richardson 1937) methods produce a lower bound for a test's reliability. This lower bound equals the test reliability if the split-halves are essentially τ-equivalent. These coefficients should only be used for homegeneous tests, since they reflect item homegeneity, else they will be inappropriately low.

A high KR-20 coefficient (e.g., > 0.90) provides an evidence for the assumption of a homogeneous or unidimensional test. However, it is possible, to have a high KR-20 with a multidimensional scale, especially with a large number of test items.

The Spearman-Brown coefficient can over or underestimate the test reliability if the split-halves are not parallel tests. When the tests are parallel the Spearman-Brown formula is useful to test the effects of test length on reliability

A test can produce different reliability estimates when administered to different samples of examinees. **Generazibility Theory** examines such systematic effects on reliability.

1.2 Standard Error of Measurement

Standard error of measurement using Cronbach's alpha as the reliability statistic cab be computed as

$$S_E = S_X\sqrt{1 - r_{XX'}}$$

```
SEM <-
  function(X){
    source("cronbachs.alpha.R")
    X <- data.matrix(X)

    return(sd(rowSums(X)) * sqrt(1 - cronbachs.alpha(X)))
  }
```

```
SEM(responses)
```

```
## [1] 2.1068
```

1.3 Confidence Intervals for True Scores

A confidence interval for an examinees true score can be constructed as

$$X_i - z_c S_E \leq \tau \leq X_i + z_c S_E$$

For example, the critical-value for a 90% confidence interval is 1.65, so for the first examinee ($i = 1$) with an observed score of 19, the confidence interval is:

$$19 - 1.65 * 2.11 \leq \tau \leq 19 + 1.65 * 2.11$$

$$15.52 \leq \tau \leq 22.48$$

```
# 90% confidence interval for the true score
head(cbind(lower_bound = round(rowSums(responses)-1.65*
                                sd(rowSums(responses))*
                                sqrt(1-KR20(responses)), 2),
           observed = rowSums(responses),
           upper_bound = round(rowSums(responses)+1.65*
                                sd(rowSums(responses))*
                                sqrt(1-KR20(responses)), 2)), 20)
```

```
##       lower_bound observed upper_bound
##  [1,]       15.52       19       22.48
##  [2,]       16.52       20       23.48
##  [3,]        9.52       13       16.48
##  [4,]        4.52        8       11.48
##  [5,]       20.52       24       27.48
##  [6,]        5.52        9       12.48
##  [7,]        8.52       12       15.48
##  [8,]       18.52       22       25.48
##  [9,]        7.52       11       14.48
## [10,]       13.52       17       20.48
## [11,]        8.52       12       15.48
## [12,]        9.52       13       16.48
## [13,]        9.52       13       16.48
## [14,]       15.52       19       22.48
## [15,]       16.52       20       23.48
## [16,]       13.52       17       20.48
## [17,]        8.52       12       15.48
## [18,]        5.52        9       12.48
## [19,]       10.52       14       17.48
## [20,]       16.52       20       23.48
```

1.4 Item Analysis

Item analysis within the classical approach often relies on two statistics for evaluating single items: the P-value and the point-biserial correlation coefficient. The P-value represents the proportion of examinees responding in the keyed direction, and is typically referred to as item difficulty.

The point-biserial correlation coefficient is an item's correlation with all of the other items and it provides an index of differentiating power of the item, which is typically referred to as item discrimination.

```r
item.analysis <-
  function(responses){
    # CRITICAL VALUES
    cvpb = 0.20
    cvdl = 0.15
    cvdu = 0.85

    require(CTT)
    (ctt.analysis <- CTT::reliability(responses, itemal = TRUE, NA.Delete = TRUE))

    # Mark items that are potentially problematic
    item.analysis <- data.frame(item = seq(1:ctt.analysis$nItem),
                                r.pbis = ctt.analysis$pBis,
                                bis = ctt.analysis$bis,
                                item.mean = ctt.analysis$itemMean,
                                alpha.del = ctt.analysis$alphaIfDeleted)

    # code provided by Dr. Gordon Brooks
    if (TRUE) {
      item.analysis$check <-
        ifelse(item.analysis$r.pbis < cvpb |
                 item.analysis$item.mean < cvdl |
                 item.analysis$item.mean > cvdu, "*", "")
    }
    return(item.analysis)
  }

dump("item.analysis", file = "item.analysis.R")

item.analysis(responses)
```

```
## Loading required package: CTT
##
## Attaching package: 'CTT'
##
## The following object is masked from 'package:polycor':
##
##     polyserial
```

```
##    item  r.pbis     bis item.mean alpha.del check
## 1     1 0.39437 0.56156   0.76337   0.81652
## 2     2 0.35252 0.46533   0.69651   0.81811
## 3     3 0.47979 0.59660   0.60988   0.81249
## 4     4 0.50022 0.60206   0.53895   0.81145
## 5     5 0.39173 0.47368   0.49535   0.81642
## 6     6 0.13943 0.63792   0.98721   0.82432     *
## 7     7 0.29056 0.36333   0.57674   0.82092
## 8     8 0.28749 0.35424   0.37500   0.82097
## 9     9 0.15304 0.19154   0.36105   0.82661     *
## 10   10 0.39760 0.50498   0.63256   0.81617
## 11   11 0.41120 0.51218   0.59826   0.81555
## 12   12 0.44791 0.54820   0.56802   0.81388
## 13   13 0.24341 0.30243   0.35291   0.82278
## 14   14 0.41945 0.54457   0.67035   0.81526
## 15   15 0.25782 0.31638   0.38547   0.82227
## 16   16 0.26183 0.32438   0.35930   0.82203
## 17   17 0.28491 0.34968   0.39012   0.82111
## 18   18 0.44171 0.53537   0.52616   0.81414
## 19   19 0.47581 0.62237   0.67558   0.81285
## 20   20 0.43468 0.53120   0.56163   0.81448
## 21   21 0.51696 0.61821   0.52442   0.81067
## 22   22 0.23286 0.56967   0.94477   0.82234      *
## 23   23 0.35983 0.45719   0.63547   0.81782
## 24   24 0.27183 0.61523   0.93198   0.82137      *
## 25   25 0.47706 0.60311   0.63430   0.81267
```

1.4.1 Item Difficulty

Item difficulty of an item is the proportion of students who answer that particular item correctly. So the item difficulty of a question is useful in assessing whether it is appropriate for the level of the students taking the test. Items with a difficulty measure of 0 and 1, or measures very close to these values, are not giving useful information about the differences among the students' abilities, as they do not discriminate between students with different trait levels. Items with a difficulty measure between 0.3 and 0.7 provide the maximum level of information about differences among student's abilities or trait levels unless test is designed for an extreme group and a cutting score has been determined in which case the item difficulty measure should relate to that particular cutting score (Allen 1979). Item Difficulty Item difficulty of an item is the proportion of students who answer that particular item correctly. So the item difficulty of a question is useful in assessing whether it is appropriate for the level of the students taking the test. Items with a difficulty measure of 0 and 1, or measures very close to these values, are not giving useful information about the differences among the students' abilities, as they do not discriminate between students with different trait levels.

```r
item.difficulty <-
  function(responses){
    # CRITICAL VALUES
    cvpb = 0.20
    cvdl = 0.15
    cvdu = 0.85

    require(CTT)
    ctt.analysis <- CTT::reliability(responses, itemal = TRUE, NA.Delete = TRUE)

    test_difficulty <- data.frame(item = 1:ctt.analysis$nItem ,
                                  difficulty = ctt.analysis$itemMean)

    plot(test_difficulty,
         main = "Test Item Difficulty",
         type = "p",
         pch = 1,
         cex = 2.8,
         col = "purple",
         ylab = "Item Mean (Difficulty)",
         xlab = "Item Number",
         ylim = c(0, 1),
         xlim = c(0, ctt.analysis$nItem))

    abline(h = cvdl, col = "tomato")
    abline(h = cvdu, col = "tomato")

    abline(h = .3, col = "dodgerblue")
    abline(h = .7, col = "dodgerblue")

    text(diff(range(test_difficulty[, 1]))/2, 0.7,
         "maximum information range",
         col = "dodger blue",
         pos = 3)

    text(diff(range(test_difficulty[, 1]))/2, cvdu,
         "rule of thumb acceptable range",
         col = "tomato",
         pos = 3)

    outlier <- data.matrix(subset(cbind(test_difficulty[, 1],
                                        test_difficulty[, 2]),
                                  subset = (test_difficulty[, 2] < cvdl |
                                              test_difficulty[, 2] > cvdu)))

    text(outlier, paste("i", outlier[,1],
                        sep = ""),
         col = "red",
         cex = .7)

    outlier2 <- data.matrix(subset(cbind(test_difficulty[, 1],
                                                    test_difficulty[, 2]),
                                            subset = ((test_difficulty[, 2] > cvdl &
                                                           test_difficulty[, 2] < .3) |
                                                          (test_difficulty[, 2] < cvdu &
                                                             test_difficulty[, 2] > .7))))

    text(outlier2, paste("i", outlier2[,1], sep = ""),
         col = "dodgerblue",
         cex = .7)

    return(test_difficulty[order(test_difficulty$difficulty),])
  }

dump("item.difficulty", file = "item.difficulty.R")

item.difficulty(responses)
```

```
##    item difficulty
## 13   13    0.35291
## 16   16    0.35930
## 9     9    0.36105
## 8     8    0.37500
## 15   15    0.38547
## 17   17    0.39012
## 5     5    0.49535
## 21   21    0.52442
## 18   18    0.52616
## 4     4    0.53895
## 20   20    0.56163
## 12   12    0.56802
## 7     7    0.57674
## 11   11    0.59826
## 3     3    0.60988
## 10   10    0.63256
## 25   25    0.63430
## 23   23    0.63547
```

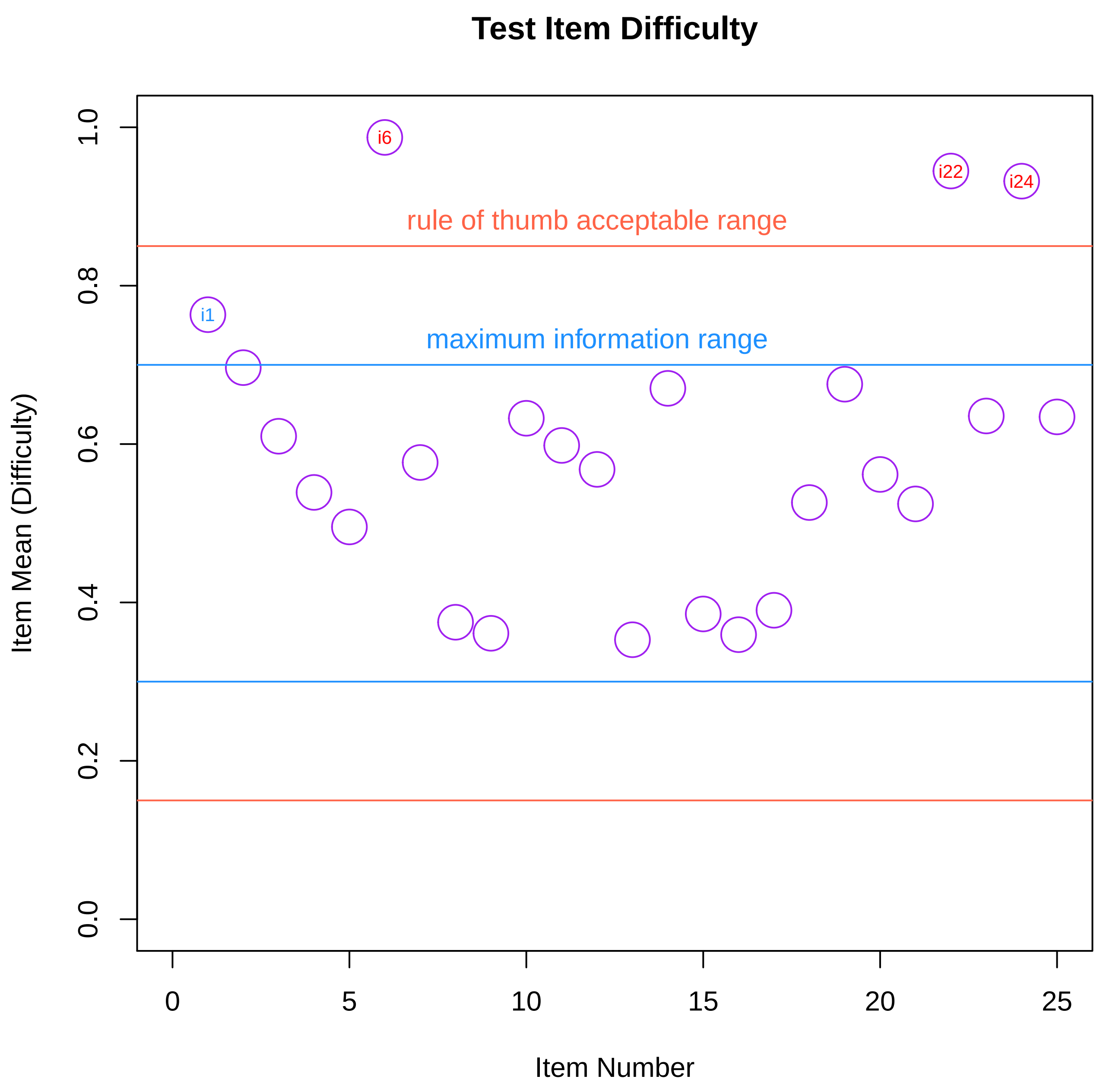
Test Item Difficulty
Item Mean (Difficulty)
Item Number
rule of thumb acceptable range
maximum information range
i1
i6
i22
i24
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25

Figure 1:

```
## 14   14    0.67035
## 19   19    0.67558
## 2     2    0.69651
## 1     1    0.76337
## 24   24    0.93198
## 22   22    0.94477
## 6     6    0.98721
```

1.4.2 Item Discrimination

The item-discrimination index (Tristan 1998), d_i of an item, i can be calculated as

$$d_i = \frac{U_i}{n_i U} - \frac{L_i}{n_i L}$$

where

U_i: The number of student who answered item, i correctly and have a total test score in the upper range, between the upper 10% and 33% but optimally the upper 27%.

L_i: The number of student who answered item, i correctly and have a total test score in the lower range, between the lower 10% and 33% but optimally the lower 27%

n_{iU}: The number of students in U_i

n_{iL}: The number of students in L_i.

```
item.discrimination <-
  function(responses){
    # CRITICAL VALUES
    cvpb = 0.20
    cvdl = 0.15
    cvdu = 0.85

    require(CTT)
    ctt.analysis <- CTT::reliability(responses, itemal = TRUE, NA.Delete = TRUE)

    item.discrimination <- data.frame(item = 1:ctt.analysis$nItem ,
                                      discrimination = ctt.analysis$pBis)
    plot(item.discrimination,
         type = "p",
         pch = 1,
         cex = 3,
         col = "purple",
         ylab = "Item-Total Correlation",
         xlab = "Item Number",
         ylim = c(0, 1),
         main = "Test Item Discriminations")

    abline(h = cvpb, col = "red")

    outlier <- data.matrix(subset(item.discrimination,
                                  subset = (item.discrimination[, 2] < cvpb)))

    text(outlier, paste("i", outlier[,1], sep = ""), col = "red", cex = .7)

    return(item.discrimination[order(item.discrimination$discrimination),])
  }

dump("item.discrimination", file = "item.discrimination.R")

item.discrimination(responses)
```

```
##    item discrimination
## 6     6        0.13943
## 9     9        0.15304
## 22   22        0.23286
## 13   13        0.24341
## 15   15        0.25782
## 16   16        0.26183
## 24   24        0.27183
## 17   17        0.28491
## 8     8        0.28749
## 7     7        0.29056
## 2     2        0.35252
```

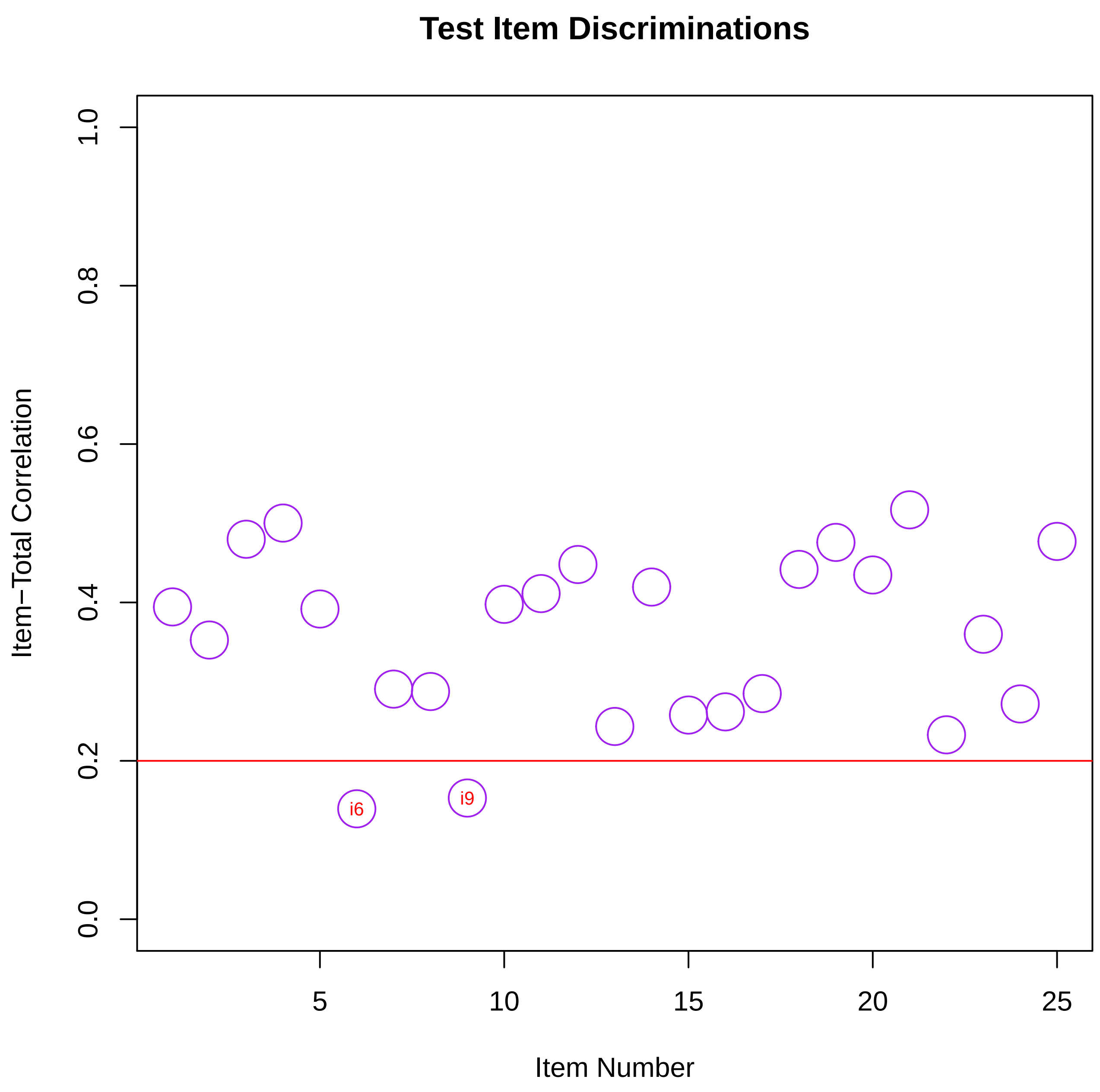
Test Item Discriminations
Item−Total Correlation
Item Number
0.0
0.2
0.4
0.6
0.8
1.0
5
10
15
20
25
i6
i9

Figure 2:

```
## 23   23          0.35983
## 5     5          0.39173
## 1     1          0.39437
## 10   10          0.39760
## 11   11          0.41120
## 14   14          0.41945
## 20   20          0.43468
## 18   18          0.44171
## 12   12          0.44791
## 19   19          0.47581
## 25   25          0.47706
## 3     3          0.47979
## 4     4          0.50022
## 21   21          0.51696
```

1.4.3 Item-Total Correlation

A measure of item discrimination that is alternative to d_i is the item-total test-score point-biserial correlation, r_{iX} between scores on item i and total test scores, X can be calculated as:

$$r_{iX} = \frac{\bar{X}_i - \bar{X}}{S_X}\sqrt{\frac{p_i}{1-p_i}}$$

where

X_i : The mean of the test scores for students who answered the item, i correctly.

p_i : The item difficulty.

$\bar{X}$: The mean of X.

S_X: The standard deviation of X.

The higher the correlation between an item and the test total score for a student, the higher the r_{iX} or d_i measures. As a rule of thumb, an item with a discrimination index that is smaller than 0.2 is not acceptable (Allen 1979). Both r_{iX} and d_i should be positive on any reasonable item, meaning more high scoring students than low scoring students should answer the item correctly. When these measures are negative the item is measuring the opposite of what the rest of the test is measuring. In that case, either an error is made in scoring that particular item or it is poorly worded. For example, the scale for the negatively worded items should be reversed. Items with negative or low r_{iX} or d_i valued should be revised or eliminated.

When performance on an item is uncorrelated with performance on all the other items in the test, the item point biserial correlation still will be positive if that particular item's score is included in the total test score. Therefore, calculating item point biserial correlation for an item without including that particular item's score in the total score will provide a more accurate item discrimination measure. Items with a negative item discrimination index are counterproductive regarding the purpose of testing.

```
test_item.total <-
  function(responses){
    # CRITICAL VALUES
    cvpb = 0.20
    cvdl = 0.15
    cvdu = 0.85

    require(CTT)
    ctt.analysis <- CTT::reliability(responses, itemal = TRUE, NA.Delete = TRUE)

    test_item.total <- data.frame(item = 1:ctt.analysis$nItem ,
                                  biserial = ctt.analysis$bis)

    plot(test_item.total,
         main = "Test Item-Total Correlation",
         type = "p",
         pch = 1,
         cex = 2.8,
         col = "purple",
         ylab = "Item-Total Correlation",
         xlab = "Item Number",
         ylim = c(0, 1),
         xlim = c(0, ctt.analysis$nItem))

    abline(h = cvpb, col = "red")

    outlier <- data.matrix(subset(test_item.total,
                                  subset = test_item.total[,2] < cvpb))

    text(outlier, paste("i", outlier[,1], sep = ""), col = "red", cex = .7)
    return(test_item.total[order(test_item.total$biserial),])
  }

dump("test_item.total", file = "test_item.total.R")

test_item.total(responses)
```

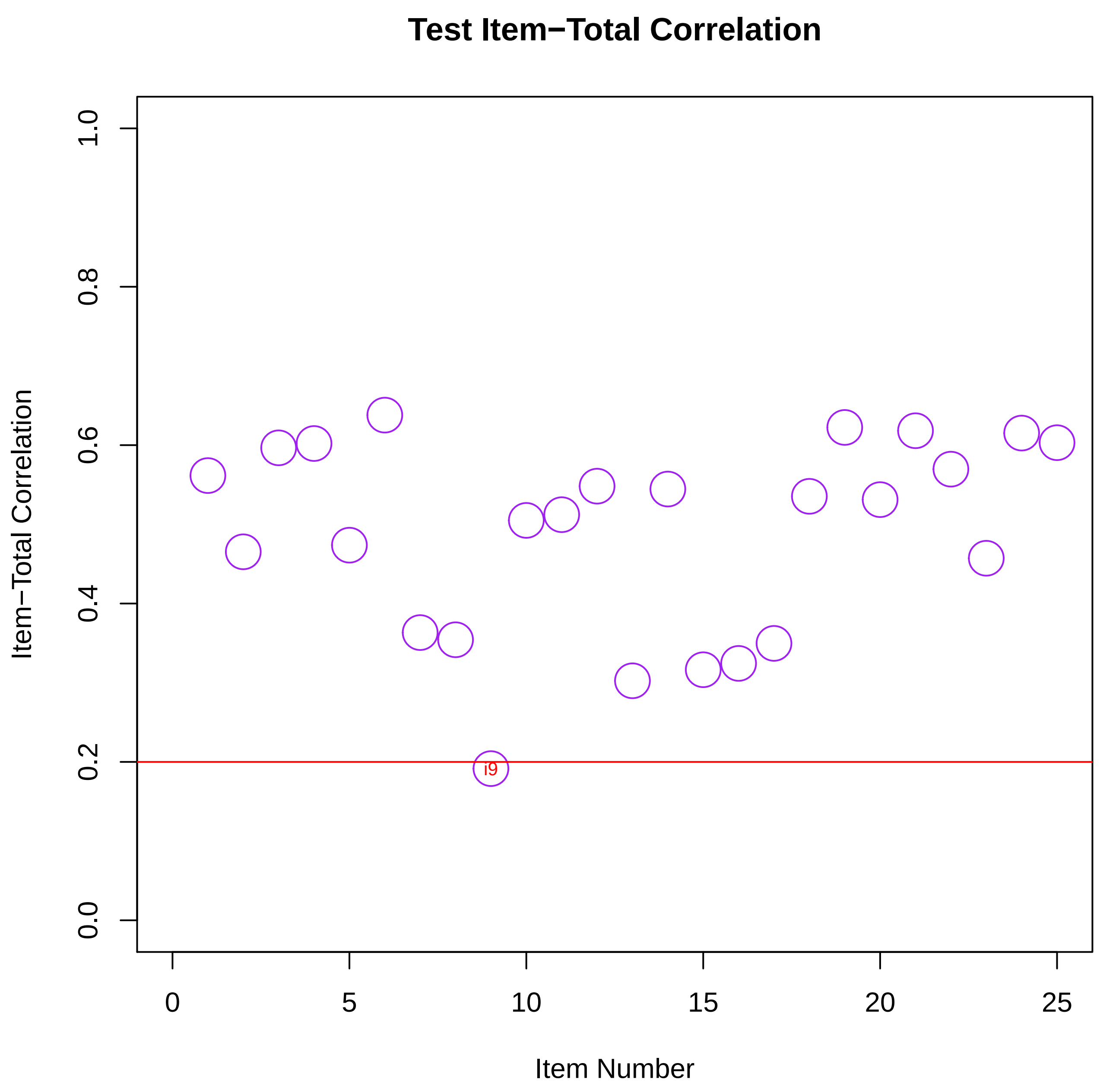

Figure 3:

```
##    item biserial
## 9     9  0.19154
## 13   13  0.30243
## 15   15  0.31638
## 16   16  0.32438
## 17   17  0.34968
## 8     8  0.35424
## 7     7  0.36333
## 23   23  0.45719
## 2     2  0.46533
## 5     5  0.47368
## 10   10  0.50498
## 11   11  0.51218
## 20   20  0.53120
## 18   18  0.53537
## 14   14  0.54457
## 12   12  0.54820
## 1     1  0.56156
## 22   22  0.56967
## 3     3  0.59660
## 4     4  0.60206
## 25   25  0.60311
## 24   24  0.61523
## 21   21  0.61821
## 19   19  0.62237
## 6     6  0.63792
```

1.4.4 Distractor/option analysis

In distractor analysis examinees are divided into three ability levels lower, middle and upper third of the sample based on their total test score. The proportions of examinees who mark each option in each of the three ability levels are compared. In the lower ability level, we would expect to see a smaller proportion of examinees choosing the correct option and a larger proportion of them marking the incorrect options or distractors. Ideally, good distractors would attract about the same proportion of examinees. Distractors that don't attract any or attract very small proportion of examinees relative to other distractors should be considered for revision. In the higher ability level, we would expect to see that the majority of examinees choose the correct option. If a distractor is more appealing than the correct option to the higher ability level examinees, then it should be eliminated or revised.

SIMULATE TEST DATA

```
# import data
mytest <- read.csv("EDRE7110_FINAL_RAW.CSV",
                   header = TRUE,
                   na.strings = " ",
                   stringsAsFactors = FALSE)

#  answer key
CTTkey <- mytest[1,-1] # Drops the first column, names variable

# data
CTTdata <- mytest[-1,-1] # Drops the first column, names variable
```

Distractor/option analysis

```
distractor.analysis(CTTdata, CTTkey, p.table = TRUE, write.csv = "distractor_analysis")
```

```
## $Item01
##         score.level
## response lower middle upper
##        1 0.000  0.286 0.200
##       *2 0.875  0.429 0.800
##        3 0.000  0.143 0.000
##        4 0.125  0.143 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item02
##         score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##        2 0.250  0.143 0.000
##       *3 0.500  0.571 1.000
##        4 0.250  0.286 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item03
##         score.level
## response lower middle upper
##        1 0.250  0.286 0.200
##        2 0.125  0.000 0.000
##        3 0.125  0.000 0.000
##       *4 0.500  0.714 0.800
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item04
##         score.level
## response lower middle upper
##        1 0.000  0.143 0.000
##        2 0.500  0.286 0.000
##       *3 0.250  0.000 0.600
##        4 0.250  0.571 0.400
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item05
##         score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##        2 0.000  0.000 0.000
##        3 0.125  0.000 0.000
##       *4 0.875  1.000 1.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item06
##         score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##       *2 0.875  1.000 1.000
##        3 0.000  0.000 0.000
##        4 0.125  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item07
##         score.level
## response lower middle upper
##       *1 0.375  0.714 0.800
##        2 0.500  0.286 0.200
##        3 0.125  0.000 0.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item08
##         score.level
## response lower middle upper
##        1  0.00   0.00  0.00
##       *2  0.50   1.00  1.00
##        3  0.25   0.00  0.00
##        4  0.25   0.00  0.00
##        5  0.00   0.00  0.00
##        9  0.00   0.00  0.00
##
## $Item09
##         score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##        2 0.000  0.143 0.000
##       *3 0.750  0.857 1.000
##        4 0.250  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
```

```
##
## $Item10
##         score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##        2 0.000  0.000 0.000
##        3 0.500  0.286 0.200
##        4 0.125  0.143 0.000
##       *5 0.375  0.571 0.800
##        9 0.000  0.000 0.000
##
## $Item11
##         score.level
## response lower middle upper
##        1 0.000  0.143 0.000
##        2 0.000  0.000 0.000
##       *3 0.750  0.857 1.000
##        4 0.250  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item12
##         score.level
## response lower middle upper
##       *1 0.625  1.000 1.000
##        2 0.250  0.000 0.000
##        3 0.125  0.000 0.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item13
##         score.level
## response lower middle upper
##        1 0.750  0.571 0.400
##       *2 0.125  0.429 0.400
##        3 0.125  0.000 0.200
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item14
##         score.level
## response lower middle upper
##        1  0.25   0.00  0.00
##       *2  0.75   1.00  1.00
##        3  0.00   0.00  0.00
##        4  0.00   0.00  0.00
##        5  0.00   0.00  0.00
##        9  0.00   0.00  0.00
##
## $Item15
##         score.level
## response lower middle upper
##        1 0.000  0.143 0.000
##        2 0.500  0.000 0.200
##       *3 0.250  0.571 0.800
##        4 0.250  0.286 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item16
##         score.level
## response lower middle upper
##        1 0.125  0.000 0.000
##       *2 0.875  1.000 1.000
##        3 0.000  0.000 0.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item17
##          score.level
## response lower middle upper
##        1 0.000  0.143 0.000
##        2 0.125  0.000 0.000
##       *3 0.750  0.857 1.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.125  0.000 0.000
##
## $Item18
##          score.level
## response lower middle upper
##        1 0.125  0.000 0.000
##       *2 0.250  1.000 1.000
##        3 0.500  0.000 0.000
##        4 0.125  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item19
##          score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##       *2 0.625  1.000 1.000
##        3 0.000  0.000 0.000
##        4 0.375  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item20
##          score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##        2 0.375  0.143 0.200
##       *3 0.500  0.714 0.800
##        4 0.125  0.143 0.000
##         5 0.000  0.000 0.000
##         9 0.000  0.000 0.000
##
## $Item21
##          score.level
## response lower middle upper
##         1 0.000  0.000 0.000
##         2 0.125  0.000 0.000
##        *3 0.875  1.000 1.000
##         4 0.000  0.000 0.000
##         5 0.000  0.000 0.000
##         9 0.000  0.000 0.000
##
## $Item22
##          score.level
## response lower middle upper
##         1 0.000  0.143 0.000
##         2 0.250  0.143 0.000
##         3 0.125  0.000 0.000
##        *4 0.625  0.714 1.000
##         5 0.000  0.000 0.000
##         9 0.000  0.000 0.000
##
## $Item23
##          score.level
## response lower middle upper
##        *1 0.875  1.000 1.000
##         2 0.000  0.000 0.000
##         3 0.000  0.000 0.000
##         4 0.125  0.000 0.000
##         5 0.000  0.000 0.000
##         9 0.000  0.000 0.000
##
## $Item24
##          score.level
## response lower middle upper
##        1 0.125  0.143 0.000
##        2 0.250  0.000 0.000
##       *3 0.625  0.857 1.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item25
##         score.level
## response lower middle upper
##        1 0.250  0.000 0.200
##        2 0.125  0.000 0.000
##       *3 0.500  0.857 0.800
##        4 0.125  0.143 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item26
##         score.level
## response lower middle upper
##        1 0.125  0.000 0.000
##       *2 0.250  0.714 1.000
##        3 0.000  0.143 0.000
##        4 0.625  0.143 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item27
##         score.level
## response lower middle upper
##       *1 0.625  1.000 1.000
##        2 0.250  0.000 0.000
##        3 0.125  0.000 0.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
```

```
##
## $Item28
##         score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##       *2 0.750  1.000 1.000
##        3 0.125  0.000 0.000
##        4 0.125  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item29
##         score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##        2 0.000  0.143 0.000
##       *3 1.000  0.857 1.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item30
##         score.level
## response lower middle upper
##        1 0.625  0.429 0.000
##       *2 0.375  0.571 1.000
##        3 0.000  0.000 0.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item31
##         score.level
## response lower middle upper
##       *1 0.375  0.714 1.000
##        2 0.000  0.000 0.000
##        3 0.125  0.143 0.000
##        4 0.500  0.143 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item32
##         score.level
## response lower middle upper
##       *1 0.500  0.571 1.000
##        2 0.125  0.143 0.000
##        3 0.125  0.000 0.000
##        4 0.250  0.286 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item33
##         score.level
## response lower middle upper
##        1 0.125  0.143 0.000
##       *2 0.500  0.571 0.800
##        3 0.000  0.286 0.200
##        4 0.375  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item34
##         score.level
## response lower middle upper
##        1 0.125  0.000 0.000
##        2 0.125  0.000 0.000
##       *3 0.250  0.714 1.000
##        4 0.500  0.286 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item35
##          score.level
## response lower middle upper
##        1 1.000  0.143 0.000
##       *2 0.000  0.857 1.000
##        3 0.000  0.000 0.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item36
##          score.level
## response lower middle upper
##       *1 0.875  0.857 1.000
##        2 0.125  0.143 0.000
##        3 0.000  0.000 0.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item37
##          score.level
## response lower middle upper
##        1 0.000  0.000 0.000
##        2 0.000  0.000 0.000
##       *3 0.875  1.000 1.000
##        4 0.125  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item38
##          score.level
## response lower middle upper
##        1     0      0     0
##       *2     1      1     1
##        3     0      0     0
##        4     0      0     0
##        5     0      0     0
##        9     0      0     0
##
## $Item39
##         score.level
## response lower middle upper
##        1 0.000  0.143 0.200
##       *2 1.000  0.857 0.800
##        3 0.000  0.000 0.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item40
##         score.level
## response lower middle upper
##        1 0.375  0.143 0.400
##        2 0.000  0.286 0.200
##       *3 0.125  0.429 0.400
##        4 0.500  0.143 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item41
##         score.level
## response lower middle upper
##        1 0.125  0.000 0.000
##       *2 0.875  1.000 1.000
##        3 0.000  0.000 0.000
##        4 0.000  0.000 0.000
##        5 0.000  0.000 0.000
##        9 0.000  0.000 0.000
##
## $Item42
##         score.level
## response lower middle upper
##         1     0      0     0
##         2     0      0     0
##        *3     1      1     1
##         4     0      0     0
##         5     0      0     0
##         9     0      0     0
##
## $Item43
##          score.level
## response lower middle upper
##         1 0.000  0.000 0.000
##         2 0.125  0.000 0.000
##         3 0.125  0.143 0.000
##        *4 0.750  0.857 1.000
##         5 0.000  0.000 0.000
##         9 0.000  0.000 0.000
##
## $Item44
##          score.level
## response lower middle upper
##         1 0.250  0.429 0.200
##         2 0.125  0.000 0.200
##        *3 0.625  0.571 0.600
##         4 0.000  0.000 0.000
##         5 0.000  0.000 0.000
##         9 0.000  0.000 0.000
```

Overall, this test doesn't have good distractors. Students with higher ability either chose the correct option one-hundred percent of the time, or they were slightly distracted by a second option only. There are three questions (items 13, 40 and 44), which had two or more distractors that were plausible to the upper level students. Items 4, 13 and 40 had stronger distractors at higher levels such that the proportion of students marking the correct option among the high level students were smaller than the proportion of students marking the correct option among the middle or lower level students. Of the few questions that had only a single distractor, the upper level students were distracted the most by item 4. Item 10 had two distractors for each of the middle and lower level students. The remaining items had some distractors that were not functional at every ability level.

1.5 Validity of Tests

Constructs are abstractions that conceptualize a latent variable that is responsible for scores on a given measure. **Construct validity** is whether a measure is actually measuring or representing the construct that it intends to measure. Multitrait-Multimethod Matrix, Convergent and Discriminant Analysis, and Factor Analysis are the most commonly used methods to establish construct validity. **Content or logical validity** is the extent to which a measure represents all facets of a given construct. According to Alen and Yen there are two types of content validity: Face validity and logical validity. If a test looks like it is going to measure what it purports to be measuring, then it has **face validity**. **Criterion validity** is the extent to which a variable or set of variables are able to predict a future behavioral criterion. It is measured by a correlation coefficient or correlation between the test score and criterion score.

1.5.1 Multitrait-Multimethod Matrix

Multitrait-Multimethod Matrix is a correlation matrix between assessment traits (or dimensions, or components, or constructs) and different methods of measurement that are designed to study the discriminant convergence of trait indicators in validation studies. Tests designed to measure the same construct should correlate highly amongst themselves and the construct being measured by a test should not correlate highly with different constructs. The main diagonal in such a matrix contains the reliabilities of instruments or monotrait–monomethod correlations. Adjacent to the main diagonal are triangles with heterotrait–monomethod correlations. Also, there are blocks with correlations involving two different methods diagonals of which are correlations for a single trait. These monotrait–heteromethod correlations are validity indices.

According to Campbell and Fiske (1959), convergence of the trait indicators and the discriminability of traits provide evidence for validity of the measurement instrument. We have discriminant validity or convergence:

if the correlations between measurements of the same trait with different methods are large;

if the correlations between measurements of a trait with different methods are higher than the correlations of different traits measured with the same method (i.e. the validity diagonals should be higher than the correlations in the monomethod–heterotrait triangles);

if a validity coefficient r_{ii} of two different methods for the same trait is larger than the correlations r_{ij} and r_{ji} of two different methods for two different traits; and

if in the heterotrait triangles of the monomethod blocks and the heteromethod blocks, the pattern of correlations is the same.

1.5.2 Unidimensionality

Individual items on a test must be unidimensional because test items that lack unidimensionality are impossible to interpret accurately. As a result, binary test items are reliable proxies for a construct only when they are unidimensional. The closer to a test to pure state that is when all variables have high factor loadings on a given factor and very low factor loadings on all other factors, the more evidence is inferred for construct validity and unidimensionality.

According to the **Kaiser's Rule** all components with eigenvalues under 1.0 is dropped. It is not recommended when used as the sole cut-off criterion for estimating the number of factors because it tends to overextract factors. Henry Kaiser (1970) introduced a Measure of Sampling Adequacy (MSA) of factor analytic data matrices. The index is known as the Kaiser-Meyer-Olkin (KMO) index.

Kaiser's description of MSAs:

above .9 as marvelous,

above .8 as meritorious,

above .7 as middling,

above .6 as mediocre,

above .5 as miserable, and

below .5 as unacceptable.

Horn-Glorfeld Parallel Analysis (Glorfeld 1995; Horn 1965) compares the observed eigenvalues with those obtained from a Monte-Carlo simulation. Eigenvalues are the variances of the principal components. A factor or component is retained if the associated eigenvalue is bigger than the 95th of the distribution of eigenvalues derived from the random data. The first component will always account for the most variance, thus have the highest eigenvalue. The next component will account for as much of the left over variance as it can, so each successive component will account for less variance than the previous component.

Scree Plot graphs the components on the X axis and the corresponding eigenvalues on the Y-axis. Moving towards right on the curve, to a greater number of components the eigenvalues drop sharply, then gradualy forming an elbow. According to **Cattell's Scree Test** (Cattell 1966) only components above the elbowsare kept.

```r
# import data
ctt.data <- read.csv("test2.csv", header = TRUE, na.strings = " ")
str(ctt.data)
```

```
## 'data.frame':    1720 obs. of  26 variables:
##  $ Gender: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V1    : int  1 1 1 0 1 0 0 1 0 1 ...
##  $ V2    : int  1 1 1 1 1 1 0 1 1 1 ...
##  $ V3    : int  1 1 0 0 1 1 1 1 1 1 ...
##  $ V4    : int  1 1 0 1 1 0 1 1 0 1 ...
##  $ V5    : int  0 0 1 1 1 0 0 1 0 1 ...
##  $ V6    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V7    : int  1 1 1 0 0 0 1 1 1 1 ...
##  $ V8    : int  0 1 0 0 1 0 0 0 0 0 ...
##  $ V9    : int  0 1 0 0 1 0 1 1 0 0 ...
##  $ V10   : int  1 1 1 0 1 0 0 1 1 1 ...
##  $ V11   : int  1 1 1 0 1 1 1 1 1 0 ...
##  $ V12   : int  0 1 0 0 1 1 1 1 0 1 ...
##  $ V13   : int  0 0 1 0 1 1 0 1 0 1 ...
##  $ V14   : int  0 1 1 0 1 0 1 1 0 1 ...
##  $ V15   : int  1 1 0 0 1 0 1 1 1 0 ...
##  $ V16   : int  1 0 0 1 1 0 0 0 0 0 ...
##  $ V17   : int  1 0 0 0 1 0 0 1 1 0 ...
##  $ V18   : int  1 1 0 0 1 0 0 1 0 0 ...
##  $ V19   : int  1 1 1 0 1 1 0 1 1 1 ...
##  $ V20   : int  1 1 0 0 1 0 1 1 0 1 ...
##  $ V21   : int  1 1 1 0 1 0 0 1 0 0 ...
##  $ V22   : int  1 1 1 1 1 1 1 1 0 1 ...
##  $ V23   : int  1 0 0 1 1 0 0 0 0 1 ...
##  $ V24   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V25   : int  1 1 0 0 1 0 0 1 1 1 ...
```

```r
items <- ctt.data[-1,]

extract <- 1
minload <- .3
nsize <- nrow(items)
numitems <- ncol(items)

require(paran)
```

```
## Loading required package: paran
```

```
# Horn’s Parallel Analysis of Principal Components/Factors
# In principal components analysis, we aim to explain the
# total variance of the variables, whereas in factor analysis,
# we focus on the common variance of the variables (the amount
# of variance the variables share, which is always less than the total variance).
pca <- paran(items,
             centile = 95,
             cfa = FALSE,
             seed = 1804,
             iterations = 50,
             graph = TRUE,
             color = TRUE,
             all = TRUE)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 50 iterations, using the 95 centile estimate
##
## --------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## --------------------------------------------------
## 1           4.754325    5.019439      0.265114
## 2           1.096809    1.319736      0.222926
## 3           1.002174    1.195616      0.193441
## 4           0.896347    1.063022      0.166675
## 5           0.858584    1.003085      0.144501
## 6           0.846299    0.968674      0.122375
## 7           0.847581    0.955900      0.108319
## 8           0.843857    0.932251      0.088394
## 9           0.830062    0.906281      0.076219
## 10          0.829225    0.889351      0.060126
## 11          0.829579    0.874409      0.044829
## 12          0.819541    0.851430      0.031888
## 13          0.832013    0.847271      0.015258
## 14          0.810888    0.813621      0.002733
## 15          0.808565    0.791470     -0.01709
## 16          0.805515    0.776549     -0.02896
## 17          0.800165    0.759064     -0.04110
## 18          0.798922    0.743350     -0.05557
## 19          0.784267    0.714016     -0.07025
## 20          0.786668    0.703445     -0.08322
## 21          0.785304    0.689318     -0.09598
## 22          0.797706    0.681443     -0.11626
## 23          0.779976    0.651025     -0.12895
## 24          0.784114    0.634823     -0.14929
## 25          0.786410    0.618915     -0.16749
## 26          0.790853    0.596481     -0.19437
## --------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (3 components retained)
```

```r
paf <- paran(items,
             centile = 95,
             cfa = TRUE,
             seed = 1804,
             iterations = 50,
             graph = TRUE,
             color = TRUE,
             all = TRUE)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
```

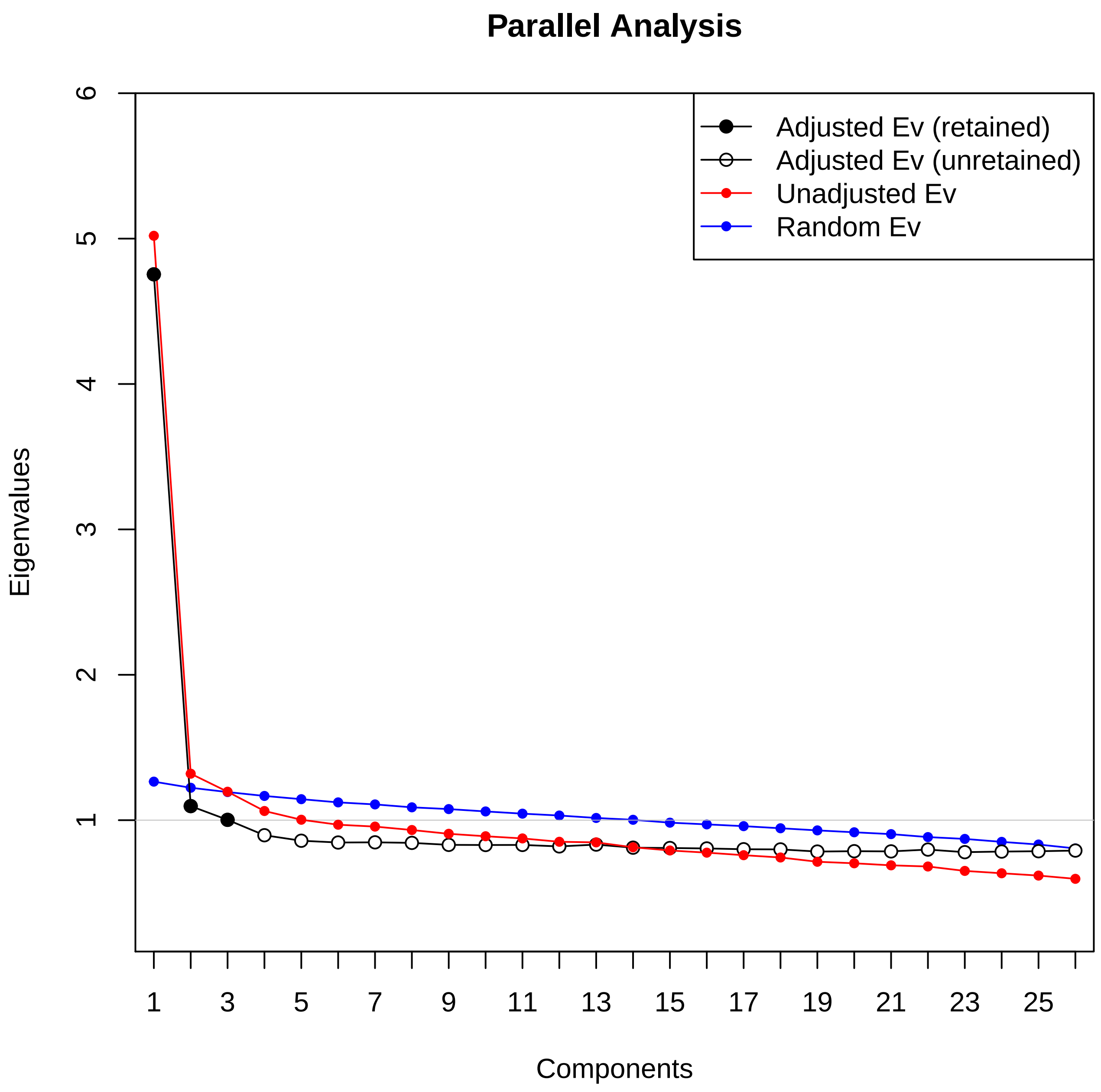
Parallel Analysis
Adjusted Ev (retained)
Adjusted Ev (unretained)
Unadjusted Ev
Random Ev
Eigenvalues
Components
1
2
3
4
5
6
1
3
5
7
9
11
13
15
17
19
21
23
25

Figure 4:

```
## Results of Horn's Parallel Analysis for factor retention
## 50 iterations, using the 95 centile estimate
##
## --------------------------------------------------
## Factor      Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## --------------------------------------------------
## No components passed.
## --------------------------------------------------
## 1            3.935116    4.218301      0.283184
## 2            0.193276    0.437077      0.243801
## 3            0.108745    0.319477      0.210731
## 4           -0.003764    0.180516      0.184281
## 5           -0.031233    0.128528      0.159761
## 6           -0.047550    0.091235      0.138785
## 7           -0.042554    0.080268      0.122822
## 8           -0.044331    0.059082      0.103413
## 9           -0.033697    0.056585      0.090283
## 10          -0.039086    0.036281      0.075368
## 11          -0.040027    0.018854      0.058882
## 12          -0.037332    0.007864      0.045197
## 13          -0.035260   -0.00723      0.028030
## 14          -0.039761   -0.02355      0.016205
## 15          -0.034738   -0.03826     -0.00352
## 16          -0.036002   -0.05129     -0.01528
## 17          -0.034055   -0.06151     -0.02745
## 18          -0.021760   -0.06509     -0.04333
## 19          -0.032407   -0.08962     -0.05721
## 20          -0.034710   -0.10509     -0.07038
## 21          -0.040592   -0.12366     -0.08307
## 22          -0.034309   -0.13682     -0.10252
## 23          -0.036316   -0.15162     -0.11530
## 24          -0.035522   -0.17076     -0.13524
## 25          -0.051703   -0.20368     -0.15198
## 26          -0.051806   -0.23101     -0.17920
## --------------------------------------------------
```

```
##
## Adjusted eigenvalues > 0 indicate dimensions to retain.
## (3 factors    retained)
```

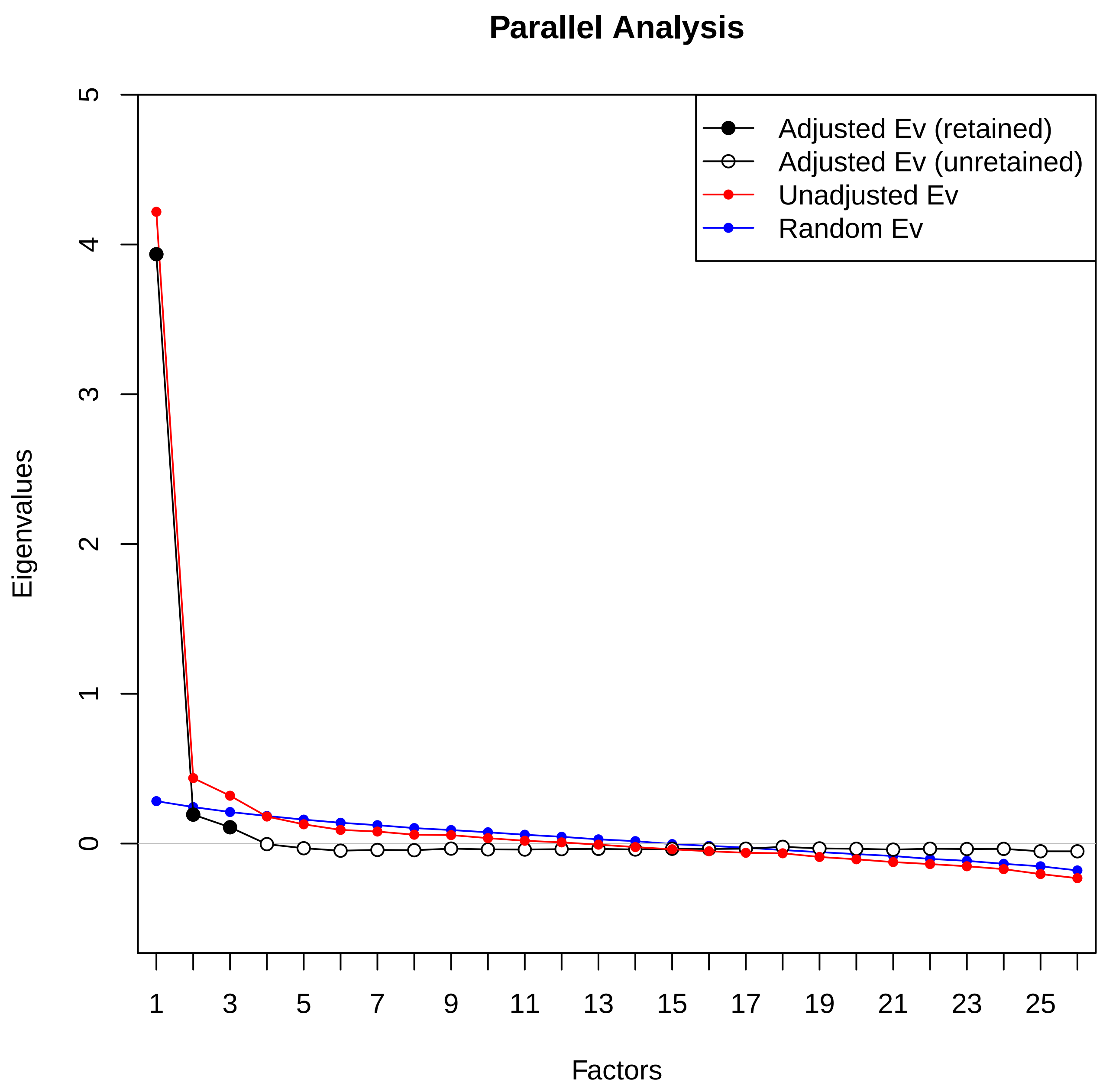

Figure 5:

```r
require(GPArotation)
```

```
## Loading required package: GPArotation
```

```r
require(psych)
```

```
## Loading required package: psych
##
## Attaching package: 'psych'
##
## The following object is masked from 'package:CTT':
##
##     polyserial
##
## The following object is masked from 'package:eRm':
##
##     sim.rasch
##
## The following object is masked from 'package:irtoys':
##
##     sim
##
## The following object is masked from 'package:ltm':
##
##     factor.scores
##
## The following object is masked from 'package:polycor':
##
##     polyserial
```

```r
cormat <- cor(items)

# Kaiser-Myer-Olkin (KMO) (should be > ~.5)
KMO(cormat)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = cormat)
## Overall MSA =  0.92
## MSA for each item =
## Gender     V1     V2     V3     V4     V5     V6     V7     V8     V9
##   0.46   0.93   0.94   0.94   0.95   0.93   0.74   0.94   0.91   0.88
##    V10    V11    V12    V13    V14    V15    V16    V17    V18    V19
##   0.94   0.89   0.94   0.89   0.94   0.91   0.94   0.91   0.94   0.94
##    V20    V21    V22    V23    V24    V25
##   0.94   0.94   0.82   0.94   0.87   0.94
```

```
fit <- principal(r = cormat,
                 nfactors = extract,
                 rotate = "varimax",
                 n.obs = nsize,
                 residuals = FALSE,
                 scores = FALSE)

print(loadings(fit),
      digits = 2,
      cutoff = minload,
      sort = TRUE)
```

```
##
## Loadings:
##  [1]  0.58  0.59  0.54  0.51  0.52  0.57  0.52  0.61  0.57        0.48
## [12]  0.43  0.47        0.36  0.34        0.48  0.50        0.31  0.32
## [23]  0.34  0.30  0.44  0.35
##
##                  PC1
## SS loadings     5.02
## Proportion Var 0.19
```

```
summary(fit)
```

```
##
## Factor analysis with Call: principal(r = cormat, nfactors = extract, residuals = FALSE,
##     rotate = "varimax", n.obs = nsize, scores = FALSE)
##
## Test of the hypothesis that 1 factor is sufficient.
## The degrees of freedom for the model is 299  and the objective function was  0.39
## The number of observations was  1719  with Chi Square =  665.48  with prob <  6.1e-30
##
## The root mean square of the residuals (RMSA) is  0.04
```

```r
fit <- fa(r = cormat,
          covar = FALSE,
          nfactors = extract,
          rotate = "varimax",
          fm = "pa",
          n.obs = nsize,
          SMC = TRUE,
          residuals = FALSE)

print(loadings(fit),
      digits = 2,
      cutoff = minload,
      sort = TRUE)
```

```
##
## Loadings:
##  [1] 0.54 0.55 0.53 0.57 0.53      0.44 0.39 0.43      0.32
## [12] 0.31      0.44 0.46 0.50      0.47           0.31 0.48
## [23] 0.48      0.40 0.31
##
##                 PA1
## SS loadings    4.24
## Proportion Var 0.16
```

```r
summary(fit)
```

```
##
## Factor analysis with Call: fa(r = cormat, nfactors = extract, n.obs = nsize, rotate = "v
##     residuals = FALSE, SMC = TRUE, covar = FALSE, fm = "pa")
##
## Test of the hypothesis that 1 factor is sufficient.
## The degrees of freedom for the model is 299  and the objective function was  0.36
## The number of observations was  1719  with Chi Square =  607.44  with prob <  3.5e-23
##
## The root mean square of the residuals (RMSA) is  0.03
## The df corrected root mean square of the residuals is  0.03
##
## Tucker Lewis Index of factoring reliability =  0.936
## RMSEA index =  0.025  and the 90 % confidence intervals are  0.022 0.027
## BIC =  -1620
```

References

Allen, & Yen, M. J. 1979. *Introduction to Measurement Theory.* Monterey, CA: Brooks/Cole.

Brown, W. 1910. “Some Experimental Results in the Correlation of Mental Abilities.” *British Journal of Psychology* 3 (271-295).

Campbell, D. T., and D. W. Fiske. 1959. “Convergent and Discriminant Validationbythemultitrait-Multimethodmatrix.” *PsychologicalBulletin* 56: 81–105.

Cattell, R. B. 1966. “The Scree Test for the Number of Factors.” *Multivariate Behavioral Research* 1: 245–76.

Cronbach, L. J. 1951. “Coefficient Alpha and the Internal Structure of Tests.” *Psychometrika* 16 (3): 297–334.

Fan, X. 1998. “Item Response Theory and Classical Test Theory: An Empirical Comparison of Their Item/person Statistics.” *Educational and Psychological Measurement* 58: 357–81.

Glorfeld, L. W. 1995. “An Improvement on Horn’s Parallel Analysis Methodology for Selecting the Correct Number of Factors to Retain.” *Educational and Psychological Measurement* 55 (3): 377–93.

Guttman, L. 1945. “A Basis for Analyzing Test-Retest Reliability.” *Psychometrika* 10 (4): 255–82.

Horn, J. L. 1965. “A Rationale and a Test for the Number of Factors in Factor Analysis.” *Psychometrika* 30: 179–85.

Kaiser, H. F. 1970. “A Second Generation Little Jiffy.” *Psychometrika* 35 (4): 401–15.

Kuder, G. F., and M. W. Richardson. 1937. “The Theory of the Estimation of Test Reliability.” *Psychometrika* 2 (3): 151–60.

Spearman, C. 1910. “Correlation Calculated with Faulty Data.” *British Journal of Psychology* 3 (271-295).

Tristan, L. A. 1998. “The Item Discrimination Index: Does It Work?” *Rasch Measurement Transactions* 12 (1): 626.