Psychometrics

Part 2: Guessing/Carelessness, Test Equating, Scaling, and Linking, Differential Item Functioning and The Rasch Model

SALVADOR CASTRO

Contents

1 Item Response Theory (Part 2)

1.1 Comparison of IRT Models

1.1.1 Comparison of Ability Estimates

Simulate data for a 1-PL model where 400 persons take a 30-item test. If the same persons take the same test a second time, do the person ability estimates on the second testing correlate more strongly with the ability estimates from the first testing or with the true thetas?

The Algorithm:

- Step 1: Simulation of true item parameters (discrimination, difficulty and guessing) and person location (ability)
- Step 2: Simulation of responses using the true item and person parameters produced in step 1
- Step 3: Estimation of item parameters using the responses produced in step 2
- Step 4: Estimation of ability using the responses produced in step 2
- Step 5: Correlate the estimations of ability using the data produced in step 4

```r
# Specify seed for Random Number Generation to reproduce same results
set.seed(22222, kind = "Mersenne-Twister", normal.kind = "Inversion")

# CONSTANTS
npersons <- 400 # number of test-takers
numitems <- 30 # number of test questions

# True theta, random standard normal values
# A vector of values of the latent variable ("abilities").
true_theta <- rnorm(npersons, mean = 0, sd = 1)

# check
require(psych)
```

```
## Loading required package: psych
##
```

```
## Attaching package: 'psych'
##
## 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
describe(true_theta)
```

```
##   vars   n mean sd median trimmed  mad   min  max range skew kurtosis   se
## 1    1 400 0.06  1      0    0.04 1.05 -2.77 3.15  5.93 0.21    -0.18 0.05
```

```r
str(true_theta)
```

```
##  num [1:400] -0.4408 1.887 -1.234 -1.2711 -0.0139 ...
```

```r
set.seed(1234)

# Notice in a 1PL model discrimination is constant (a = 1) and
# pseudo guessing is zero (c = 0)
true_items <- as.matrix(cbind(alpha = rep(1, numitems),
                              delta = rnorm(numitems, 0, 1),
                              chi = rep(0, numitems)))

# check
head(true_items)
```

```
##      alpha      delta chi
## [1,]     1 -1.2070657   0
## [2,]     1  0.2774292   0
## [3,]     1  1.0844412   0
## [4,]     1 -2.3456977   0
## [5,]     1  0.4291247   0
## [6,]     1  0.5060559   0
```

```r
str(true_items)
```

```
##  num [1:30, 1:3] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:3] "alpha" "delta" "chi"
```

```r
# Simulate responses
require(irtoys)
test <- irtoys::sim(ip = true_items, x = true_theta)
str(test)
```

```
##  num [1:400, 1:30] 0 1 1 1 1 1 1 0 1 0 ...
```

Simulate responses from the 1PL

A matrix of responses: persons as rows, items as columns, entries are either 0 or 1, no missing data

```r
require(irtoys)

# Test 1
set.seed(111)
test1 <- irtoys::sim(ip = true_items, x = true_theta)

# Test 2
set.seed(333)
test2 <- irtoys::sim(ip = true_items, x = true_theta)
```

1.1.2 Estimate item parameters

A list with two elements, est and se, for the parameter estimates and their standard errors, correspondingly. Each element is a matrix with one row per item, and three columns: [,1] item discrimination a, [,2] item difficulty b, and [,3] asymptote c. For the 1PL and 2PL models, all asymptotes are equal to 0; for the 1PL, the discriminations are all equal but not necessarily equal to 1. When ICL is used as estimation engine, se is NULL as ICL does not compute standard errors for the item parameter estimates.

When engine="bilog" and model="1PL" and "rasch"=T, the common value for discriminations is forced to 1, and the sum of the difficulties is 0. When engine="ltm" and model="1PL" and "rasch"=T, the common value for discriminations is forced to 1. Ignored in all other cases. Default is F.

```r
require(irtoys)

# Test 1
set.seed(222)
itemhats1   <- irtoys::est(resp = test1,
                           model = "1PL",
                           engine = "ltm")
# check
# Notice the discrimination parameter (slope), a,
# is essentially 1 (first column), and
# the pseudo-guessing parameter (chance), c ,
# is 0 (third column) in a 1PL model
head(itemhats1$est)
```

```
##          [,1]       [,2] [,3]
## [1,] 1.013378 -1.3231331    0
## [2,] 1.013378  0.4425004    0
## [3,] 1.013378  0.9371251    0
## [4,] 1.013378 -2.3941643    0
## [5,] 1.013378  0.6074424    0
## [6,] 1.013378  0.4424685    0
```

```r
# Test 2
set.seed(444)
itemhats2   <- irtoys::est(resp = test2,
                           model = "1PL",
                           engine = "ltm")

# check
# Notice the discrimination parameter (slope), a,
# is essentially 1 (first column), and
# the pseudo-guessing parameter (chance), c ,
# is 0 (third column) in a 1PL model
head(itemhats2$est)
```

```
##           [,1]       [,2] [,3]
## [1,] 0.9935143 -1.2036061    0
## [2,] 0.9935143  0.2893938    0
## [3,] 0.9935143  0.9803496    0
## [4,] 0.9935143 -2.5729553    0
## [5,] 0.9935143  0.4265448    0
## [6,] 0.9935143  0.4137838    0
```

1.1.3 EAP estimation of ability

Estimates the expectation of the posterior distribution of the latent variable ("ability") for each person. Produces s matrix with the ability estimates in column 1, and their standard errors of measurement (SEM) in column 2, and the number of non-missing reponses in column 3.

```r
require(irtoys)

# Test 1
set.seed(555)
thetahat1 <- irtoys::eap(resp = test1,
                         ip = itemhats1$est,
                         qu = normal.qu())

# check
head(thetahat1)
```

```
##              est       sem  n
## [1,] -0.69820506 0.3688240 30
## [2,]  1.06967307 0.4234196 30
## [3,] -1.27726503 0.3885469 30
## [4,] -0.56183849 0.3655947 30
## [5,]  0.12597595 0.3728586 30
## [6,] -0.01304055 0.3686476 30
```

```r
describe(thetahat1[,2])
```

```
##   vars   n mean   sd median trimmed  mad  min  max range skew kurtosis se
## 1    1 400 0.39 0.03   0.38    0.39 0.02 0.37 0.54  0.17 1.88     3.56  0
```

```r
# Test 2
set.seed(666)
thetahat2 <- irtoys::eap(resp = test2,
                         ip = itemhats2$est,
                         qu = normal.qu())

# check
head(thetahat2)
```

```
##             est       sem  n
## [1,] -0.4679662 0.3715951 30
## [2,]  1.4407840 0.4648550 30
## [3,] -1.4864961 0.4046482 30
## [4,] -0.4679662 0.3715951 30
## [5,]  0.2342032 0.3846806 30
## [6,] -0.1904340 0.3748424 30
```

```r
describe(thetahat2[,2])
```

```
##   vars   n mean   sd median trimmed  mad  min  max range skew kurtosis se
## 1    1 400  0.4 0.03   0.39    0.39 0.02 0.37 0.55  0.18  1.8     3.26  0
```

Combine the true thetas and the ability estimates for the two tests in a data frame

```r
# A matrix of true theta, ability estimates for test 1 and
# ability estimates for test 2
thetas <- data.frame(True_theta = true_theta,
                     estimate_1 = thetahat1[,"est"],
                     estimate_2 = thetahat2[,1])
# check
head(thetas)
```

```
##    True_theta  estimate_1 estimate_2
## 1 -0.44084939 -0.69820506 -0.4679662
## 2  1.88703812  1.06967307  1.4407840
## 3 -1.23398269 -1.27726503 -1.4864961
## 4 -1.27108348 -0.56183849 -0.4679662
## 5 -0.01392890  0.12597595  0.2342032
## 6 -0.06256098 -0.01304055 -0.1904340
```

1.1.4 Comparison of Ability Estimates

The correlations of the two estimates with the true values are around low 0.90's which are larger than the correlation between the estimates of the two tests, about mid 0.80's. This makes sense, since the estimates reflect the randomness within the simulated data while the true values do not.

```r
cor(thetas)
```

```
##            True_theta estimate_1 estimate_2
## True_theta  1.0000000  0.9079954  0.9131782
## estimate_1  0.9079954  1.0000000  0.8365036
## estimate_2  0.9131782  0.8365036  1.0000000
```

1.1.5 Correlation Matrix with Significance

Prefix	Significance
•	0.1
*	0.05
* *	0.01
* * *	0.001

```r
# histograms on the diagonal
panel.hist.density <-
  function(x,...)
  {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks
    nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col = "dodgerblue")
    tryd <- try( d <- density(x, na.rm = TRUE, bw = "nrd", adjust = 1.2),
                 silent = TRUE)

    if (class(tryd) != "try-error") {
      d$y <- d$y/max(d$y)
      lines(d, lwd = 2, col = "tomato")
    }
  } # end panel.hist.density

dump("panel.hist.density", file = "panel.hist.density.R")
```

```r
# scatter plot with tolerance ellipsoids and lowess line on the lower-triangle
panel.smooth <-
  function(x, y,
           col = par("col"),
           bg = NA,
           pch = 21,
           cex = .5,
           col.smooth = "tomato",
           span = 2/3,
           iter = 3, ...){
    require(cluster)

    points(x, y,
```

```
          pch = pch,
          col = col,
          bg = bg,
          cex = cex)

  ok <- is.finite(x) & is.finite(y)

  if (any(ok))
    lines(stats::lowess(x[ok], y[ok], f = span, iter = iter),
          col = col.smooth,
          lwd = 2, ...)

  # classical and robust cov.matrix ellipsoids
  X <- cbind(x,y)
  C.ls <- cov(X)
  m.ls <- colMeans(X)

  # calculates a 99% ellipsoid
  d2.99 <- qchisq(0.99, df = 2)

  # classical cov.matrix ellipsoids
  lines(ellipsoidPoints(C.ls, d2.99, loc = m.ls),
        lwd = 2,
        lty = 3,
        col = "purple")

  if (require(MASS)) {
    Cxy <- cov.rob(cbind(x,y))
    # robust cov.matrix ellipsoids
    lines(ellipsoidPoints(Cxy$cov, d2 = d2.99, loc = Cxy$center),
          lty = 3,
          lwd = 2,
          col = "brown")
  }
} # end panel.smooth
```

```r
dump("panel.smooth.R", file = "panel.smooth.R")
```

```r
# correlations with significance on the upper panel
panel.cor <-
  function(x, y,
           method = "pearson",
           digits = 3,
           cex.cor = 1,
           no.col = FALSE) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- cor(x, y, method = method)
    ra <- cor.test(x, y, method = method)$p.value
    txt <- format(c(r, 0.123456789), digits = digits)[1]

    # mark significant correlations
    prefix <- ""
    if (ra <= 0.1) prefix <- "•"
    if (ra <= 0.05) prefix <- "*"
    if (ra <= 0.01) prefix <- "**"
    if (ra <= 0.001) prefix <- "***"
    if (no.col)
    {
      color <- "tomato"
      if (r < 0) {if (ra <= 0.001) sig <- 4 else sig <- 3}
      else {if (ra <= 0.001) sig <- 2 else sig <- 1}
    }
    else
    {
      sig <- 1
      if (ra <= 0.001) sig <- 2
      color <- 2
      if (r < 0) color <- 4
    }
    txt <- paste(txt, prefix, sep = "\n")
```

```
    if (missing(cex.cor)) cex.cor <- 0.5/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r, col = "maroon")
  } # end panel.cor

dump("panel.cor", file = "panel.cor.R")
```

Both of the correlations between the estimates and the true values are 0.97, which are slightly larger than the correlation between the estimates of ability (r = 0.94). This makes sense, since the estimates reflect the randomness within the simulated data while the true values do not.

```
library(graphics, quietly = TRUE)
graphics::pairs(~True_theta + estimate_1 + estimate_2,
      data = thetas,
      lower.panel = panel.smooth,
      upper.panel = panel.cor,
      diag.panel = panel.hist.density,
      main = "Scatterplot Matrix with Correlations",
      na.action = stats::na.omit)
```

```
## Loading required package: cluster
```

1.1.6 Comparison of Item Difficulties and Person Locations by 1PL, 2PL and 3PL Models

Simulate data for a 3-PL model where 1000 persons take a 40-item test. The test should be somewhat easy (average examinee score greater than 50% correct). Include a reasonable or realistic amount of guessing. Estimate parameters for 1-Pl, 2-PL model and 3-PL models. Do the item difficulty parameter estimates for the 2-PL model correlate more strongly with the estimates for the 3-PL model or with the true values?

The Algorithm:

- Step 1: Simulation of true item parameters (discrimination, difficulty and guessing) and person location (ability).
- Step 2: Simulation of response data using the true item and person parameters produced in step 1.

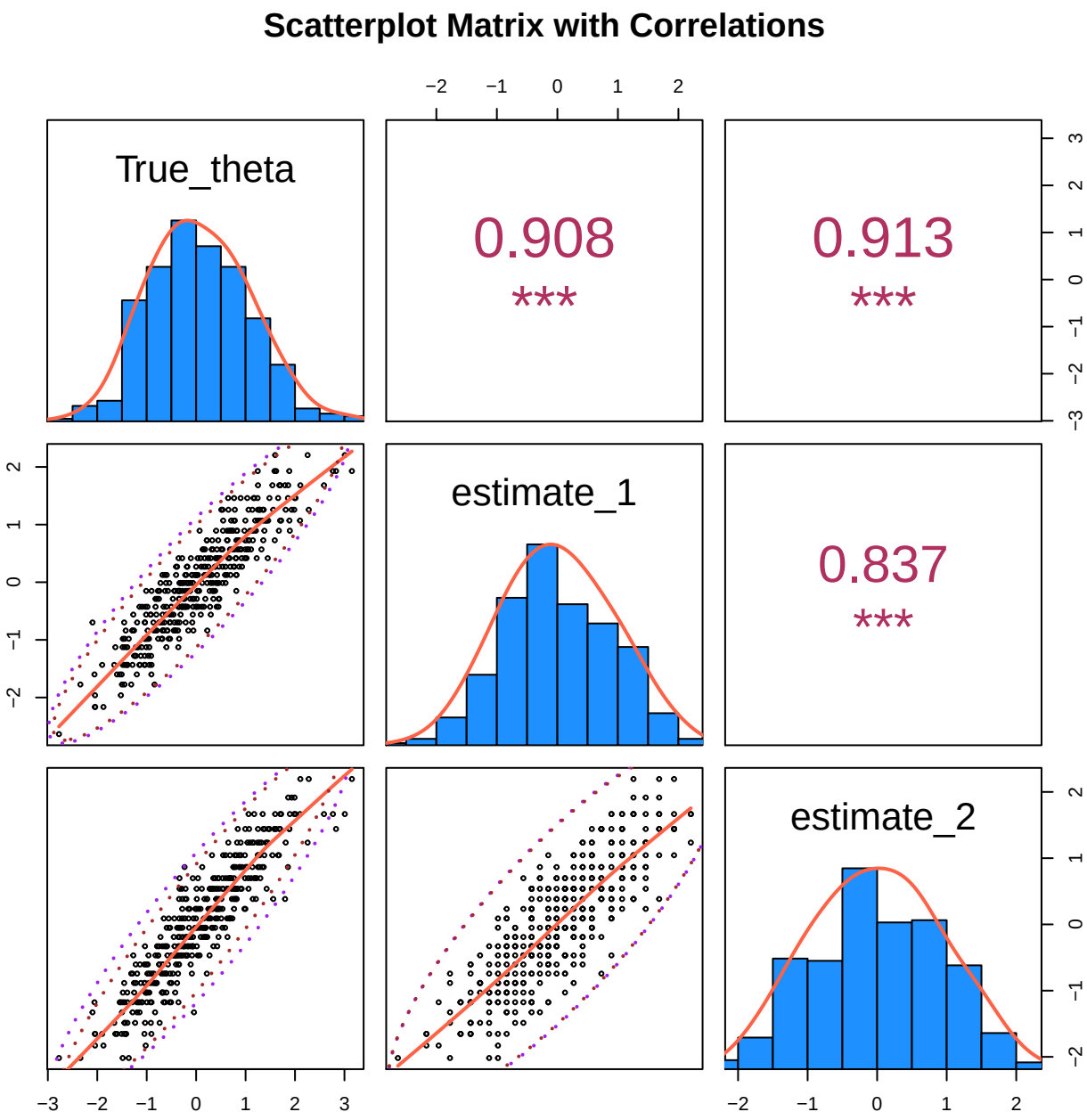

Scatterplot Matrix with Correlations
True_theta
0.908
***
0.913
***
estimate_1
0.837
***
estimate_2

Figure 1:

- Step 3: Estimation of item difficulties and peson locationsby 1PL, 2PL and 3PL models using the responses produced in step 2.
- Step 4: Correlate the estimations of item difficulties and person locations using the data produced in step 3.

```r
# CONSTANTS
npersons <- 2000
nitems <- 100

set.seed(77777777)
true_theta <- rnorm(npersons, mean = 0, sd = 1)

# check
describe(true_theta)
```

```
##   vars    n mean   sd median trimmed mad  min  max range  skew kurtosis
## 1    1 2000 0.03 0.99   0.03    0.03   1 -3.5 3.52  7.02 -0.01    -0.12
##     se
## 1 0.02
```

```r
str(true_theta)
```

```
##  num [1:2000] 1.778 -1.085 -1.008 1.096 0.275 ...
```

```r
set.seed(5555)

true_items <- cbind(a = rnorm(nitems, 1, .4),
                    b = rnorm(nitems, 0, 1),
                    c = rnorm(nitems, .1, .04))

# check
describe(true_items)
```

```
##   vars   n  mean   sd median trimmed  mad   min  max range skew kurtosis
## a    1 100  1.03 0.34   1.04    1.03 0.36  0.34 1.97  1.63 0.06    -0.45
## b    2 100 -0.06 1.03  -0.03   -0.07 1.02 -2.53 2.54  5.06 0.07    -0.39
```

```
## c    3 100  0.09 0.04   0.09    0.09 0.04  0.00 0.18  0.18 0.11    -0.45
##     se
## a 0.03
## b 0.10
## c 0.00
```

```
colnames(true_items) <- c("alpha", "beta", "chi")
rownames(true_items) <- paste("p", 1:nrow(true_items), sep = "")
head(true_items)
```

```
##        alpha        beta        chi
## p1 0.6923330 -0.14255711 0.13270900
## p2 0.8209209  1.08755760 0.11026175
## p3 0.4382826 -0.01746314 0.11641760
## p4 1.1568194  0.28882302 0.06249148
## p5 1.2977297 -0.49266077 0.07819389
## p6 0.4155925  0.08750811 0.06510608
```

```
# Item response function (irf) plot
palette(rainbow(nitems))
plot(irf(true_items),
     label = TRUE,
     co = NA)
```

Simulate response data

```
#simulate raw data
require(irtoys)
set.seed(88888)
sim.test <- irtoys::sim(true_items, true_theta)
dimnames(sim.test) <- list(paste("p", 1:nrow(sim.test), sep = ""),
                           paste("i", 1:ncol(sim.test), sep = ""))

# check
str(sim.test)
```

```
##  num [1:2000, 1:100] 1 0 0 0 1 0 1 1 0 0 ...
```

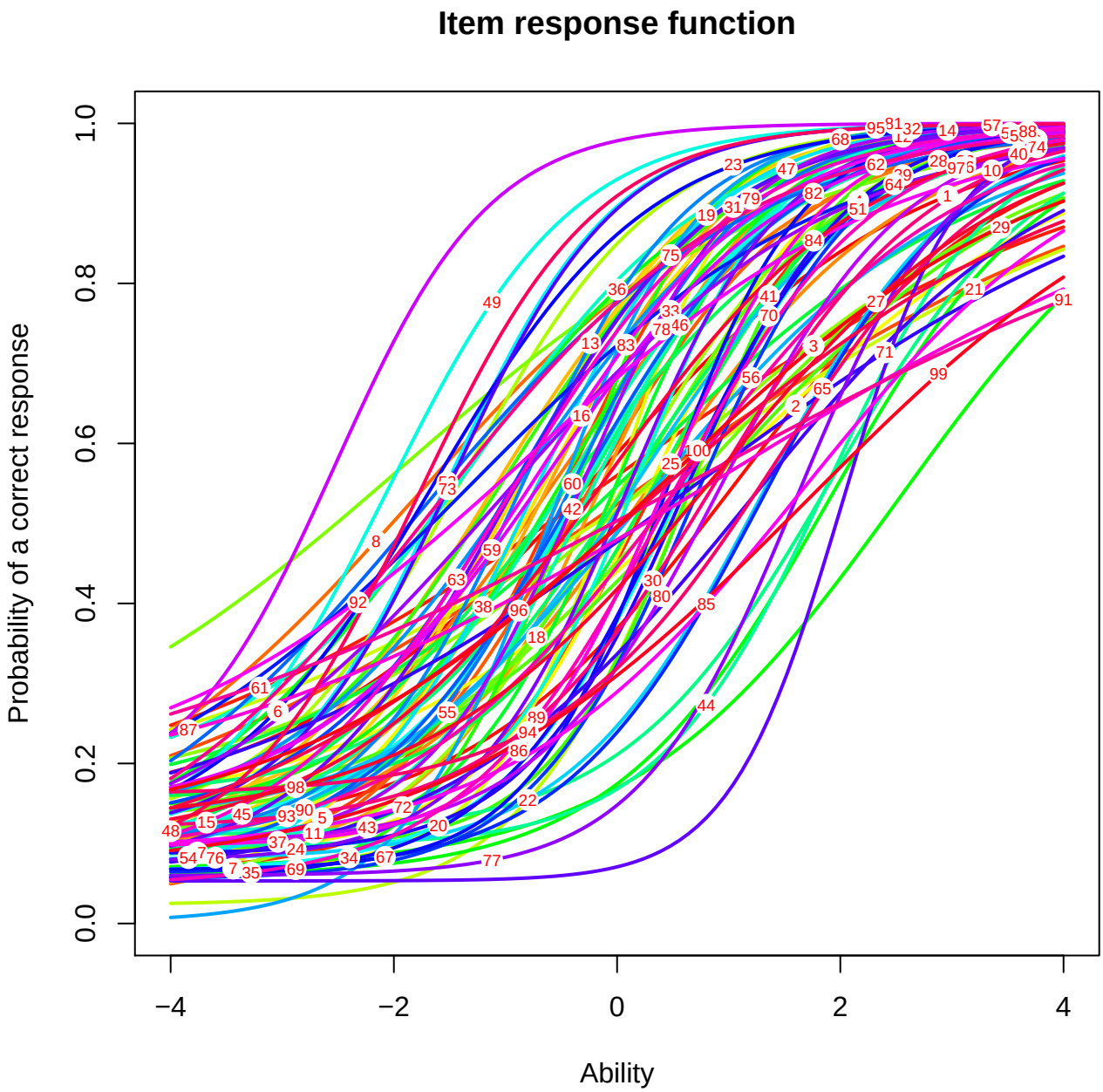

Item response function
Probability of a correct response
0.0
0.2
0.4
0.6
0.8
1.0
−4
−2
0
2
4
Ability

Figure 2:

```
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:2000] "p1" "p2" "p3" "p4" ...
##   ..$ : chr [1:100] "i1" "i2" "i3" "i4" ...
```

Calculate raw test scores

```r
scores <- NULL
for (i in 1:npersons) {scores[i] <- sum(sim.test[i,1:nitems])}
summary(scores)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   44.00   57.00   56.11   69.00   95.00
```

Scores as percentages

```r
percents <- scores/nitems
summary(percents)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0900  0.4400  0.5700  0.5611  0.6900  0.9500
```

1.1.7 1PL Model

```r
# Estimate item parameters for 1PL model
require(irtoys)
set.seed(3333)

delta_1PL <- irtoys::est(resp = sim.test,
                         model = "1PL",
                         engine = "ltm")

summary(delta_1PL$est)
```

```
##        V1                V2                 V3
##  Min.   :0.8324   Min.   :-3.8947   Min.   :0
##  1st Qu.:0.8324   1st Qu.:-1.0830   1st Qu.:0
```

```
##  Median :0.8324   Median :-0.3613   Median :0
##  Mean   :0.8324   Mean   :-0.3701   Mean   :0
##  3rd Qu.:0.8324   3rd Qu.: 0.2583   3rd Qu.:0
##  Max.   :0.8324   Max.   : 2.5577   Max.   :0
```

```
## To write a CSV file for input
write.table(delta_1PL$est,
            file = "D_1PL.csv",
            col.names = c("a",  "b",  "c"),
            row.names = TRUE)

set.seed(1111111)

# EAP estimation of ability
theta_1PL <- irtoys::eap(resp = sim.test,
                         ip = delta_1PL$est,
                         qu = normal.qu())

summary(theta_1PL)
```

```
##       est                 sem               n
##  Min.   :-3.05343   Min.   :0.2101   Min.   :100
##  1st Qu.:-0.63900   1st Qu.:0.2409   1st Qu.:100
##  Median : 0.02141   Median :0.2672   Median :100
##  Mean   : 0.01602   Mean   :0.2661   Mean   :100
##  3rd Qu.: 0.68444   3rd Qu.:0.2903   3rd Qu.:100
##  Max.   : 2.91340   Max.   :0.4112   Max.   :100
```

```
# Item response function (irf) plot
palette(rainbow(nitems))
plot(irf(delta_1PL$est),
     main = "Item characteristic curve 1PL",
     label = TRUE,
     co = NA)
```

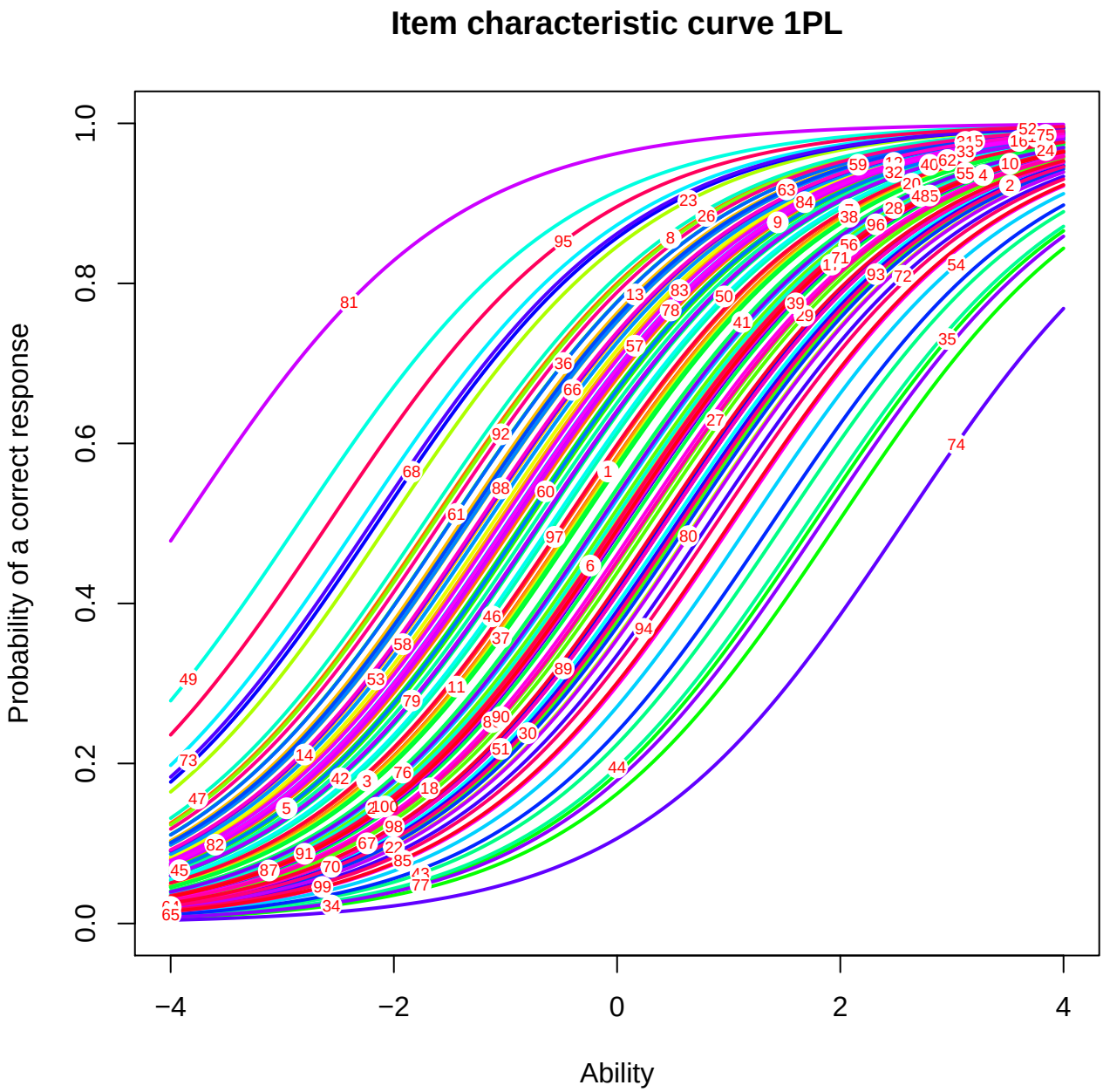

Item characteristic curve 1PL
Probability of a correct response
0.0
0.2
0.4
0.6
0.8
1.0
−4
−2
0
2
4
Ability

Figure 3:

1.1.8 2PL Model

```
# Estimate item parameters for 2PL model
require(irtoys)
set.seed(111111)

delta_2PL <- irtoys::est(resp = sim.test,
                         model = "2PL",
                         engine = "ltm")

summary(delta_2PL$est)
```

```
##        V1                V2                 V3
##  Min.   :0.2375   Min.   :-2.9030   Min.   :0
##  1st Qu.:0.5993   1st Qu.:-0.9597   1st Qu.:0
##  Median :0.8736   Median :-0.2768   Median :0
##  Mean   :0.8688   Mean   :-0.2802   Mean   :0
##  3rd Qu.:1.1072   3rd Qu.: 0.3548   3rd Qu.:0
##  Max.   :1.5397   Max.   : 3.0418   Max.   :0
```

```
## To write a CSV file for input
write.table(delta_2PL$est,
            file = "D_2PL.csv",
            col.names = c("a",  "b",  "c"),
            row.names = TRUE)

set.seed(1111111)

# EAP estimation of ability
theta_2PL <- irtoys::eap(resp = sim.test,
                         ip = delta_2PL$est,
                         qu = normal.qu())

summary(theta_2PL)
```

```
##       est                 sem               n
```

```
##  Min.   :-2.895781   Min.   :0.1699   Min.   :100
##  1st Qu.:-0.630398   1st Qu.:0.2149   1st Qu.:100
##  Median : 0.008659   Median :0.2591   Median :100
##  Mean   : 0.036780   Mean   :0.2537   Mean   :100
##  3rd Qu.: 0.693137   3rd Qu.:0.2871   3rd Qu.:100
##  Max.   : 3.060449   Max.   :0.4486   Max.   :100
```

```r
# Item response function (irf) plot
palette(rainbow(nitems))
plot(irf(delta_2PL$est),
     main = "Item characteristic curve 2PL",
     label = TRUE,
     co = NA)
```

1.1.9 3 PL Model

```r
# Estimate item parameters for 3PL model
require(irtoys)
set.seed(111111111)

delta_3PL <- irtoys::est(resp = sim.test,
                         model = "3PL",
                         engine = "ltm")
```

```
## Warning in tpm(resp, control = list(GHk = nqp), max.guessing = 1): Hessian matrix at con
```

```r
summary(delta_3PL$est)
```

```
##        V1               V2                  V3
##  Min.   :0.2468   Min.   :-3.005847   Min.   :0.001748
##  1st Qu.:0.7700   1st Qu.:-0.547643   1st Qu.:0.033816
##  Median :1.0582   Median : 0.016410   Median :0.089167
##  Mean   :1.0262   Mean   : 0.008986   Mean   :0.109690
##  3rd Qu.:1.2163   3rd Qu.: 0.658943   3rd Qu.:0.174916
##  Max.   :2.0939   Max.   : 2.902543   Max.   :0.459564
```

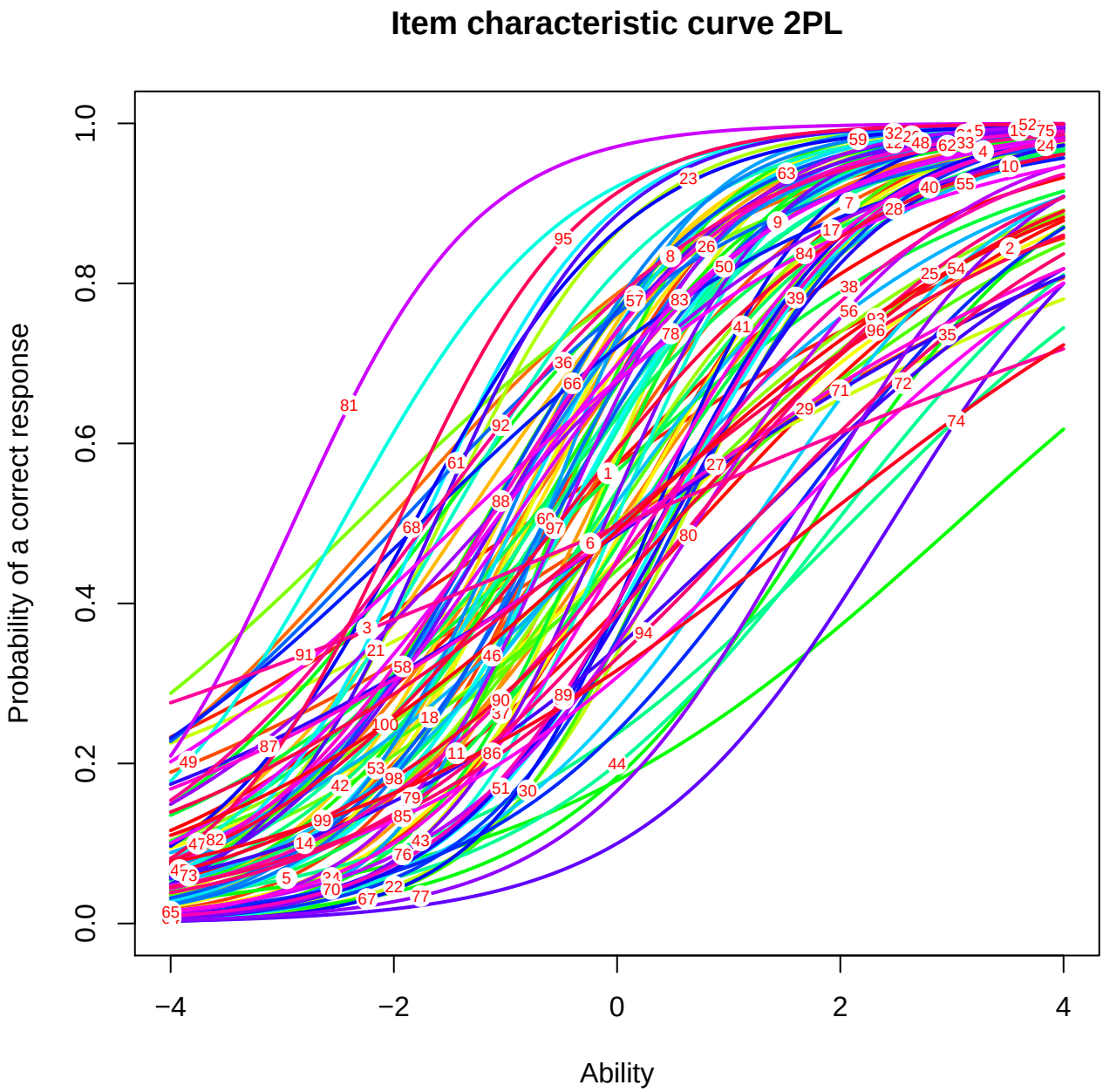

Item characteristic curve 2PL
Probability of a correct response
Ability
0.0
0.2
0.4
0.6
0.8
1.0
−4
−2
0
2
4

Figure 4:

```r
## To write a CSV file for input
write.table(delta_3PL$est,
            file = "D_3PL.csv",
            col.names = c("a",  "b",  "c"),
            row.names = TRUE)

set.seed(11111111)
# EAP estimation of ability
theta_3PL <- irtoys::eap(resp = sim.test,
                         ip = delta_3PL$est,
                         qu = normal.qu())

summary(theta_3PL)
```

```
##       est                 sem                n
##  Min.   :-3.28773   Min.   :0.1576   Min.   :100
##  1st Qu.:-0.61927   1st Qu.:0.2148   1st Qu.:100
##  Median : 0.02540   Median :0.2579   Median :100
##  Mean   : 0.03265   Mean   :0.2527   Mean   :100
##  3rd Qu.: 0.69954   3rd Qu.:0.2860   3rd Qu.:100
##  Max.   : 2.92870   Max.   :0.4594   Max.   :100
```

```r
# Item response function (irf) plot
palette(rainbow(40))
plot(irf(delta_3PL$est),
     main = "Item characteristic curve 3PL",
     label = TRUE,
     co = NA)
```

```r
# a matrix of true DELTA and 2PL and 3PL item difficulty estimates
deltas <- data.frame(true_delta = true_items[,2],
                     delta_1PL = delta_1PL$est[,2],
                     delta_2PL = delta_2PL$est[,2],
                     delta_3PL = delta_3PL$est[,2])

head(deltas)
```

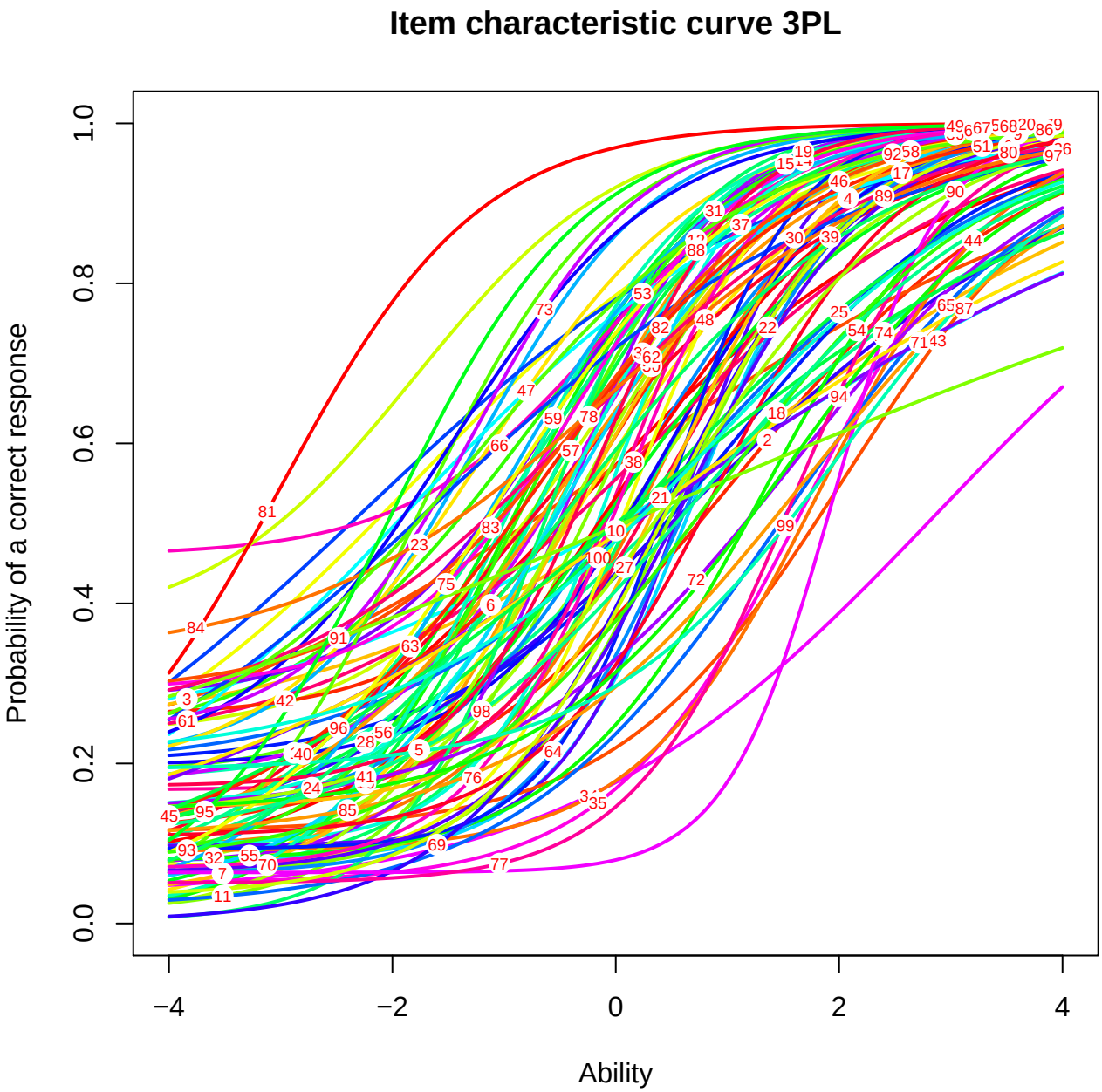

Item characteristic curve 3PL
Probability of a correct response
Ability
0.0
0.2
0.4
0.6
0.8
1.0
−4
−2
0
2
4

Figure 5:

```
##     true_delta   delta_1PL   delta_2PL     delta_3PL
## p1 -0.14255711 -0.39383189 -0.51288466 -0.3976077381
## p2  1.08755760  0.54635016  0.72356283  1.0566942608
## p3 -0.01746314 -0.40507174 -0.80309556 -0.0003398069
## p4  0.28882302  0.06944937  0.07624994  0.3164903119
## p5 -0.49266077 -0.81832735 -0.62081509 -0.5106294985
## p6  0.08750811  0.01400952  0.03617528  1.2912865876
```

```r
describe(deltas)
```

```
##            vars   n  mean   sd median trimmed  mad   min  max range  skew
## true_delta    1 100 -0.06 1.03  -0.03   -0.07 1.02 -2.53 2.54  5.06  0.07
## delta_1PL     2 100 -0.37 1.09  -0.36   -0.37 1.03 -3.89 2.56  6.45 -0.10
## delta_2PL     3 100 -0.28 1.10  -0.28   -0.33 0.97 -2.90 3.04  5.94  0.38
## delta_3PL     4 100  0.01 1.06   0.02    0.01 0.96 -3.01 2.90  5.91 -0.04
##            kurtosis   se
## true_delta    -0.39 0.10
## delta_1PL      0.52 0.11
## delta_2PL      0.29 0.11
## delta_3PL      0.07 0.11
```

```r
write.table(deltas,
            file = "delta_estimates.csv",
            sep = ",",
            col.names = c("true_delta",  "delta_1PL",  "delta_2PL",  "delta_3PL"),
            row.names = TRUE,
            qmethod = "double")

# Item difficulty parameter estimates for the 2PL and 3PL model correlate more strongly u
# (r=0.87 and r=0.85, respectively) than between themself (r=0.83).
cor(deltas)
```

```
##            true_delta delta_1PL delta_2PL delta_3PL
## true_delta  1.0000000 0.9676396 0.9834786 0.9554793
## delta_1PL   0.9676396 1.0000000 0.9700120 0.9389976
## delta_2PL   0.9834786 0.9700120 1.0000000 0.9515130
## delta_3PL   0.9554793 0.9389976 0.9515130 1.0000000
```

Correlation Matrix

There were not significant differences between 1PL, 2PL and 3PL models in predicting the true item locations (correlations .97, .98 and .96, respectively).

```
pairs(~true_delta + delta_1PL + delta_2PL + delta_3PL,
      data = deltas,
      lower.panel = panel.smooth,
      upper.panel = panel.cor,
      diag.panel = panel.hist.density,
      main = "Scatterplot Matrix with Correlations",
      na.action = stats::na.omit)
```

```
thetas <- data.frame(true_theta = true_theta,
                     theta_1PL = theta_1PL[,"est"],
                     theta_2PL = theta_2PL[,"est"],
                     theta_3PL = theta_3PL[,"est"])
rownames(thetas) <- paste("p", 1:nrow(thetas), sep = "")
head(thetas)
```

```
##    true_theta  theta_1PL  theta_2PL  theta_3PL
## p1  1.7775119  1.5081855  1.8010950  1.7180526
## p2 -1.0854741 -0.6871546 -0.6809017 -0.6699946
## p3 -1.0075611 -0.9528740 -0.9400242 -0.9603400
## p4  1.0962103  1.1511140  1.1043136  1.1008772
## p5  0.2747158  0.1093575  0.1464077  0.1674511
## p6 -0.2802855 -0.5985489 -0.5896257 -0.5679569
```

```
write.table(thetas,
            file = "theta_estimates.csv",
            sep = ",",
            col.names = c("true_theta",  "theta_1PL",  "theta_2PL",  "theta_3PL"),
            row.names = TRUE,
            qmethod = "double")
```

All three IRT models performed well in predicting abilities of respondents. The correlations between the true person location and all three IRT models (1PL, 2PL and 3PL) were about 0.96.

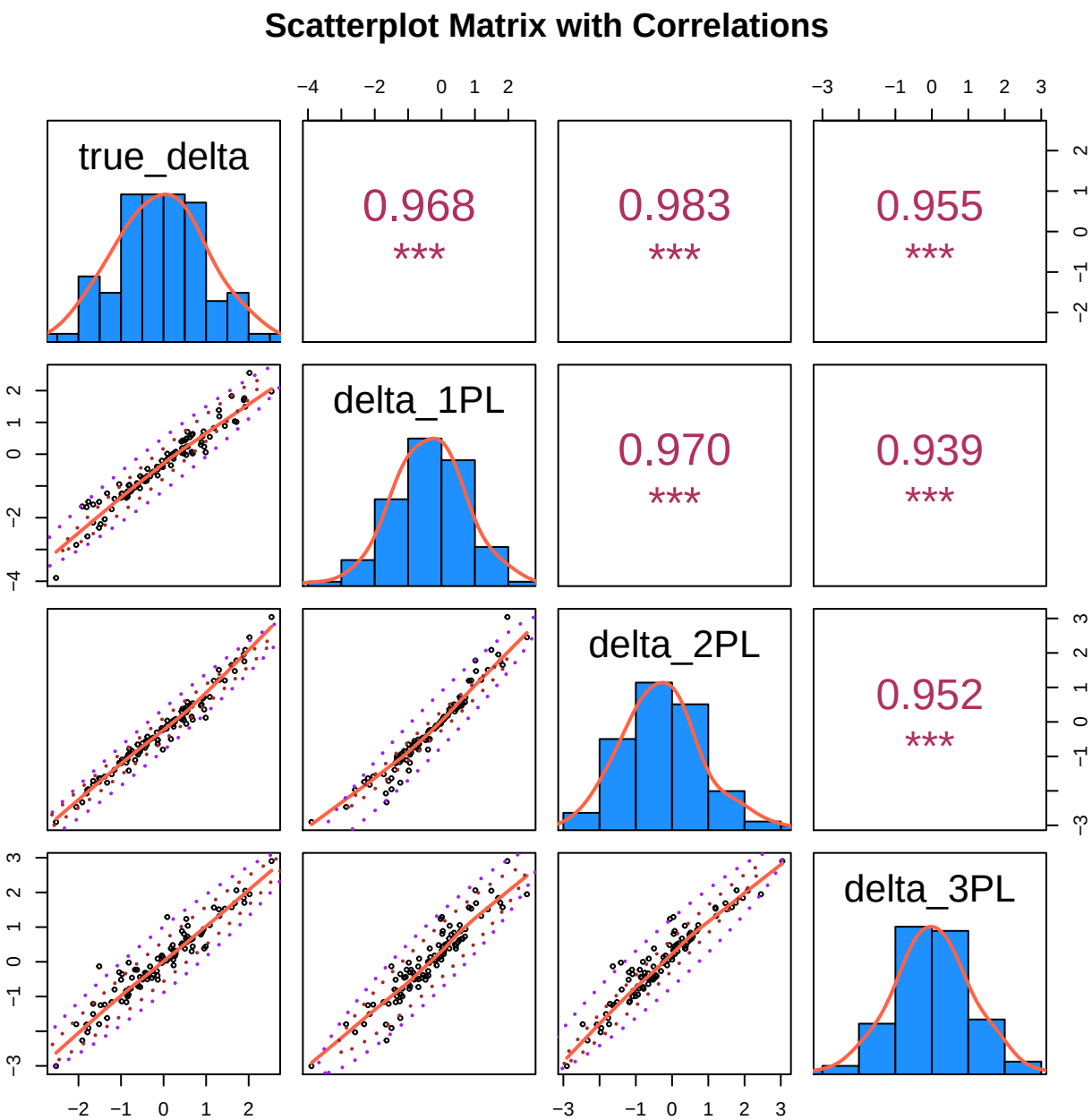

Scatterplot Matrix with Correlations
true_delta
0.968
***
0.983
***
0.955
***
delta_1PL
0.970
***
0.939
***
delta_2PL
0.952
***
delta_3PL

Figure 6:

```r
pairs(~true_theta + theta_1PL + theta_2PL + theta_3PL,
      data = thetas,
      lower.panel = panel.smooth,
      upper.panel = panel.cor,
      diag.panel = panel.hist.density,
      main = "Scatterplot Matrix with Correlations",
      na.action = stats::na.omit)
```

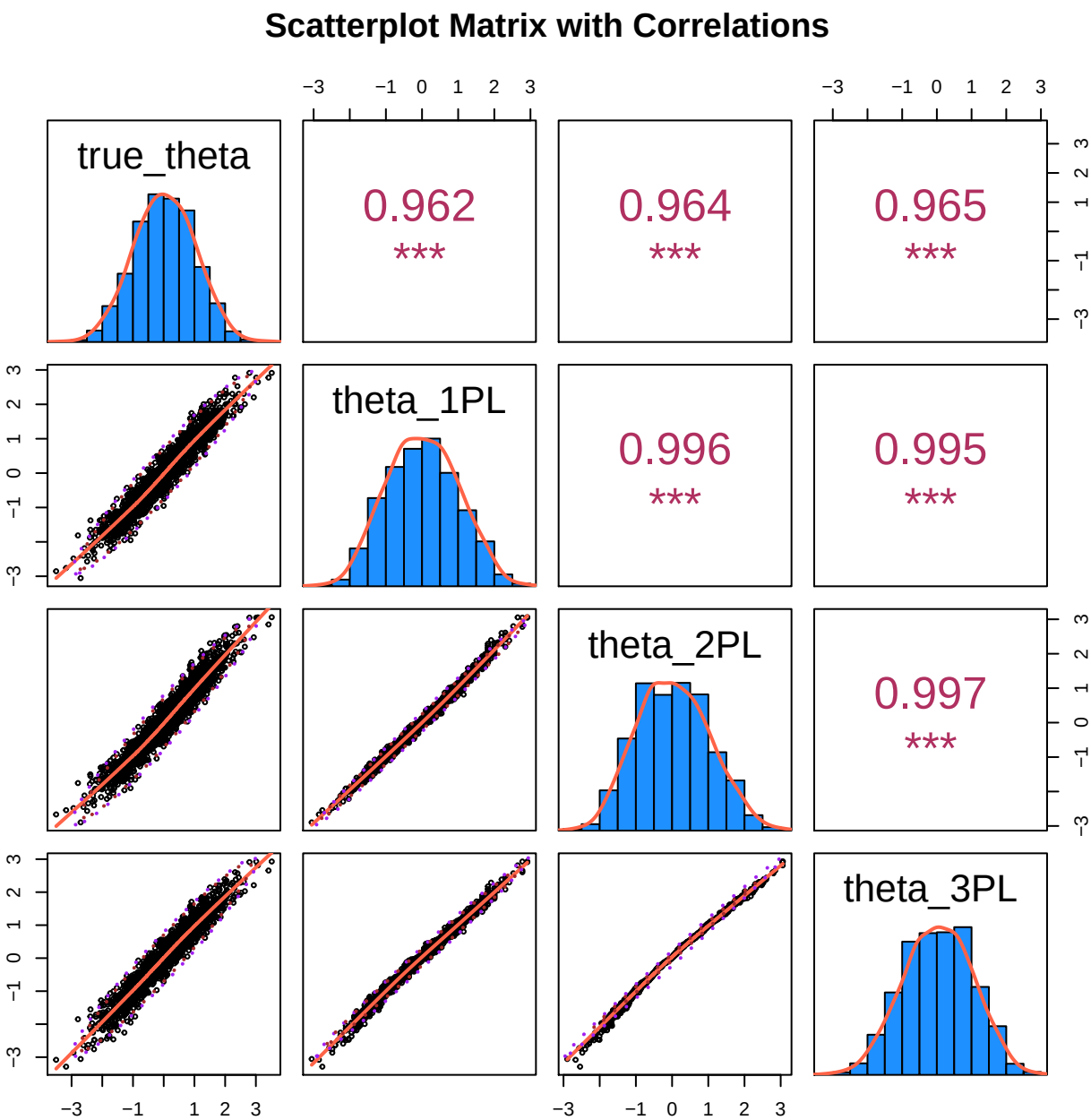


Figure 7:

1.1.10 Comparing Likelihood Ratio Tests

The Theory and Practice of Item Response Theory (Ayala 2013):

A likelihood ratio test is a statistical test used to compare the fit of two models, one of which (the null or the reduced model) is a special case of the other (the alternative or the full model). The test is based on the likelihood ratio, which expresses how many times more likely the data are under one model than the other. This likelihood ratio, or equivalently its logarithm, can then be used to compute a p-value, or compared to a critical value to decide whether to reject the null model in favour of the alternative model. Each of the two competing models, the null model and the alternative model, is separately fitted to the data and the log-likelihood recorded. The test statistic, G^2 *is twice the difference in these log-likelihoods:*

$$\Delta G^2 = -2ln(L_R) - (-2ln(L_F))$$

The degrees of freedom for evaluating the significance of D statistics is the difference in the number of parameters between the full model and the reduced model. D statistic is distributed as a χ^2 *when the sample size is large and the full model is statistically significant (p < 0.05). A nonsignificant result indicates that the additional complexity (increased number of item parameters) is unnecessary. For example, if the comparison of 2PL and 3PL models is not significant, then the pseudo-guessing parameter is not necessary to improve model-data fit over and above that obtained wit the 2PL model.*

1.1.11 Calculate model fit

The df for a model is given by $2^K -$ (number of parameters) where K is the number of items on the instrument and the number of parameters is based on the model and the number of items.

For the 3PL model, there are three item parameters $(\alpha_j, \delta_j, \chi_j)$ and for a 100-item instrument the number of item parameters is $3 * 100 = 300$. Therefore, the $df = 2^{100} - 300 - 1 = 1.267651 \times 10^{30}$.

For the 2PL model, there are two item parameters (α_j, δ_j) and for a 100-item instrument the number of item parameters is $2 * 100 = 200$. Therefore, the $df = 2^{200} - 200 - 1$.

For the 1PL model, each item has a location (δ_j) and a common discrimination (α_j) and for a 100-item instrument the number of item parameters is $100 + 1 = 101$. Therefore, the $df = 2^{100} - 101 - 1$.

```
D_1PL <- read.table("D_1PL.csv")
head(D_1PL)
```

```
##          a           b c
```

```
## i1 0.83237 -0.39383189 0
## i2 0.83237  0.54635016 0
## i3 0.83237 -0.40507174 0
## i4 0.83237  0.06944937 0
## i5 0.83237 -0.81832735 0
## i6 0.83237  0.01400952 0
```

```r
D_2PL <- read.table("D_2PL.csv")
head(D_2PL)
```

```
##            a           b c
## i1 0.5816239 -0.51288466 0
## i2 0.6036604  0.72356283 0
## i3 0.3727341 -0.80309556 0
## i4 1.0372320  0.07624994 0
## i5 1.2032374 -0.62081509 0
## i6 0.3607031  0.03617528 0
```

```r
D_3PL <- read.table("D_3PL.csv")
head(D_3PL)
```

```
##            a             b          c
## i1 0.5923792 -0.3976077381 0.03248629
## i2 0.7578926  1.0566942608 0.10670307
## i3 0.4317820 -0.0003398069 0.14363132
## i4 1.2269767  0.3164903119 0.09424311
## i5 1.2387730 -0.5106294985 0.05063588
## i6 0.5283073  1.2912865876 0.22997945
```

```r
source("likelihood.R")

mytest <- read.csv("test.csv", header = TRUE, na.strings = " ")
# CONSTANTS
(K <- ncol(mytest)) # Number of items
```

```
## [1] 15
```

```r
(N <- nrow(mytest)) # Number of examinees
```

```
## [1] 450
```

```r
(NumParams <- c(K + 1, 2*K, 3*K))
```

```
## [1] 16 30 45
```

```r
(DegreesFreedom <- c(2^K - (K + 1) - 1, 2^K - (2*K) - 1, 2^K - (3*K) - 1))
```

```
## [1] 32751 32737 32722
```

```r
# D = -2 ln(L)
D1PL <- -2 * log(max(likelihood(par.mat = D_1PL, resp.mat = mytest)))
D2PL <- -2 * log(max(likelihood(par.mat = D_2PL, resp.mat = mytest)))
D3PL <- -2 * log(max(likelihood(par.mat = D_3PL, resp.mat = mytest)))

## D = D.reduced - D.full = -2ln(Lr) - (-2ln(Lf))
deltaD12 <- D1PL - D2PL
deltaD23 <- D2PL - D3PL

# deltaRSQ = (GSQr - GSQf)/GSQr
deltaR12 <- (D1PL - D2PL)/D1PL
deltaR23 <- (D2PL - D3PL)/D2PL
```

1.1.12 Akaike Information Criterion (AIC)

$$AIC = -2lnL + 2 \times (\text{number of parameters})$$

```r
# AIC = -2lnL + 2*Nparm
AIC1PL <- D1PL + 2*(K+1)
AIC2PL <- D2PL + 2*(K*2)
AIC3PL <- D3PL + 2*(K*3)

AIC <- c(AIC1PL, AIC2PL, AIC3PL)
```

1.1.13 Bayesian Criterion Information (BIC)

$$BIC = -2lnL + ln(N) \times (\text{number of parameters})$$

The model with the smalest BIC indicates the model with the best comparative fit and tends to favor constraided (reduced) models.

```
# BIC = -2lnL + ln(N)*Nparm
BIC1PL <- D1PL + log(N)*(K+1)
BIC2PL <- D2PL + log(N)*(K*2)
BIC3PL <- D3PL + log(N)*(K*3)

BIC <- c(BIC1PL, BIC2PL, BIC3PL)
```

```
(Model <- c("1PL","2PL","3PL"))
```

```
## [1] "1PL" "2PL" "3PL"
```

```
Negative2LL <- c(D1PL, D2PL, D3PL)
RelativeChange <- c(NA, deltaD12, deltaD23)
(comparison <-
  data.frame(Model, Negative2LL, DegreesFreedom, RelativeChange, NumParams, AIC, BIC))
```

```
##   Model Negative2LL DegreesFreedom RelativeChange NumParams       AIC
## 1   1PL   -23.40212          32751             NA        16  8.597883
## 2   2PL   -23.40212          32737       0.000000        30 36.597883
## 3   3PL   -21.56441          32722      -1.837707        45 68.435590
##         BIC
## 1  74.34584
## 2 159.87531
## 3 253.35173
```

Using a more complex 2PL model results in a deterioration of fit by 30% over the 1PL model and using 3PL model results in a slight improvement of fit by 3% over the 2PL model. Overall 1Pl model fits significantly better than the 2PL and 3PL models.

1.2 Evidence for Guessing

To evaluate the prevalence of guessing, we take the lowest scoring examinees and see how they performed on the most difficult items. There should be little success for these persons on these items if guessing is not a factor.

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

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

```
# set the default working directory
setwd("/Users/salvadorcastro/Desktop/R Files/IRT/Part2")

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

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

Import the data file

```
# A data frame of responses: persons (450) as rows, items (15) as columns,
# entries are either 0 or 1, no missing values
mytest <- read.csv("test.csv", header = TRUE, na.strings = " ")

# CONSTANTS
(npersons <- nrow(mytest))
```

```
## [1] 450
```

```
(numitems <- ncol(mytest))
```

```
## [1] 15
```

```r
# Check
str(mytest)
```

```
## 'data.frame':    450 obs. of  15 variables:
##  $ V1 : int  0 0 0 1 0 0 1 0 0 0 ...
##  $ V2 : int  0 1 1 1 1 1 1 1 0 1 ...
##  $ V3 : int  1 1 1 1 1 0 1 0 0 1 ...
##  $ V4 : int  0 1 0 0 0 0 1 0 0 1 ...
##  $ V5 : int  1 0 0 0 0 1 0 1 1 0 ...
##  $ V6 : int  0 0 1 1 1 1 1 0 0 1 ...
##  $ V7 : int  0 1 1 1 1 1 1 0 0 1 ...
##  $ V8 : int  0 0 1 1 1 0 1 0 0 1 ...
##  $ V9 : int  0 0 1 1 1 0 1 0 0 1 ...
##  $ V10: int  1 1 1 1 1 1 1 0 1 1 ...
##  $ V11: int  0 1 1 1 1 1 1 1 1 1 ...
##  $ V12: int  0 1 0 0 0 0 1 0 0 1 ...
##  $ V13: int  0 0 1 1 0 0 1 0 0 1 ...
##  $ V14: int  0 0 0 1 0 0 1 0 0 1 ...
##  $ V15: int  1 1 1 1 1 0 1 0 1 1 ...
```

Estimate of the IRT item parameters using ltm (Latent Trait Model).

```r
library(irtoys)
est1PL <- irtoys::est(resp = mytest,
                      model = "1PL",
                      engine = "ltm")

head(est1PL$est)
```

```
##      [,1]      [,2] [,3]
## V1 1.1925  1.904672    0
## V2 1.1925 -1.956884    0
## V3 1.1925 -1.626909    0
## V4 1.1925  1.816504    0
## V5 1.1925  0.091406    0
## V6 1.1925 -0.531918    0
```

Estimates of the expectation of the posterior distribution of the latent variable ("ability") for each person.

```
theta1PL <- irtoys::eap(resp = mytest,
                        ip = est1PL$est,
                        qu = normal.qu())

head(theta1PL)
```

```
##           est     sem  n
## [1,] -1.30373 0.50181 15
## [2,] -0.13636 0.49487 15
## [3,]  0.45816 0.50510 15
## [4,]  1.08973 0.52656 15
## [5,]  0.15786 0.49892 15
## [6,] -0.71640 0.49293 15
```

```
head(mydf <- data.frame(mytest, thetahat = theta1PL[,1]))
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 thetahat
## 1  0  0  1  0  1  0  0  0  0   1   0   0   0   0   1 -1.30373
## 2  0  1  1  1  0  0  1  0  0   1   1   1   0   0   1 -0.13636
## 3  0  1  1  0  0  1  1  1  1   1   1   0   1   0   1  0.45816
## 4  1  1  1  0  0  1  1  1  1   1   1   0   1   1   1  1.08973
## 5  0  1  1  0  0  1  1  1  1   1   1   0   0   0   1  0.15786
## 6  0  1  0  0  1  1  1  0  0   1   1   0   0   0   0 -0.71640
```

Create an ordinal variable of the estimates of ability split into quintiles

```
(quintiles <- quantile(theta1PL[,1], probs = seq(from = 0.2, to = 1.0, by = 0.2)))
```

```
##      20%      40%      60%      80%     100%
## -0.71640 -0.13636  0.15786  0.76743  1.80146
```

```r
x <- cut(mydf$thetahat, c(-Inf, quintiles))
summary(x)
```

```
##   (-Inf,-0.716] (-0.716,-0.136]  (-0.136,0.158]   (0.158,0.767]
##              98              85              88              92
##     (0.767,1.8]
##              87
```

```r
mydf <- data.frame(mytest,
                   thetahat = theta1PL[,1],
                   group = as.numeric(x))
head(mydf)
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 thetahat group
## 1  0  0  1  0  1  0  0  0  0   1   0   0   0   0   1 -1.30373     1
## 2  0  1  1  1  0  0  1  0  0   1   1   1   0   0   1 -0.13636     3
## 3  0  1  1  0  0  1  1  1  1   1   1   0   1   0   1  0.45816     4
## 4  1  1  1  0  0  1  1  1  1   1   1   0   1   1   1  1.08973     5
## 5  0  1  1  0  0  1  1  1  1   1   1   0   0   0   1  0.15786     3
## 6  0  1  0  0  1  1  1  0  0   1   1   0   0   0   0 -0.71640     1
```

```r
# Create a data frame of item difficulty estimates
delta <- data.frame(item = c(paste("V", 1:numitems, sep = "")),
                    delta = est1PL$est[,2])

# order item difficulties in an ascenting order
delta[order(delta$delta, decreasing = FALSE), ]
```

```
##     item     delta
## V10  V10 -3.107648
## V11  V11 -1.976310
## V2    V2 -1.956884
## V3    V3 -1.626909
## V7    V7 -1.595678
## V15  V15 -1.068847
## V8    V8 -0.794759
```

```
## V6    V6 -0.531918
## V5    V5  0.091406
## V9    V9  0.129619
## V13  V13  0.638424
## V14  V14  1.290031
## V12  V12  1.381420
## V4    V4  1.816504
## V1    V1  1.904672
```

```r
# Put item difficulty, person location estimates and ability group together in a data fra
mydfsort <- data.frame(item = delta$item,
                       delta = delta$delta,
                       theta = mydf$thetahat,
                       grp = mydf$group)
head(mydfsort)
```

```
##   item     delta    theta grp
## 1   V1  1.904672 -1.30373   1
## 2   V2 -1.956884 -0.13636   3
## 3   V3 -1.626909  0.45816   4
## 4   V4  1.816504  1.08973   5
## 5   V5  0.091406  0.15786   3
## 6   V6 -0.531918 -0.71640   1
```

```r
mean.estimates <- data.frame(Q1 = sapply(mydfsort[mydfsort$grp == 1,
                                                  c("delta", "theta")], mean),
                             Q2 = sapply(mydfsort[mydfsort$grp == 2,
                                                  c("delta", "theta")], mean),
                             Q3 = sapply(mydfsort[mydfsort$grp == 3,
                                                  c("delta", "theta")], mean),
                             Q4 = sapply(mydfsort[mydfsort$grp == 4,
                                                  c("delta", "theta")], mean),
                             Q5 = sapply(mydfsort[mydfsort$grp == 5,
                                                  c("delta", "theta")], mean))

mean.estimates
```

```
##              Q1        Q2        Q3       Q4       Q5
## delta -0.35357 -0.062349 -0.253789 -0.63037 -0.48194
## theta -1.23991 -0.351627  0.064248  0.51886  1.12668
```

Function: Produces a Sunflower Scatter Plot for identification of guessing and carelessness

```
sunflower.plot <-
  function(x, y){
    ## add extra space to right margin of plot within frame
    par(mar = c(4, 4.5, 5.5, 0) + 0.1)

    # Sunflower Scatter Plot
    # Multiple points are plotted as sunflowers with multiple leaves (petals)
    # such that overplotting is visualized instead of accidental and invisible.
    sunflowerplot(x, y,
                  xlim = c(-3, 3),
                  ylim = c(-0.5, 1.5),
                  ylab = "",
                  xlab = "",
                  axes = FALSE)

    box()

    # x-axis
    axis(1, pretty(c(-3, 3), 7))
    axis(3, pretty(c(-3, 3), 7))
    mtext("Ability",
          side = 1,
          line = 2.5)

    # y-axis
    axis(2, at = c(0, 1),
         labels = c("incorrect", "correct"),
         col.axis = "tomato",
         col.ticks = "tomato")

    mtext("Responses",
```

```
        side = 2,
        line = 2.5)

  # Simple Regression Line
  # a, b: the intercept and slope, single values.
  abline(a = lm(y ~ x)[1], # intercept
         b = lm(y ~ x)[2], # slope
         col = "dodgerblue",
         lwd = 3)

  # Quadrant tracing lines
  abline(a = .5, # intercept
         b = -.5, # slope
         h = .5, # horizontal divider
         col = "green",
         lwd = 3,
         lty = "dotted")

  # Quadrant labels
  points(c(1, -2.5, -1, 2.5),
         c(1.25, 1.25, -0.25, -0.25),
         pch = 21,
         bg = "white",
         cex = 4)
  text(c(1, -2.5, -1, 2.5),
       c(1.25, 1.25, -0.25, -0.25),
       c("I", "II", "III", "IV"))

  # gridlines
  grid(nx = NULL, ny = NULL,
       col = "lightgray",
       lty = 3,
       lwd = .5,
       equilogs = TRUE)
}
```

```r
dump("sunflower.plot", file = "sunflower.plot.R")
```

1.2.1 Most Difficult Items and evidence for guessing

If low performing students ($\Theta < -1$) give a correct response to most difficult questions, there is evidence for guessing.

```r
# Most difficult items
head(delta[order(delta$delta, decreasing = TRUE), ], 4)
```

```
##     item  delta
## V1    V1 1.9047
## V4    V4 1.8165
## V12  V12 1.3814
## V14  V14 1.2900
```

Item 1

There aren't any sunflowers in the second quadrant (low performing students give correct response), so there isn't any evidence for guessing.

```r
lm(mydf$V1 ~ mydf$thetahat)
```

```
##
## Call:
## lm(formula = mydf$V1 ~ mydf$thetahat)
##
## Coefficients:
##   (Intercept)  mydf$thetahat
##         0.138          0.189
```

```r
sunflower.plot(mydf$thetahat, mydf$V1)
title(main = "Item 1", col.main = "dodgerblue")
```

Item 4

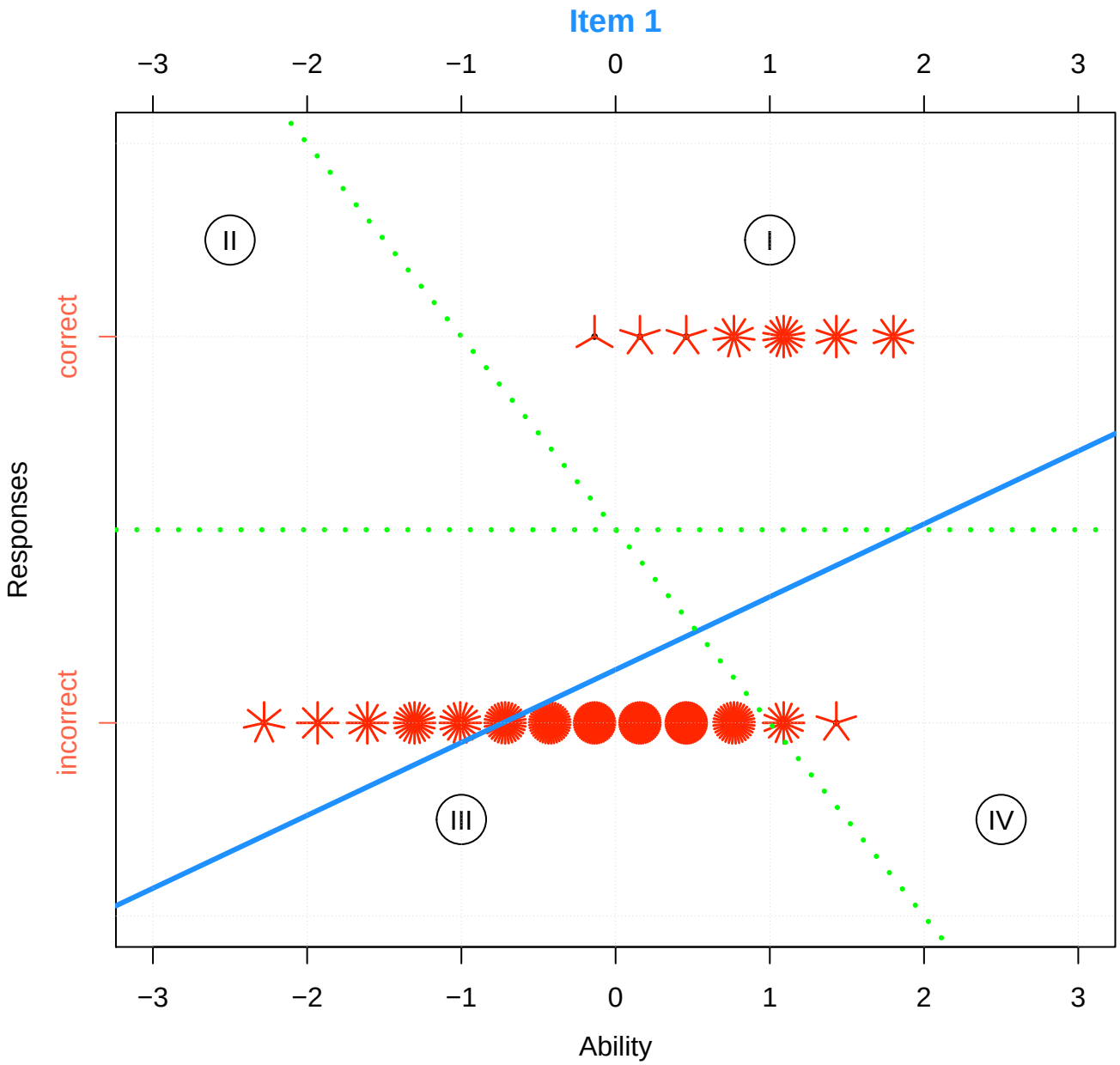

Item 1
−3
−2
−1
0
1
2
3
II
I
III
IV
correct
incorrect
Responses
Ability

Figure 8:

```r
lm(mydf$V4 ~ mydf$thetahat)
```

```
##
## Call:
## lm(formula = mydf$V4 ~ mydf$thetahat)
##
## Coefficients:
##   (Intercept)  mydf$thetahat
##         0.149          0.163
```

```r
sunflower.plot(mydf$thetahat, mydf$V4)
title(main = "Item 4", col.main = "dodgerblue")
```

Item 14

```r
lm(mydf$V14 ~ mydf$thetahat)
```

```
##
## Call:
## lm(formula = mydf$V14 ~ mydf$thetahat)
##
## Coefficients:
##   (Intercept)  mydf$thetahat
##         0.229          0.275
```

```r
sunflower.plot(mydf$thetahat, mydf$V14)
title(main = "Item 14", col.main = "dodgerblue")
```

Item 12

```r
lm(mydf$V12 ~ mydf$thetahat)
```

```
##
## Call:
## lm(formula = mydf$V12 ~ mydf$thetahat)
##
```

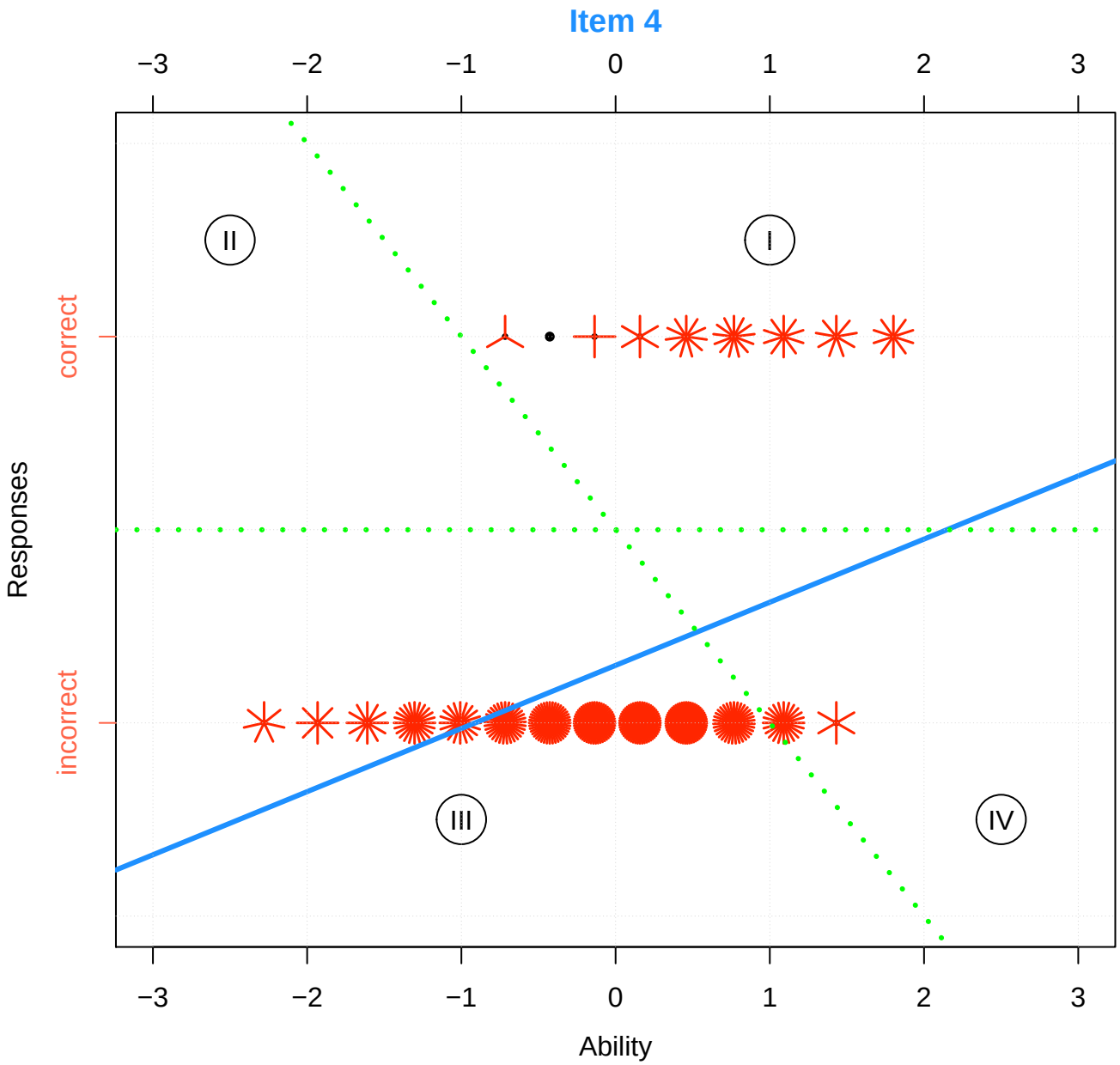

Item 4
−3
−2
−1
0
1
2
3
II
I
III
IV
correct
incorrect
Responses
Ability

Figure 9:

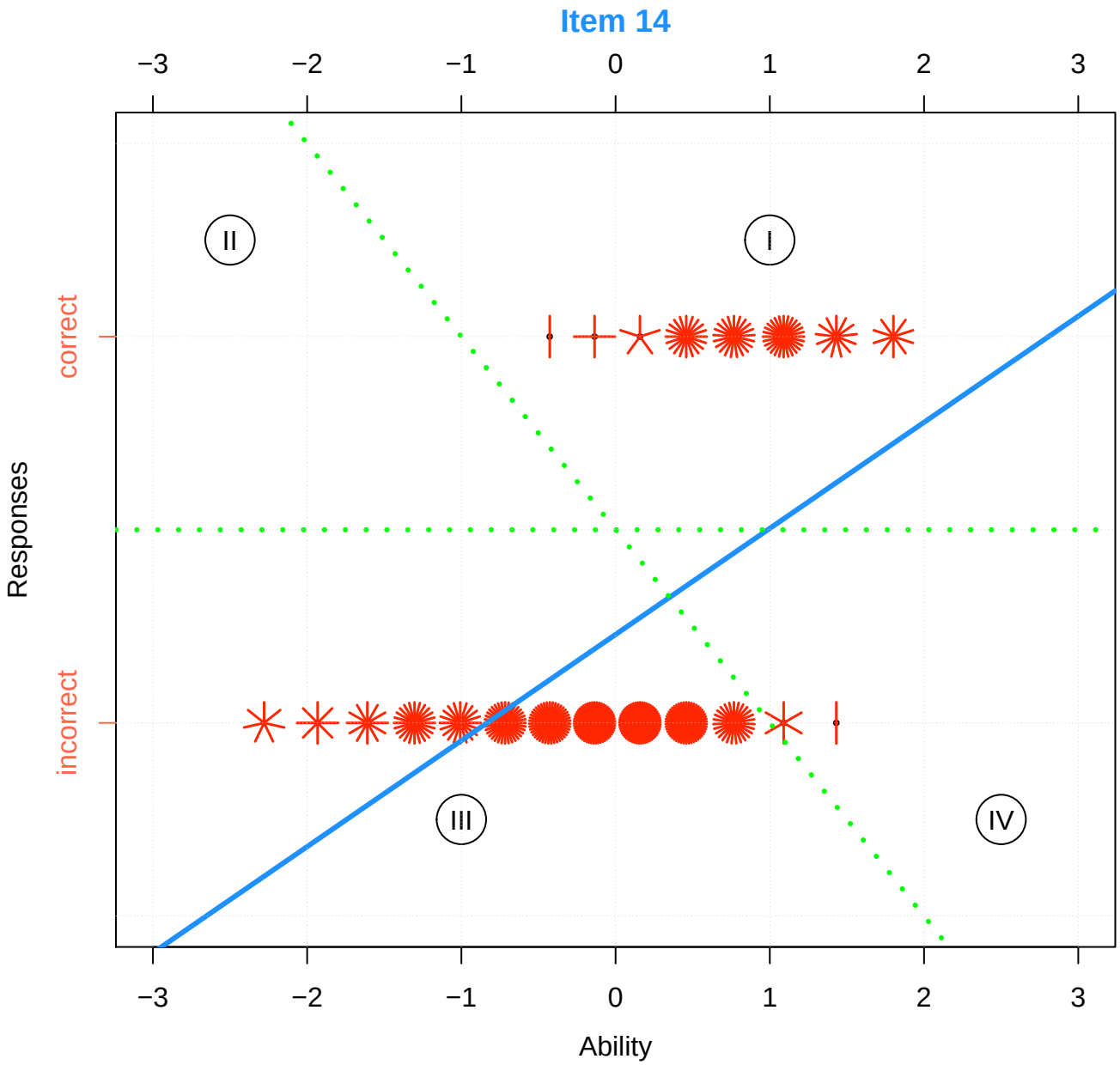

Item 14
−3
−2
−1
0
1
2
3
II
I
III
IV
correct
incorrect
Responses
Ability

Figure 10:

```
## Coefficients:
##   (Intercept)  mydf$thetahat
##         0.213          0.194
```

```r
sunflower.plot(mydf$thetahat, mydf$V12)
title(main = "Item 12", col.main = "dodgerblue")
```

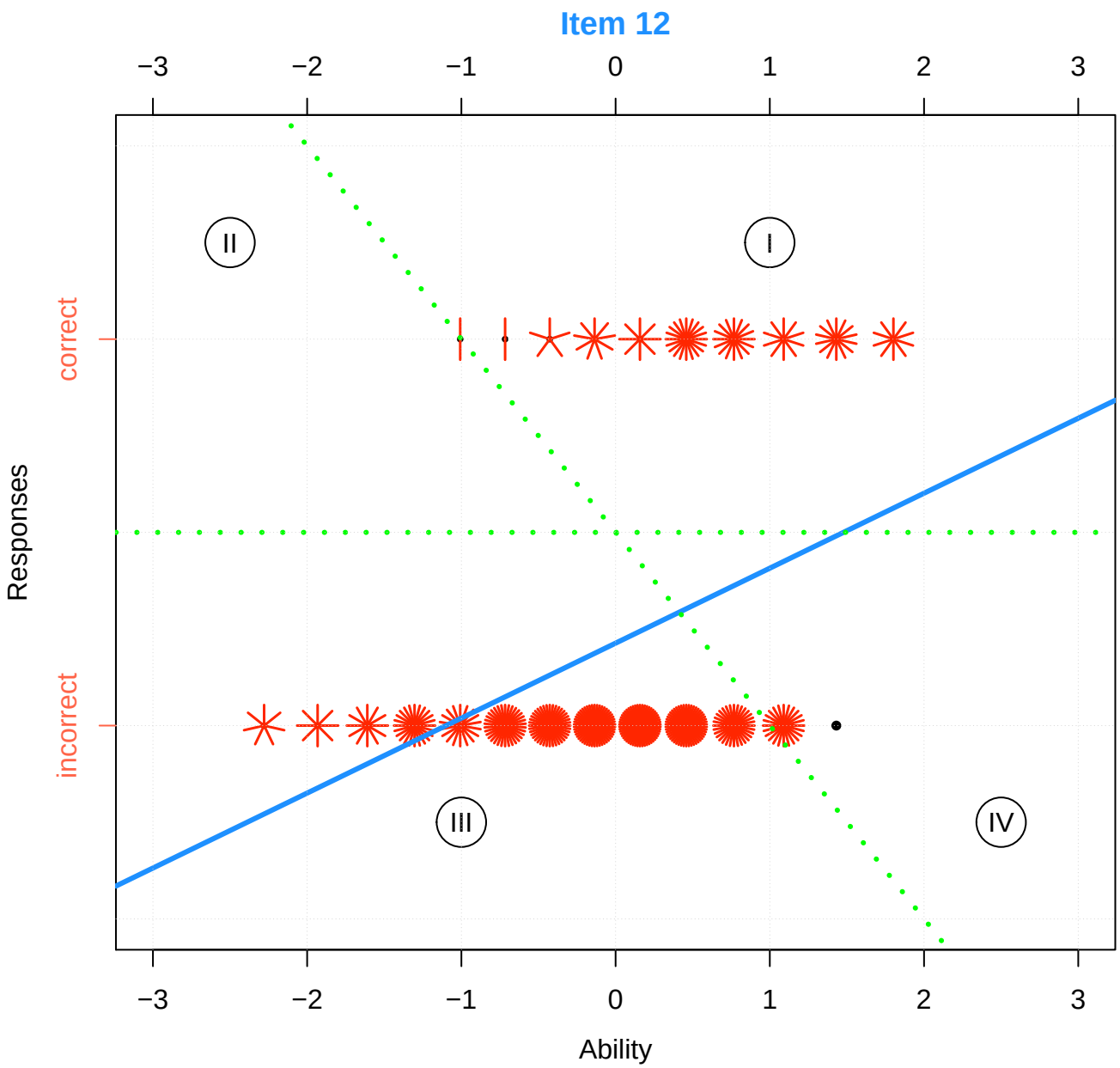


Figure 11:

1.2.2 Easiest Items and Evidence for Carelessness

If high performing students ($\Theta > 1$) give an incorrect response to easiest questions, there is evidence for carelessness.

```
# Easiest items
head(delta[order(delta$delta, decreasing = FALSE), ], 4)
```

```
##     item   delta
## V10  V10 -3.1076
## V11  V11 -1.9763
## V2    V2 -1.9569
## V3    V3 -1.6269
```

Item 10

```
lm(mydf$V10 ~ mydf$thetahat)
```

```
##
## Call:
## lm(formula = mydf$V10 ~ mydf$thetahat)
##
## Coefficients:
##   (Intercept)  mydf$thetahat
##         0.956          0.089
```

```
sunflower.plot(mydf$thetahat, mydf$V10)
title(main = "Item 10", col.main = "dodgerblue")
```

Item 11

```
lm(mydf$V11 ~ mydf$thetahat)
```

```
##
## Call:
## lm(formula = mydf$V11 ~ mydf$thetahat)
##
```

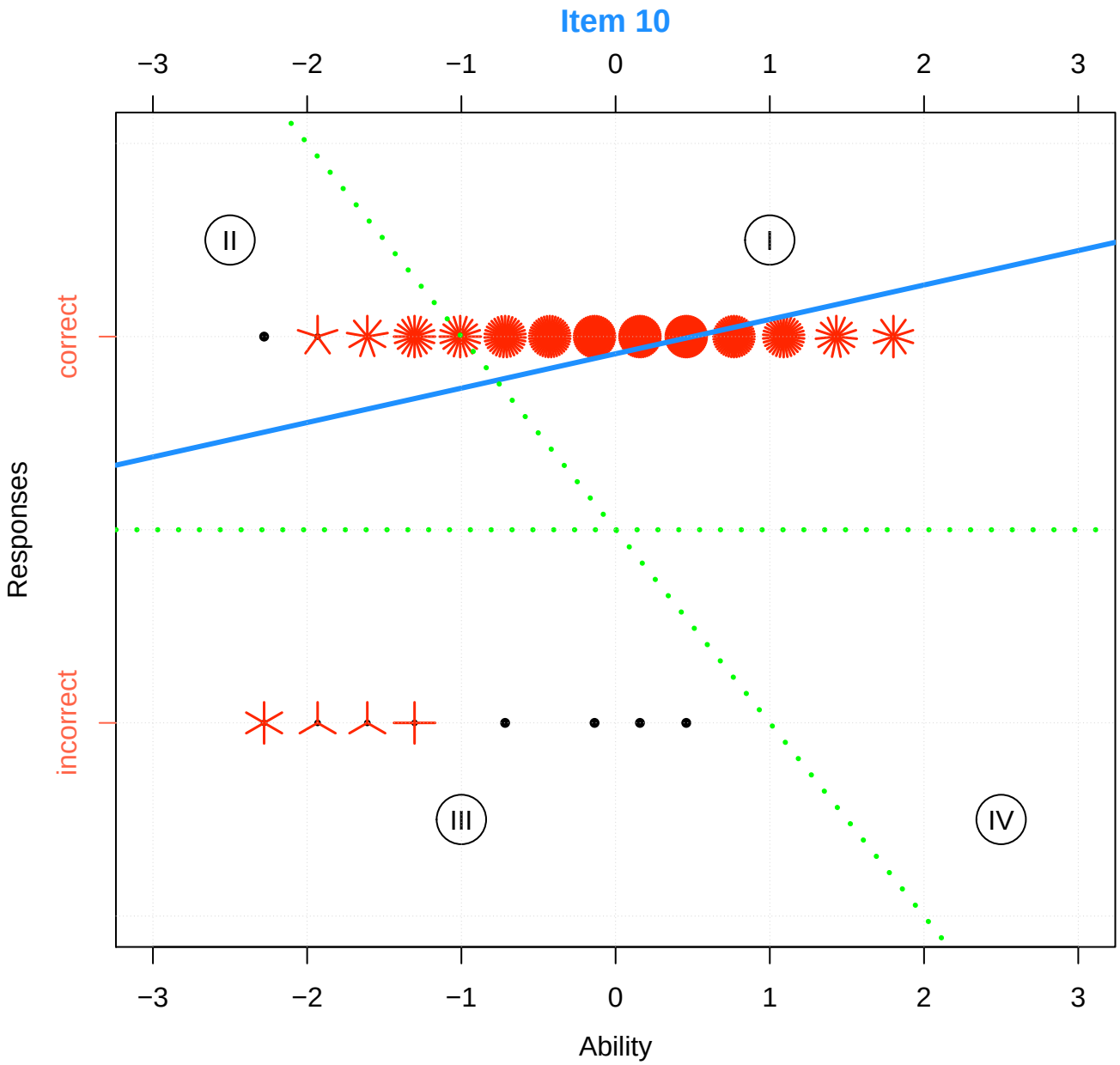

Item 10
−3
−2
−1
0
1
2
3
II
I
correct
Responses
incorrect
III
IV
Ability

Figure 12:

```
## Coefficients:
##   (Intercept)  mydf$thetahat
##         0.869          0.215
```

```r
sunflower.plot(mydf$thetahat, mydf$V11)
title(main = "Item 11", col.main = "dodgerblue")
```

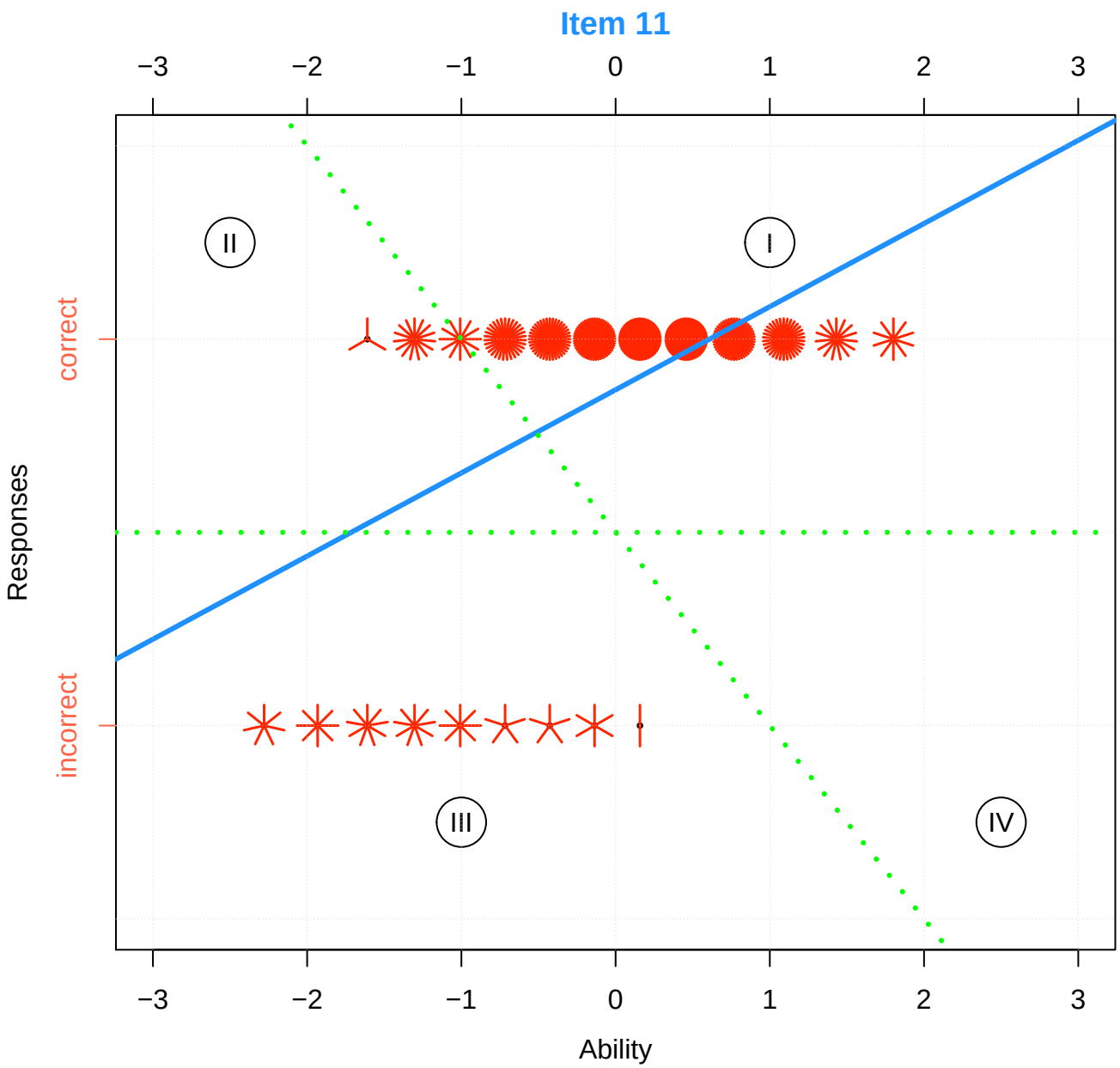


Figure 13:

Item 2

```r
lm(mydf$V2 ~ mydf$thetahat)
```

```
##
## Call:
## lm(formula = mydf$V2 ~ mydf$thetahat)
##
## Coefficients:
##   (Intercept)  mydf$thetahat
##         0.867          0.209
```

```r
sunflower.plot(mydf$thetahat, mydf$V2)
title(main = "Item 2", col.main = "dodgerblue")
```

Item 3

```r
lm(mydf$V3 ~ mydf$thetahat)
```

```
##
## Call:
## lm(formula = mydf$V3 ~ mydf$thetahat)
##
## Coefficients:
##   (Intercept)  mydf$thetahat
##         0.824          0.260
```

```r
sunflower.plot(mydf$thetahat, mydf$V3)
title(main = "Item 3", col.main = "dodgerblue")
```

1.2.3 Some Other Interesting Cases

An item with average difficulty, V9

```r
delta[9,]
```

```
##    item   delta
## V9   V9 0.12962
```

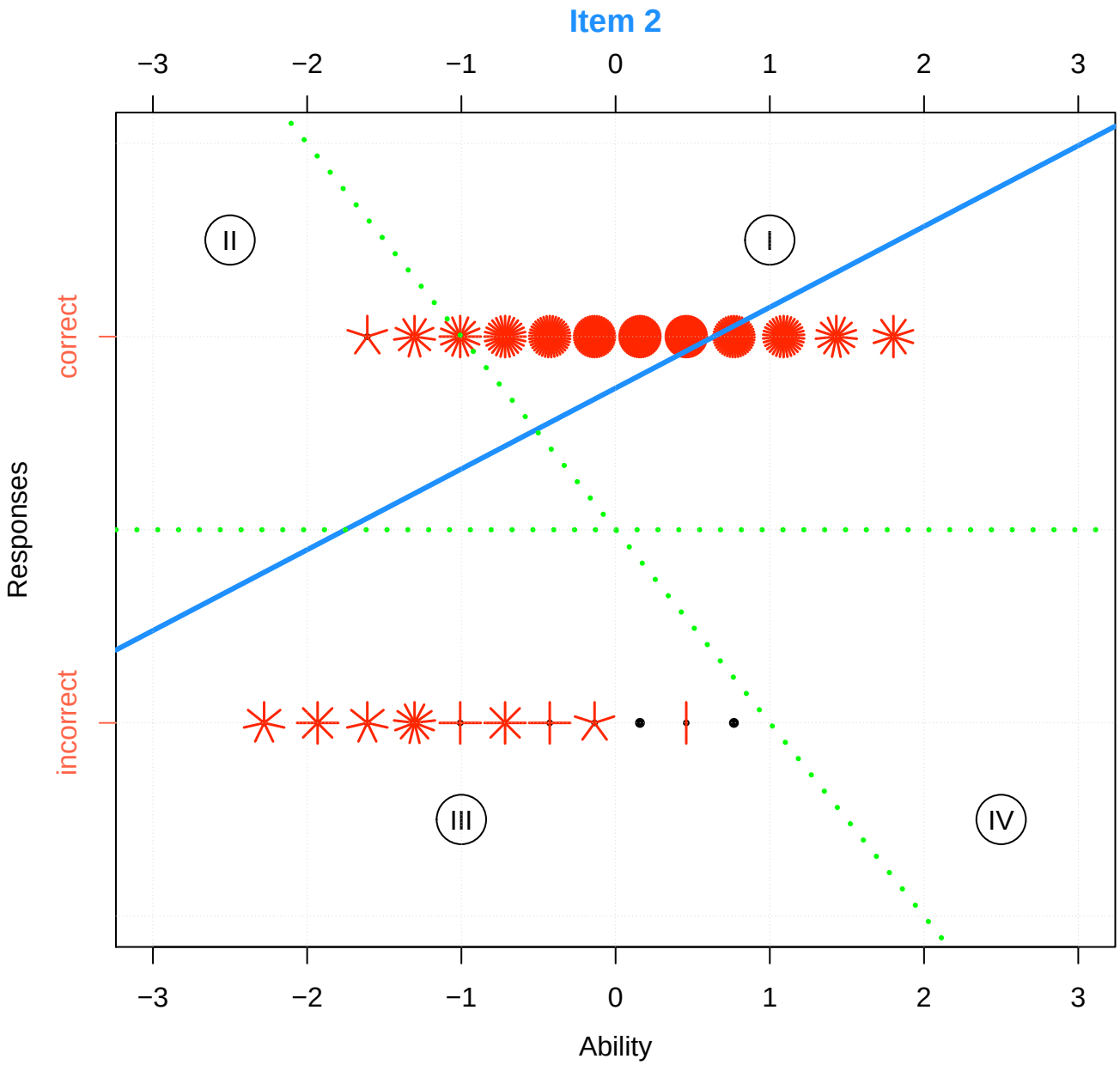

Item 2
−3
−2
−1
0
1
2
3
II
I
correct
Responses
incorrect
III
IV
Ability

Figure 14:

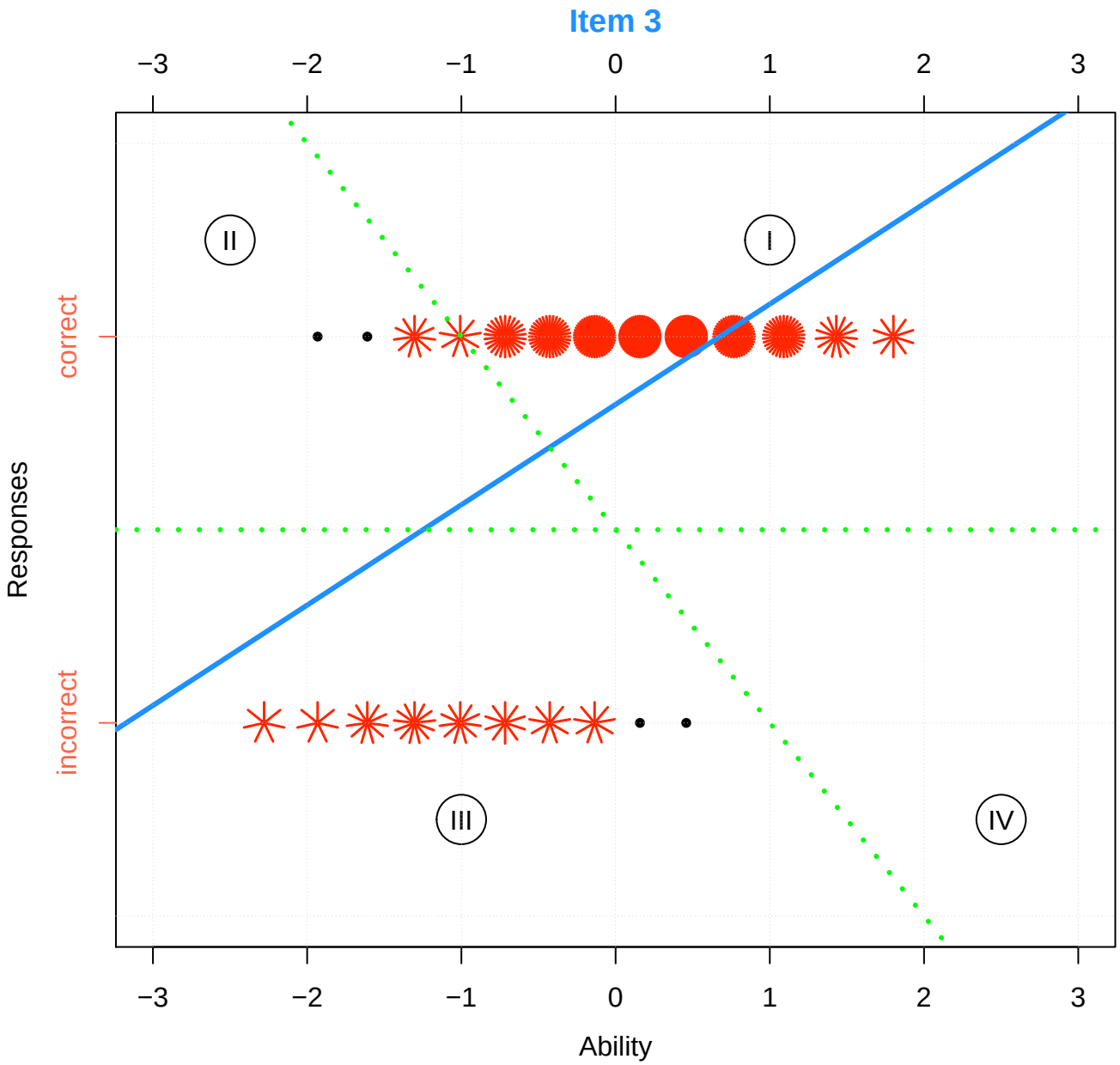

Item 3
−3
−2
−1
0
1
2
3
II
I
III
IV
correct
incorrect
Responses
Ability

Figure 15:

```
sunflower.plot(mydf$thetahat, mydf$V9)
title(main = "Item 9", col.main = "dodgerblue")
```

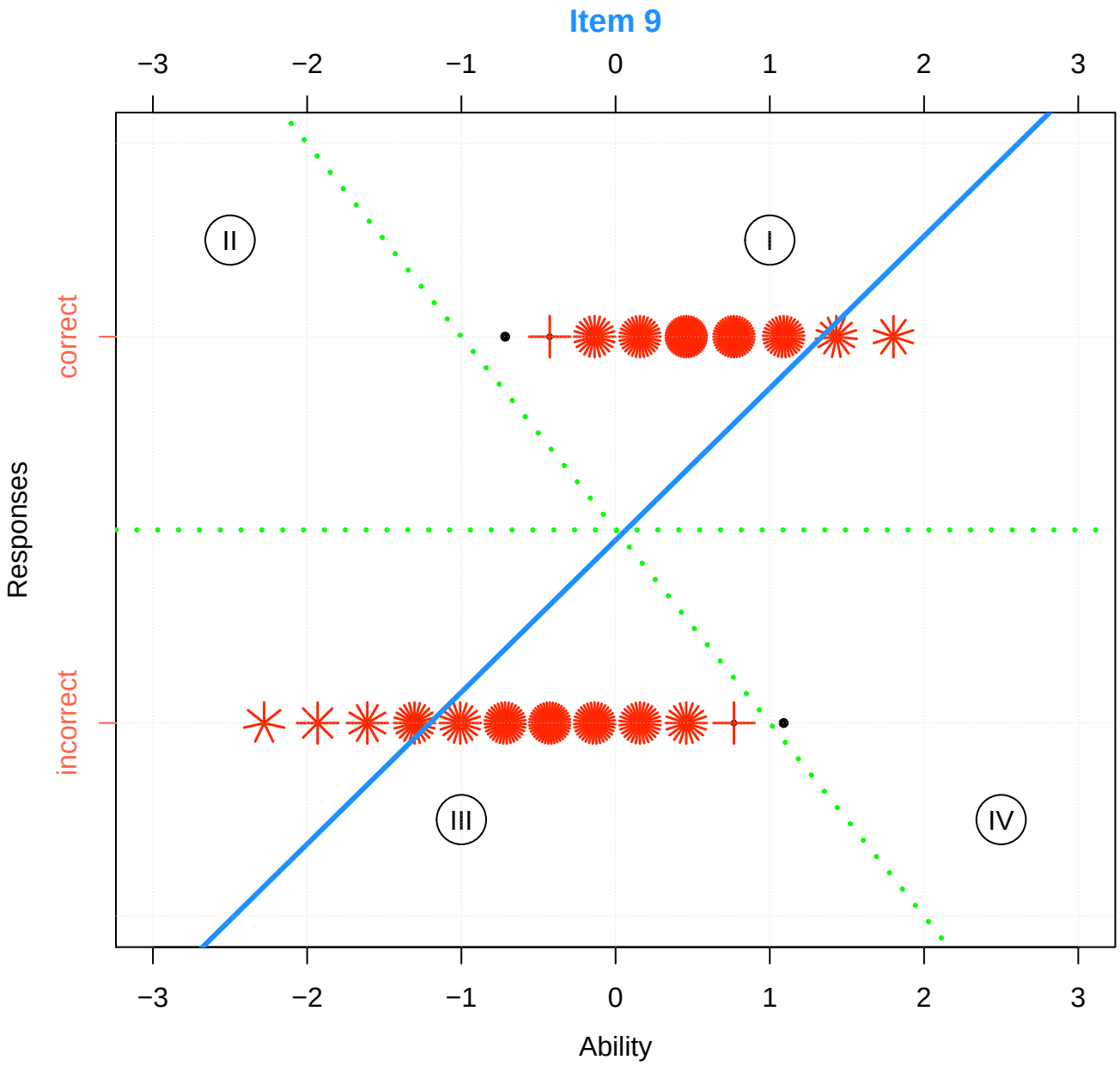


Figure 16:

A miscoded item, V5

Most of the low ability students gave a correct response (second quadrant) indicating guessing, while almost all of the high ability students gave an incorrect response (fourth quadrant). Also the linear regression line has a negative slope indicating an item that was not performing the way it was intended.

```r
delta[5,]
```

```
##    item    delta
## V5   V5 0.091406
```

```r
## add extra space to right margin of plot within frame
par(mar = c(4, 4.5, 2, 0) + 0.1)

# Sunflower Scatter Plot
# Multiple points are plotted as sunflowers with multiple leaves (petals)
# such that overplotting is visualized instead of accidental and invisible.
sunflowerplot(mydf$thetahat, mydf$V5,
              xlim = c(-3, 3),
              ylim = c(-0.5, 1.5),
              ylab = "",
              xlab = "",
              axes = FALSE,
              main = "Ability vs. Responses for Item 5")

box()

# x-axis
axis(1, pretty(c(-3, 3), 7))

mtext("Ability",
      side = 1,
      col = "black",
      line = 2.5)

# y-axis
axis(2, at = c(0, 1),
     labels = c("incorrect", "correct"),
     col.axis = "tomato",
     col.ticks = "tomato")

mtext("Responses",
```

```
      side = 2,
      line = 2.5)

# Simple Regression Line
# a, b: the intercept and slope, single values.
abline(a = lm(mydf$V5 ~ mydf$thetahat)[1], # intercept
       b = lm(mydf$V5 ~ mydf$thetahat)[2], # slope
       col = "dodgerblue",
       lwd = 3)

# Quadrant tracing lines
abline(a = .5, # intercept
       b = .5, # slope
       h = .5, # horizontal divider
       col = "green",
       lwd = 3,
       lty = "dotted")

# Quadrant labels
points(c(2.5, -1, -2.5, 1),
       c(1.25, 1.25, -0.25, -0.25),
       pch = 21,
       bg = "white",
       cex = 4)
text(c(2.5, -1, -2.5, 1),
     c(1.25, 1.25, -0.25, -0.25),
     c("I", "II", "III", "IV"))
```

1.3 Test Equating, Scaling, and Linking

Item exposure can compromise the security of tests where administrations across more than one occasion and more than one examinee group can lead to overexposure of items. It can be limited by using alternate test forms, however multiple forms lead to multiple score scales that measure the construct of interest at differing levels of difficulty. These differences in difficulty across alternate forms of a test can be adjusted by equating to produce comparable score scales.

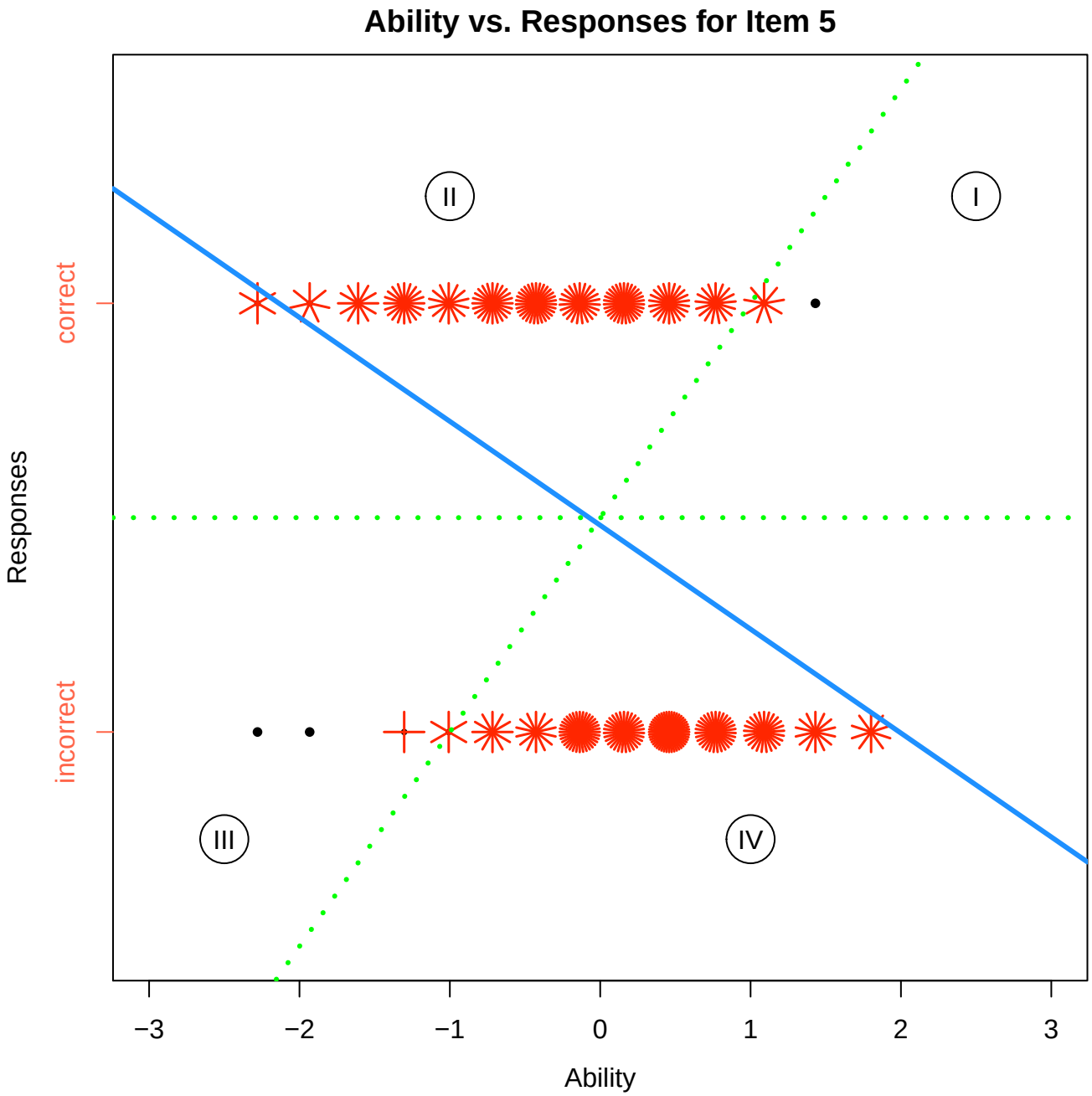

Ability vs. Responses for Item 5
II
I
III
IV
correct
incorrect
Responses
−3
−2
−1
0
1
2
3
Ability

Figure 17:

Raw scores often are transformed to scale scores to enhance the interpretability so that a scale score has the same meaning regardless of the test form administered or the group of examinees tested (Kolen M. J. and Brennan 2013, 4). Score scales typically are established using a single test form, which is maintained through an equating process that places raw scores from subsequent forms on the established score scale.

Although their purposes are different, similar statistical procedures often are used in linking and equating. Equating is used to adjust scores on test forms that are built to be as similar as possible in content and statistical characteristics, whereas tests that are purposefully built to be different are linked (Kolen M. J. and Brennan 2013, 3).

Create two data sets with unique and common items

```
# CONSTANTS
npersons <- 1000
numitems <- 50
numanchoritems <- 10
```

Population P and Form X

```
# Population P true theta (ability)
set.seed(1111)
P <- rnorm(npersons, mean = 0, sd = 1)

summary(P)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.9500 -0.6710 -0.0315 -0.0263  0.6450  2.9700
```

```
# Form X items
set.seed(55555555)
X <- cbind(a = rnorm(numitems, .5, .15),
           b = rnorm(numitems, 0, 1),
           c = runif(numitems, 0, 0))

head(X)
```

```
##            a       b c
```

```
## [1,] 0.40491 0.93542 0
## [2,] 0.69813 0.14567 0
## [3,] 0.48237 1.30953 0
## [4,] 0.21984 1.13276 0
## [5,] 0.41584 1.40929 0
## [6,] 0.69198 0.24308 0
```

```r
summary(X)
```

```
##        a               b                 c
##  Min.   :0.152   Min.   :-2.920   Min.   :0
##  1st Qu.:0.376   1st Qu.:-0.164   1st Qu.:0
##  Median :0.524   Median : 0.257   Median :0
##  Mean   :0.512   Mean   : 0.234   Mean   :0
##  3rd Qu.:0.662   3rd Qu.: 0.935   3rd Qu.:0
##  Max.   :0.851   Max.   : 1.967   Max.   :0
```

```r
# Simulate responses for population P and Form X
set.seed(11111)
XP <- data.frame(irtoys::sim(ip = X, P))
names(XP) <- c(paste("X", 1:numitems, sep = ""))
str(XP)
```

```
## 'data.frame':    1000 obs. of  50 variables:
##  $ X1 : num  0 0 0 0 0 0 1 0 1 0 ...
##  $ X2 : num  1 1 0 1 0 0 1 1 1 1 ...
##  $ X3 : num  0 1 1 0 0 0 0 1 0 0 ...
##  $ X4 : num  1 1 0 1 0 0 1 0 0 0 ...
##  $ X5 : num  1 0 0 0 1 0 1 0 1 1 ...
##  $ X6 : num  1 1 1 0 0 0 0 1 1 1 ...
##  $ X7 : num  0 1 0 0 0 1 1 0 1 0 ...
##  $ X8 : num  0 1 1 1 1 0 1 1 1 0 ...
##  $ X9 : num  1 1 0 1 0 0 0 0 1 0 ...
##  $ X10: num  0 1 0 0 0 0 0 0 1 1 ...
##  $ X11: num  1 1 1 1 1 0 1 1 1 1 ...
##  $ X12: num  0 1 1 1 0 0 1 1 0 1 ...
```

```
##  $ X13: num  1 1 1 1 0 0 1 0 1 1 ...
##  $ X14: num  0 0 1 1 1 0 1 0 1 0 ...
##  $ X15: num  1 1 1 0 0 0 1 1 0 1 ...
##  $ X16: num  0 0 0 0 1 0 1 1 1 1 ...
##  $ X17: num  1 1 1 1 0 0 0 1 1 0 ...
##  $ X18: num  0 1 0 1 0 0 0 0 0 0 ...
##  $ X19: num  1 0 1 0 1 0 1 1 1 1 ...
##  $ X20: num  1 1 0 0 1 0 1 1 1 1 ...
##  $ X21: num  0 1 1 0 1 0 1 1 0 1 ...
##  $ X22: num  0 1 1 1 0 0 1 0 1 0 ...
##  $ X23: num  1 0 1 0 1 0 1 0 1 1 ...
##  $ X24: num  0 1 1 1 0 0 0 1 1 1 ...
##  $ X25: num  1 0 0 1 0 0 0 1 1 1 ...
##  $ X26: num  1 1 0 1 0 0 1 1 1 1 ...
##  $ X27: num  1 0 1 0 1 0 1 1 0 1 ...
##  $ X28: num  0 1 0 1 1 1 1 1 0 0 ...
##  $ X29: num  0 1 1 0 1 1 1 1 1 1 ...
##  $ X30: num  0 1 1 1 0 0 1 1 0 1 ...
##  $ X31: num  1 1 1 1 1 0 1 1 1 1 ...
##  $ X32: num  0 0 0 0 0 0 1 0 1 0 ...
##  $ X33: num  1 0 0 1 0 0 1 1 1 0 ...
##  $ X34: num  0 1 0 0 1 1 1 0 0 0 ...
##  $ X35: num  0 1 0 1 1 1 0 1 1 1 ...
##  $ X36: num  1 1 0 1 1 1 1 1 1 1 ...
##  $ X37: num  0 1 1 0 1 0 0 1 1 0 ...
##  $ X38: num  0 1 1 1 0 0 1 1 1 1 ...
##  $ X39: num  0 1 0 1 1 1 1 1 1 1 ...
##  $ X40: num  1 0 0 0 1 0 1 1 1 1 ...
##  $ X41: num  1 1 0 0 1 1 0 1 0 0 ...
##  $ X42: num  0 0 0 0 1 1 1 1 0 1 ...
##  $ X43: num  1 1 1 1 1 0 1 1 1 1 ...
##  $ X44: num  1 1 1 1 0 1 1 1 1 1 ...
##  $ X45: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ X46: num  1 1 0 1 1 0 0 1 1 1 ...
##  $ X47: num  0 0 1 1 1 0 0 1 1 1 ...
##  $ X48: num  0 0 0 0 0 0 1 0 1 1 ...
```

```
##  $ X49: num  1 1 1 0 1 0 0 1 1 0 ...
##  $ X50: num  0 0 0 1 1 0 0 0 0 0 ...
```

Population Q and Form Y

```r
# Population Q true theta (ability)
set.seed(333333)
Q <- rnorm(npersons, mean = 0, sd = 1)
summary(Q)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.8200 -0.6930 -0.0663 -0.0171  0.6700  3.4100
```

```r
# Form Y items
set.seed(2222222)
Y <- cbind(a = rnorm(numitems, .5, .15),
           b = rnorm(numitems, 0, 1),
           c = runif(numitems, 0, 0))

head(Y)
```

```
##            a        b c
## [1,] 0.46168  0.26095 0
## [2,] 0.50053 -1.10724 0
## [3,] 0.53192  2.11435 0
## [4,] 0.64322  0.06710 0
## [5,] 0.73759  0.27723 0
## [6,] 0.41536 -0.45655 0
```

```r
summary(Y)
```

```
##        a               b                 c
##  Min.   :0.108   Min.   :-2.675   Min.   :0
##  1st Qu.:0.371   1st Qu.:-0.468   1st Qu.:0
##  Median :0.489   Median : 0.240   Median :0
##  Mean   :0.492   Mean   : 0.113   Mean   :0
##  3rd Qu.:0.632   3rd Qu.: 0.732   3rd Qu.:0
##  Max.   :0.810   Max.   : 2.209   Max.   :0
```

```r
# Simulate responses for population Q and Form Y
set.seed(22222)
YQ <- data.frame(irtoys::sim(ip = Y, Q))
names(YQ) <- c(paste("Y", 1:numitems, sep = ""))
str(YQ)
```

```
## 'data.frame':    1000 obs. of  50 variables:
##  $ Y1 : num  1 0 0 0 1 1 1 1 1 1 ...
##  $ Y2 : num  0 0 1 0 0 1 1 0 1 1 ...
##  $ Y3 : num  0 0 0 0 0 1 1 0 0 1 ...
##  $ Y4 : num  0 1 0 0 0 1 1 0 1 1 ...
##  $ Y5 : num  0 0 1 0 0 0 1 0 1 1 ...
##  $ Y6 : num  0 1 0 1 1 1 1 1 1 1 ...
##  $ Y7 : num  0 0 0 0 1 0 1 1 1 0 ...
##  $ Y8 : num  0 1 0 0 0 0 1 0 0 1 ...
##  $ Y9 : num  1 1 1 1 1 1 0 1 1 0 ...
##  $ Y10: num  0 1 0 0 1 1 1 1 0 1 ...
##  $ Y11: num  0 0 1 1 1 1 1 0 1 0 ...
##  $ Y12: num  1 1 1 1 1 0 1 1 1 1 ...
##  $ Y13: num  1 1 0 1 1 1 1 1 0 1 ...
##  $ Y14: num  0 1 0 0 0 0 1 1 1 0 ...
##  $ Y15: num  1 1 0 0 0 1 0 1 0 0 ...
##  $ Y16: num  0 1 1 1 1 1 1 0 0 1 ...
##  $ Y17: num  1 0 1 1 1 1 1 1 0 1 ...
##  $ Y18: num  1 1 0 0 0 0 0 0 0 0 ...
##  $ Y19: num  1 1 1 0 1 1 0 0 1 0 ...
##  $ Y20: num  0 0 0 1 1 1 1 0 0 1 ...
##  $ Y21: num  0 1 0 1 0 1 0 1 1 1 ...
##  $ Y22: num  0 1 0 0 1 0 1 1 0 0 ...
##  $ Y23: num  0 1 1 1 1 0 1 1 0 1 ...
##  $ Y24: num  0 0 0 0 1 1 1 0 0 1 ...
##  $ Y25: num  0 0 1 0 1 0 0 0 1 1 ...
##  $ Y26: num  1 1 0 1 1 1 1 0 0 1 ...
##  $ Y27: num  1 1 0 1 0 1 1 0 0 0 ...
##  $ Y28: num  1 1 1 0 0 1 0 1 1 0 ...
##  $ Y29: num  0 1 1 1 1 0 1 1 1 1 ...
```

```
##  $ Y30: num  0 1 0 1 0 0 1 0 0 0 ...
##  $ Y31: num  0 1 1 0 0 1 1 0 1 1 ...
##  $ Y32: num  0 1 1 0 1 0 1 0 1 1 ...
##  $ Y33: num  0 1 1 1 1 0 1 1 1 1 ...
##  $ Y34: num  0 1 1 0 0 1 1 0 0 1 ...
##  $ Y35: num  0 1 0 0 1 0 1 1 1 0 ...
##  $ Y36: num  0 1 0 1 1 0 1 1 1 1 ...
##  $ Y37: num  1 1 1 1 0 0 1 0 1 0 ...
##  $ Y38: num  0 0 0 1 1 1 1 0 1 1 ...
##  $ Y39: num  0 1 0 0 1 0 1 0 0 1 ...
##  $ Y40: num  1 0 1 0 0 0 1 0 1 1 ...
##  $ Y41: num  1 1 0 0 1 0 0 1 1 1 ...
##  $ Y42: num  0 0 1 0 1 1 1 1 1 1 ...
##  $ Y43: num  0 1 1 0 1 0 1 0 1 1 ...
##  $ Y44: num  0 1 1 1 1 1 1 0 0 1 ...
##  $ Y45: num  0 1 0 1 1 1 1 1 1 0 ...
##  $ Y46: num  1 1 0 0 1 0 0 0 0 0 ...
##  $ Y47: num  0 1 1 0 1 0 1 0 0 0 ...
##  $ Y48: num  0 1 0 0 0 0 0 1 1 1 ...
##  $ Y49: num  0 0 0 0 1 0 1 0 1 0 ...
##  $ Y50: num  1 1 1 0 0 0 1 0 1 0 ...
```

Anchor Items for both populations, P and Q

```r
# Anchor test items, V
set.seed(333)
V <- cbind(a = rnorm(numanchoritems, .5, .15),
           b = rnorm(numanchoritems, 0, 1),
           c = runif(numanchoritems, 0, 0))

head(V)
```

```
##            a         b c
## [1,] 0.48758 -1.124900 0
## [2,] 0.79020 -0.874304 0
## [3,] 0.19231  0.043862 0
## [4,] 0.54166 -0.583975 0
```

```
## [5,] 0.27111 -0.823299 0
## [6,] 0.45963  0.110593 0
```

```r
summary(V)
```

```
##        a               b                  c
##  Min.   :0.192   Min.   :-1.1249   Min.   :0
##  1st Qu.:0.427   1st Qu.:-0.7635   1st Qu.:0
##  Median :0.515   Median :-0.1683   Median :0
##  Mean   :0.499   Mean   :-0.1845   Mean   :0
##  3rd Qu.:0.585   3rd Qu.: 0.0964   3rd Qu.:0
##  Max.   :0.790   Max.   : 1.3687   Max.   :0
```

```r
# Simulate responses to the anchor items by pupulation P
set.seed(1111)
VP <- data.frame(irtoys::sim(ip = V, P))
names(VP) <- c(paste("V", 1:numanchoritems, sep = ""))
str(VP)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ V1 : num  1 1 0 1 0 0 0 1 1 1 ...
##  $ V2 : num  1 1 1 1 0 0 0 1 1 1 ...
##  $ V3 : num  1 1 0 0 0 0 0 1 1 1 ...
##  $ V4 : num  1 0 1 1 1 0 1 1 1 0 ...
##  $ V5 : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ V6 : num  0 0 1 0 1 0 1 1 0 1 ...
##  $ V7 : num  1 0 0 0 1 0 0 1 0 0 ...
##  $ V8 : num  0 1 1 0 1 1 1 1 0 1 ...
##  $ V9 : num  0 0 1 1 0 0 1 0 1 0 ...
##  $ V10: num  0 1 0 1 0 0 0 0 1 0 ...
```

```r
# Simulate responses to the anchor items by population Q
set.seed(66666)
VQ <- data.frame(irtoys::sim(ip = V, Q))
names(VQ) <- c(paste("V", 1:numanchoritems, sep = ""))
str(VQ)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ V1 : num  1 1 1 0 1 0 1 1 1 1 ...
##  $ V2 : num  0 1 1 0 1 1 1 0 1 1 ...
##  $ V3 : num  1 1 1 0 1 0 1 0 0 1 ...
##  $ V4 : num  1 0 1 0 1 1 1 1 1 1 ...
##  $ V5 : num  1 0 0 0 1 0 0 1 0 0 ...
##  $ V6 : num  0 0 1 0 1 1 1 1 1 1 ...
##  $ V7 : num  0 0 1 0 0 1 0 1 0 0 ...
##  $ V8 : num  0 1 0 0 0 0 1 1 0 1 ...
##  $ V9 : num  0 1 1 1 1 0 1 1 0 1 ...
##  $ V10: num  0 0 1 0 0 1 0 1 1 0 ...
```

```r
# The anchor item scores are included in the total scores
XVP <- data.frame(total = rowSums(cbind(XP, VP)),
                  anchor = rowSums(VP))

head(XVP)
```

```
##   total anchor
## 1    30      5
## 2    40      5
## 3    30      5
## 4    33      5
## 5    31      4
## 6    12      1
```

1.3.1 Scaling

```r
library(equate)
x <- equate::freqtab(XVP)
plot(x)
```

```r
xs <- loglinear(x, degrees = c(4, 1),
  stepup = TRUE, showWarnings = FALSE)
plot(x, xs, lwd = 2)
```

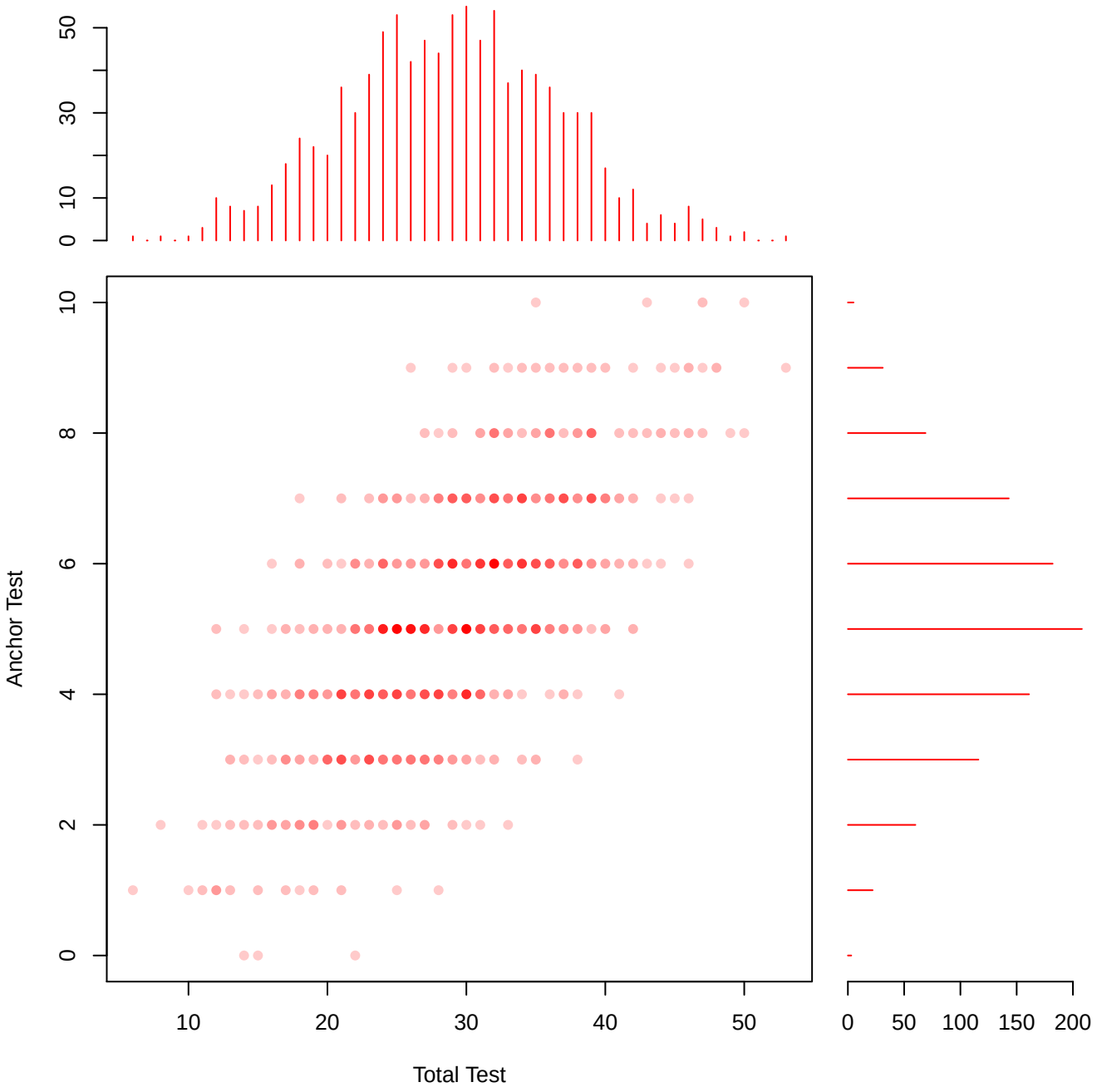

0
10
30
50
Anchor Test
0
2
4
6
8
10
10
20
30
40
50
Total Test
0
50
100
150
200

Figure 18:

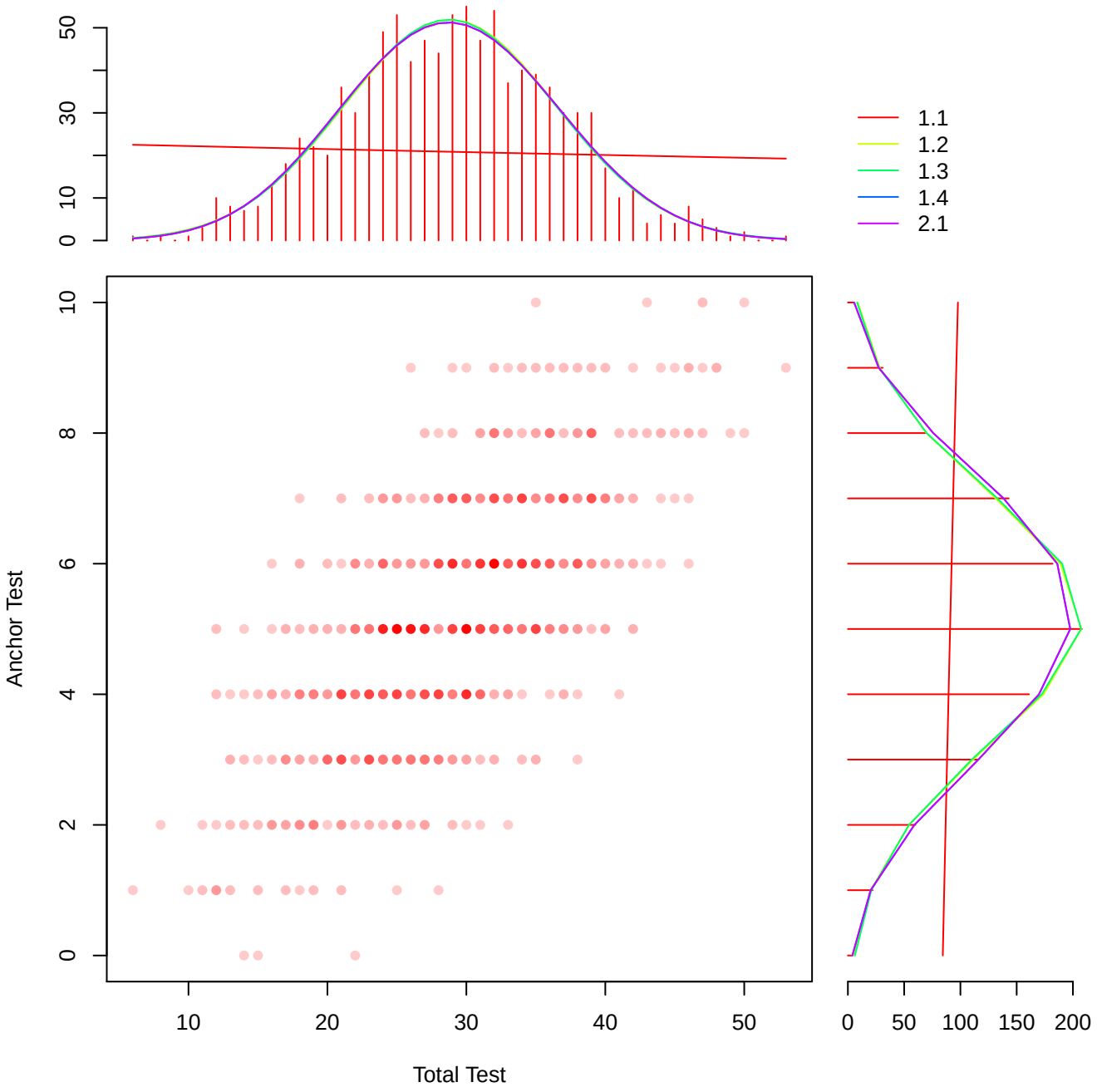

0
10
30
50
1.1
1.2
1.3
1.4
2.1
Anchor Test
0
2
4
6
8
10
10
20
30
40
50
Total Test
0
50
100
150
200

Figure 19:

```r
# The anchor item scores are included in the total scores
YQV <- data.frame(total_score = rowSums(cbind(YQ, VQ)),
                  anchor_score = rowSums(VQ))

head(YQV)
```

```
##   total_score anchor_score
## 1          20            4
## 2          41            5
## 3          31            8
## 4          21            1
## 5          39            7
## 6          30            5
```

```r
y <- equate::freqtab(YQV)

plot(y)
```

```r
ys <- loglinear(y, stepup = TRUE)
plot(y, ys, lwd = 2)
```

1.3.2 Equating and Linking

Test Scores Under a Non-Equivalent-groups Anchor Test (NEAT) Design

The NEAT design is a data collection design involving two populations of test-takers, P and Q, and a sample of examinees from each. The sample from P takes test X, the sample from Q takes test Y, and both samples take an anchor test, V, which is used to link X and Y.

Vertical equating is the process of equating tests administered to groups of students with different abilities, such as students in different grades (years of schooling).

Horizontal equating is the equating of tests administered to groups with similar abilities. Different tests are used to avoid practice effects.

In mean equating, scores on the two forms that are an equal (signed) distance away from their respective means are set equal:

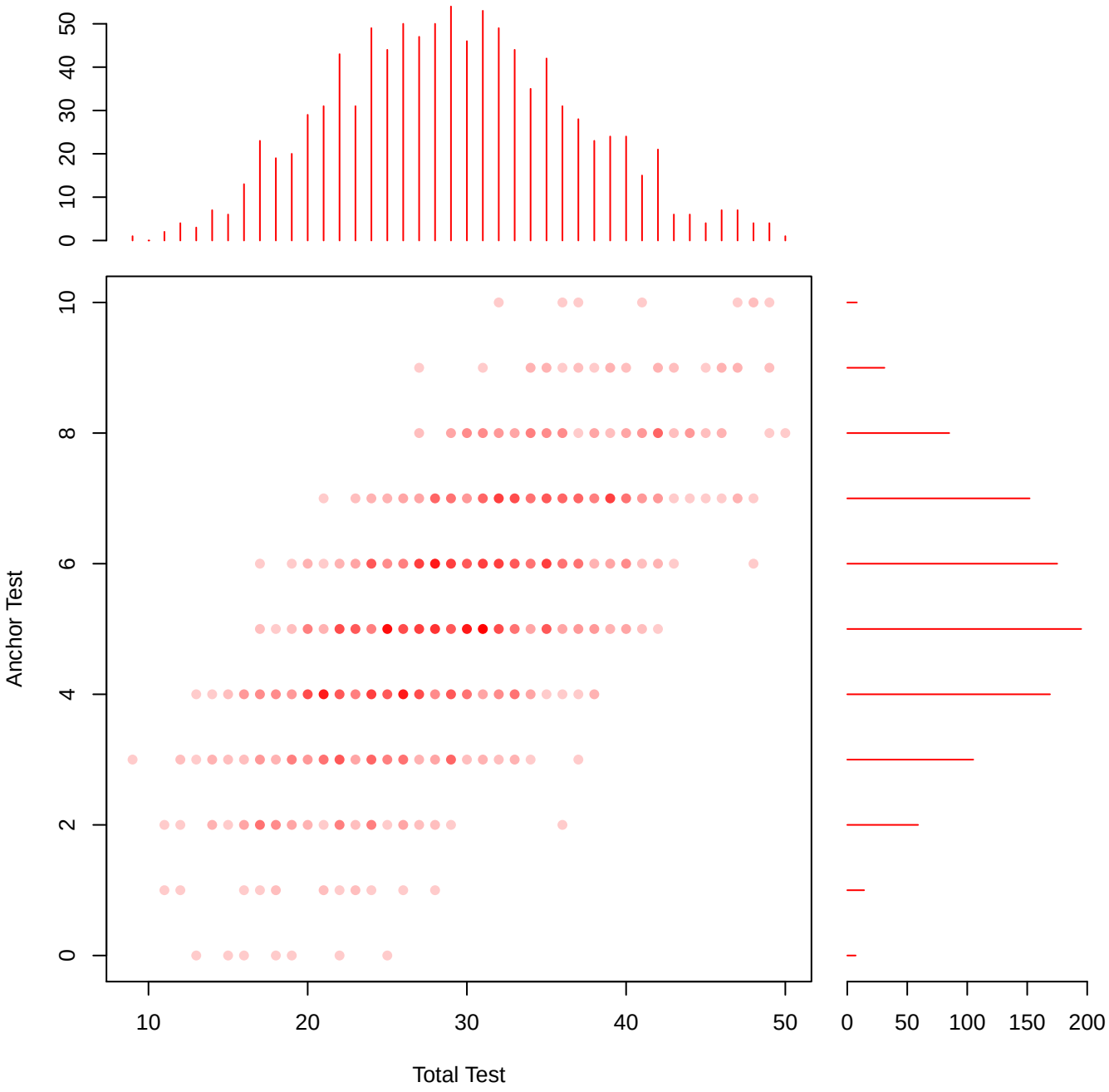


Figure 20:

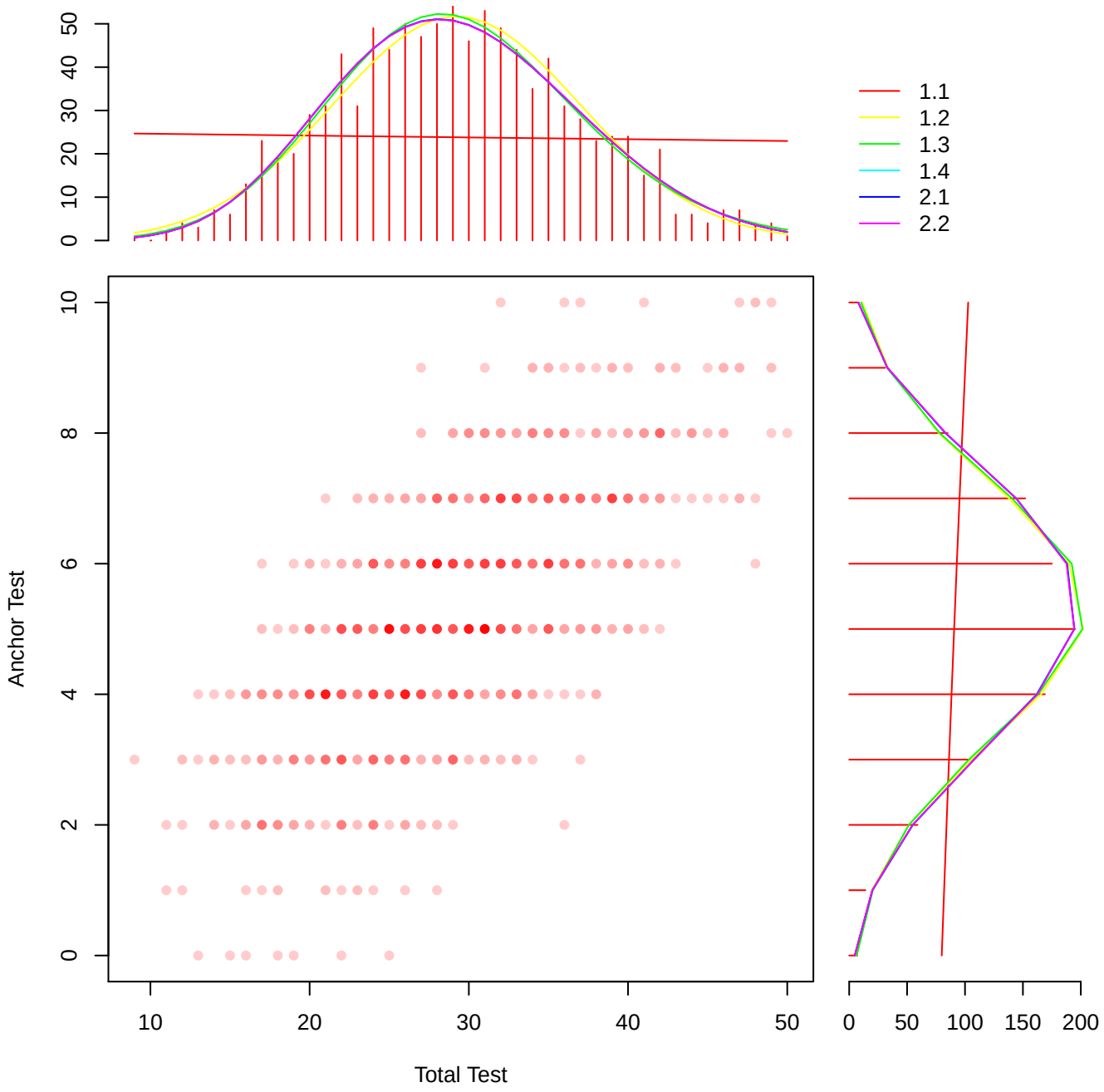

0
10
20
30
40
50
1.1
1.2
1.3
1.4
2.1
2.2
Anchor Test
0
2
4
6
8
10
10
20
30
40
50
Total Test
0
50
100
150
200

Figure 21:

$$x_i - \mu(X) = y_i - \mu(Y)$$
$$y_i = x_i - \mu(X) + \mu(Y)$$

General linear equating is a new approach to estimating a linear linking or equating function. The linear conversion is defined by setting standardized deviation scores (z-scores) on the two forms to be equal such that

$$\frac{x_i - \mu(X)}{\sigma(X)} = \frac{y_i - \mu(Y)}{\sigma(Y)}$$
$$y = \sigma(Y)\Big[\frac{x_i - \mu(X)}{\sigma(X)}\Big] + \mu(Y)$$

```r
library(equate)
linear <- equate(x, y,
                 type = "general linear",
                 smooth = "none")

linear
```

```
##
## General Linear Equating: x to y
##
## Design: nonequivalent groups
##
## Summary Statistics:
##     mean   sd skew  kurt min max    n
## x  28.86 7.63 0.03 -0.20   6  53 1000
## y  29.24 7.55 0.16 -0.34   9  50 1000
## yx 28.94 6.65 0.03 -0.20   9  50 1000
##
## Coefficients:
## intercept     slope        cx        cy        sx        sy
##    3.7660    0.8723   29.5000   29.5000   47.0000   41.0000
```

Equipercentile linking and equating define a nonlinear relationship between score scales by setting equal the cumulative distribution functions. Braun and Holland (1982) indicated that the following function is an equipercentile equating function when X and Y are continuous random variables:

$$y_i = G^{-1}[F(x_i)],$$

and by symmetry

$$x_i = F^{-1}[G(y_i)]$$

where G^{-1} is the inverse of the cumulative distribution function G.

Observed-Score Linking and Equating (Albano 2015):

The equipercentile equivalent of a form-X score on the Y scale is calculated by finding the percentile rank in X of particular score, and then finding the form-Y score associated with that form-Y percentile rank.

Equipercentile equating is appropriate when X and Y differ nonlinearly in difficulty, that is, when difficulty differences fluctuate across the score scale, potentially at each score point. Each coordinate on the equipercentile curve is estimated using information from the distributions of X and Y . Thus, compared to identity, mean, and linear equating, equipercentile equating is more susceptible to sampling error because it involves the estimation of as many parameters as there are unique score points on X.

Smoothing methods are typically used to reduce irregularities due to sampling error in either the score distributions or the equipercentile equating function itself. Two commonly used smoothing methods include polynomial loglinear presmoothing (Holland P. W. and Thayer 2000) and cubic-spline postsmoothing (M. J. Kolen 1984).

```
library(equate)
equiper <- equate(x, y,
                  type = "equipercentile",
                  smoothmethod = "loglinear",
                  internal = TRUE,
                  boot = TRUE)

equiper
```

```
##
## Equipercentile Equating: x to y
##
## Design: nonequivalent groups
```

```
##
## Smoothing Method: loglinear presmoothing
##
## Summary Statistics:
##     mean   sd skew  kurt  min   max    n
## x  28.86 7.63 0.03 -0.20 6.00 53.00 1000
## y  29.24 7.55 0.16 -0.34 9.00 50.00 1000
## yx 29.24 7.54 0.15 -0.34 8.84 50.42 1000
```

Observed-Score Linking and Equating (Albano 2015):

Circle-arc Equating *is the simplified approach to circle-arc equating, as demonstrated by (Livingston 2009), involves combining a circle-arc with the identity function. When the low and high scores differ for the X and Y scales, this becomes the identity linking function. The linear component can be omitted, and symmetric circle-arc equating used, with simple = FALSE. The result is an equating function based only on the circle-arc that passes through the points lowp, highp, and the estimated midpoint.*

```r
library(equate)
circ.arc <- equate(x, y,
                     type = "circle-arc",
                     method = "braun/holland")

circ.arc
```

```
##
## Braun/Holland Circle-Arc Equating: x to y
##
## Design: nonequivalent groups
##
## Summary Statistics:
##     mean   sd skew  kurt min max    n
## x  28.86 7.63 0.03 -0.20   6  53 1000
## y  29.24 7.55 0.16 -0.34   9  50 1000
## yx 28.97 6.66 0.03 -0.20   9  50 1000
##
## Coefficients:
##  intercept      slope    xcenter    ycenter          r
```

```
##     3.7660     0.8723    29.5000 -8466.0436  8466.0762
```

A composite of mean and identity

```r
library(equate)
id.mean <- composite(list(equate(x, y, type = "i", boot = TRUE, reps = 5),
                          equate(x, y, type = "m", boot = TRUE, reps = 5)),
                     wc = .5, symmetric = TRUE)

id.mean
```

```
## [[1]]
##
## Composite Equating: x to y
##
## Design: nonequivalent groups
##
## Summary Statistics:
##     mean   sd skew  kurt  min   max    n
## x  28.86 7.63 0.03 -0.20 6.00 53.00 1000
## y  29.24 7.55 0.16 -0.34 9.00 50.00 1000
## yx 29.09 6.65 0.03 -0.20 9.15 50.15 1000
##
##
## [[2]]
##
## Identity Equating: x to y
##
## Design: nonequivalent groups
##
## Summary Statistics:
##     mean   sd skew  kurt min max    n
## x  28.86 7.63 0.03 -0.20   6  53 1000
## y  29.24 7.55 0.16 -0.34   9  50 1000
## yx 28.94 6.65 0.03 -0.20   9  50 1000
##
## Coefficients:
```

```
## intercept     slope        cx        cy        sx        sy
##    3.7660    0.8723   29.5000   29.5000   47.0000   41.0000
##
##
## [[3]]
##
## Mean Equating: x to y
##
## Design: nonequivalent groups
##
## Summary Statistics:
##     mean   sd skew  kurt min  max    n
## x  28.86 7.63 0.03 -0.20 6.0 53.0 1000
## y  29.24 7.55 0.16 -0.34 9.0 50.0 1000
## yx 29.24 6.65 0.03 -0.20 9.3 50.3 1000
##
## Coefficients:
## intercept     slope        cx        cy        sx        sy
##    4.0626    0.8723   29.5000   29.5000   47.0000   41.0000
##
##
## attr(,"class")
## [1] "equate.list"
```

Plot all six equivalent-groups design

```r
# Conversion table containing scores on X with their form Y equivalents.
round(data.frame(scale = equiper$concordance$scale,
                 equiper = equiper$concordance$yx,
                 linear = linear$concordance$yx,
                 circ.arc = circ.arc$concordance$yx,
                 composit = id.mean[[1]]$concordance$yx,
                 identity = id.mean[[2]]$concordance$yx,
                 mean.eq = id.mean[[3]]$concordance$yx), 2)
```

```
##    scale equiper linear circ.arc composit identity mean.eq
## 1      6    8.84   9.00     9.00     9.15     9.00    9.30
```

```
## 2      7    9.62   9.87     9.88    10.02     9.87   10.17
## 3      8   10.40  10.74    10.75    10.89    10.74   11.04
## 4      9   11.16  11.62    11.62    11.77    11.62   11.91
## 5     10   11.96  12.49    12.50    12.64    12.49   12.79
## 6     11   12.78  13.36    13.37    13.51    13.36   13.66
## 7     12   13.62  14.23    14.25    14.38    14.23   14.53
## 8     13   14.46  15.11    15.12    15.25    15.11   15.40
## 9     14   15.28  15.98    16.00    16.13    15.98   16.28
## 10    15   16.11  16.85    16.87    17.00    16.85   17.15
## 11    16   16.97  17.72    17.75    17.87    17.72   18.02
## 12    17   17.84  18.60    18.62    18.74    18.60   18.89
## 13    18   18.72  19.47    19.49    19.62    19.47   19.76
## 14    19   19.61  20.34    20.37    20.49    20.34   20.64
## 15    20   20.51  21.21    21.24    21.36    21.21   21.51
## 16    21   21.42  22.09    22.11    22.23    22.09   22.38
## 17    22   22.34  22.96    22.99    23.11    22.96   23.25
## 18    23   23.27  23.83    23.86    23.98    23.83   24.13
## 19    24   24.21  24.70    24.73    24.85    24.70   25.00
## 20    25   25.17  25.57    25.61    25.72    25.57   25.87
## 21    26   26.14  26.45    26.48    26.60    26.45   26.74
## 22    27   27.13  27.32    27.35    27.47    27.32   27.62
## 23    28   28.12  28.19    28.22    28.34    28.19   28.49
## 24    29   29.13  29.06    29.10    29.21    29.06   29.36
## 25    30   30.15  29.94    29.97    30.08    29.94   30.23
## 26    31   31.18  30.81    30.84    30.96    30.81   31.11
## 27    32   32.21  31.68    31.71    31.83    31.68   31.98
## 28    33   33.26  32.55    32.59    32.70    32.55   32.85
## 29    34   34.31  33.43    33.46    33.57    33.43   33.72
## 30    35   35.36  34.30    34.33    34.45    34.30   34.59
## 31    36   36.42  35.17    35.20    35.32    35.17   35.47
## 32    37   37.48  36.04    36.07    36.19    36.04   36.34
## 33    38   38.54  36.91    36.94    37.06    36.91   37.21
## 34    39   39.60  37.79    37.81    37.94    37.79   38.08
## 35    40   40.66  38.66    38.69    38.81    38.66   38.96
## 36    41   41.71  39.53    39.56    39.68    39.53   39.83
## 37    42   42.75  40.40    40.43    40.55    40.40   40.70
```

```
## 38    43   43.77  41.28    41.30    41.42    41.28   41.57
## 39    44   44.76  42.15    42.17    42.30    42.15   42.45
## 40    45   45.71  43.02    43.04    43.17    43.02   43.32
## 41    46   46.61  43.89    43.91    44.04    43.89   44.19
## 42    47   47.45  44.77    44.78    44.91    44.77   45.06
## 43    48   48.24  45.64    45.65    45.79    45.64   45.93
## 44    49   48.91  46.51    46.52    46.66    46.51   46.81
## 45    50   49.43  47.38    47.39    47.53    47.38   47.68
## 46    51   49.89  48.26    48.26    48.40    48.26   48.55
## 47    52   50.21  49.13    49.13    49.28    49.13   49.42
## 48    53   50.42  50.00    50.00    50.15    50.00   50.30
```

```r
plot(equiper,
     linear,
     circ.arc,
     id.mean[[1]],
     legendplace = "top",
     addident = FALSE)
```

1.4 Differential Item Functioning (DIF)

1.4.1 Uniform DIF

Theory and Practice of Item Response Theory Methodology in the Social Sciences (**???**):

Uniform DIF is the simplest type of DIF where the magnitude of conditional dependency is relatively invariant across the latent trait continuum (Θ). *The item of interest consistently gives one group an advantage across all levels of ability. Within an item response theory (IRT) framework this would be evidenced when both item characteristic curves (ICC) are equally discriminating yet exhibit differences in the difficulty parameters* (i.e., $a_r = a_f$ and $b_r < b_f$) *as depicted below.*

```r
# parameter matrix for uniform diff
parameter.matrix <- matrix(c(1, -.5, 0,
                     1, .5, 0), 2, 3, byrow = TRUE)

source("IRF.R")
IRF(parameter.matrix, 1, irf.plot = TRUE)
```

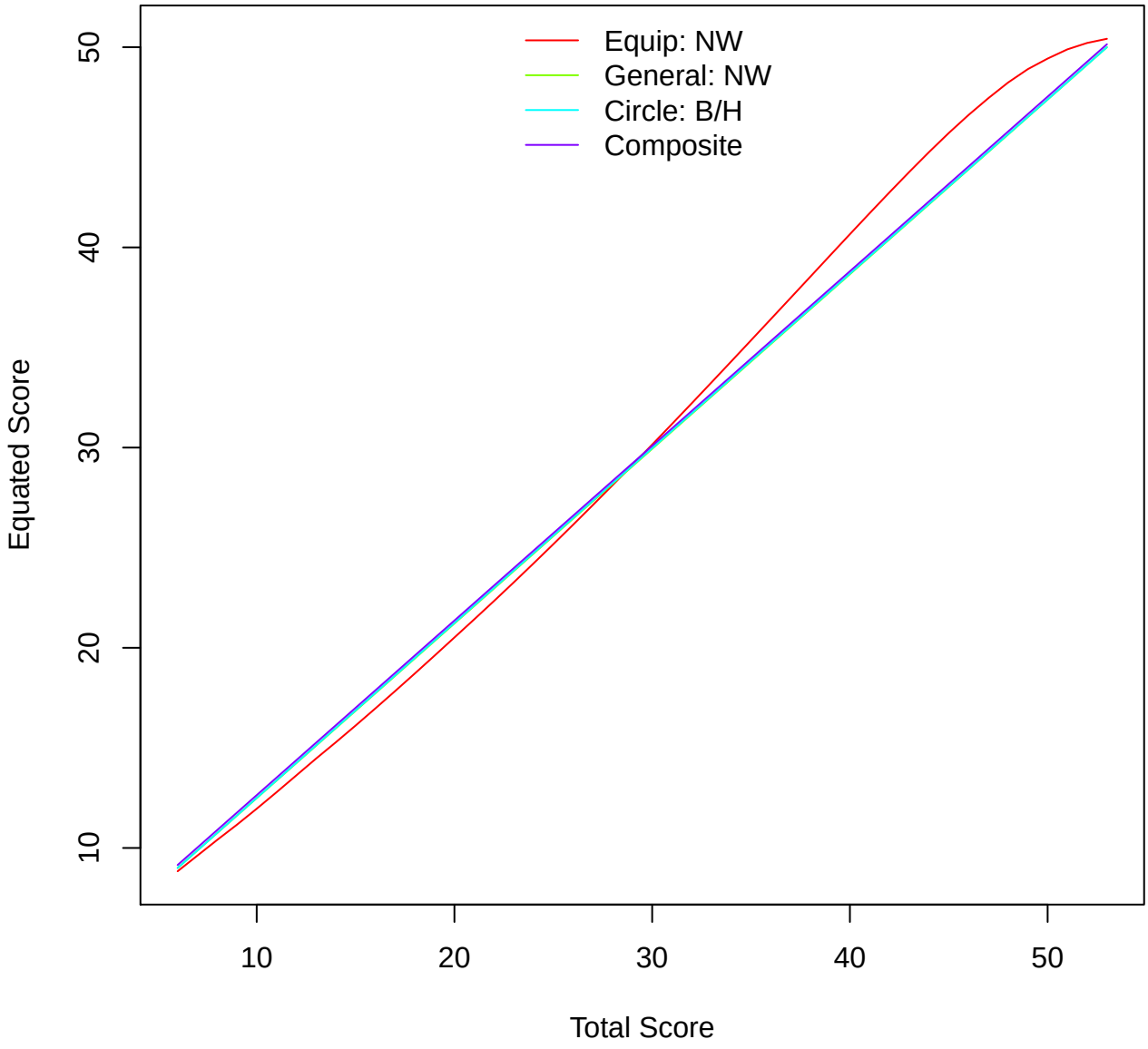

Equip: NW
General: NW
Circle: B/H
Composite
Equated Score
10
20
30
40
50
Total Score
10
20
30
40
50

Figure 22:

```
## $probabilities
##         [,1]    [,2]
## [1,] 0.81757 0.62246
##
## $expected.score
## [1] 72.002
```

```r
# legend
legend(-4, 1, c("male", "female"),
       col = c("dodgerblue", "tomato"),
       text.col = c("dodgerblue", "tomato"),
       lty = 1,
       lwd = 5,
       bty = "n",
       merge = TRUE)
```

1.4.2 Non-Uniform DIF

Theory and Practice of Item Response Theory Methodology in the Social Sciences (**???**):

In nonuniform DIF rather than a consistent advantage being given to the reference group across the ability continuum, the conditional dependency moves and changes direction at different locations on the Θ *continuum. For instance, an item may give the reference group a minor advantage at the lower end of the continuum while a major advantage at the higher end. Also, unlike uniform DIF, an item can simultaneously vary in discrimination for the two groups while also varying in difficulty* (i.e., $a_r \neq a_f$ and $b_r < b_f$).

```r
# parameter matrix for uniform diff
parameter.matrix <- matrix(c(1.3, 0, 0,
                             0.7, 0, 0), 2, 3, byrow = TRUE)

source("IRF.R")
IRF(parameter.matrix, irf.plot = TRUE)
```

```
## $probabilities
##      [,1] [,2]
## [1,]  0.5  0.5
```

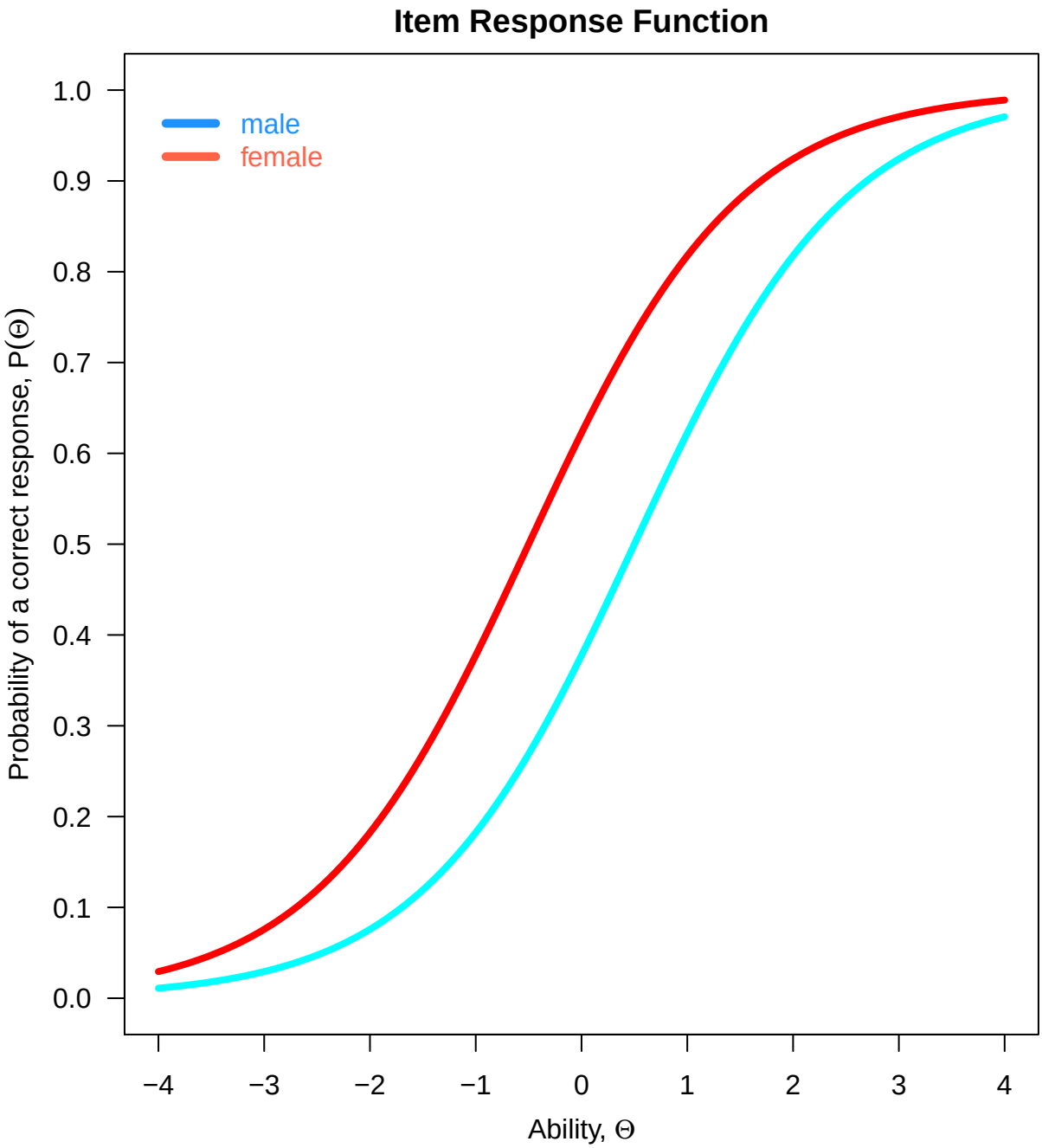

Item Response Function
male
female
Probability of a correct response, P(Θ)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−4
−3
−2
−1
0
1
2
3
4
Ability, Θ

Figure 23:

```
##
## $expected.score
## [1] 50
```

```r
# legend
legend(-4, 1, c("male", "female"),
       col = c("dodgerblue", "tomato"),
       text.col = c("dodgerblue", "tomato"),
       lty = 1,
       lwd = 5,
       bty = "n",
       merge = TRUE)
```

1.4.3 Perform DIF Detection Using Mantel-Haenszel, Method

```r
# import data
dif.data <- read.csv("test2.csv", header = TRUE, sep = ",")

# check
head(dif.data)
```

```
##   Gender V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18
## 1      1  1  1  1  1  0  1  1  0  0   1   1   0   0   0   1   1   1   1
## 2      1  1  1  1  1  0  1  1  1  1   1   1   1   0   1   1   0   0   1
## 3      1  1  1  0  0  1  1  1  0  0   1   1   0   1   1   0   0   0   0
## 4      1  0  1  0  1  1  1  0  0  0   0   0   0   0   0   0   1   0   0
## 5      1  1  1  1  1  1  1  0  1  1   1   1   1   1   1   1   1   1   1
## 6      1  0  1  1  0  0  1  0  0  0   0   1   1   1   0   0   0   0   0
##   V19 V20 V21 V22 V23 V24 V25
## 1   1   1   1   1   1   1   1
## 2   1   1   1   1   0   1   1
## 3   1   0   1   1   0   1   0
## 4   0   0   0   1   1   1   0
## 5   1   1   1   1   1   1   1
## 6   1   0   0   1   0   1   0
```

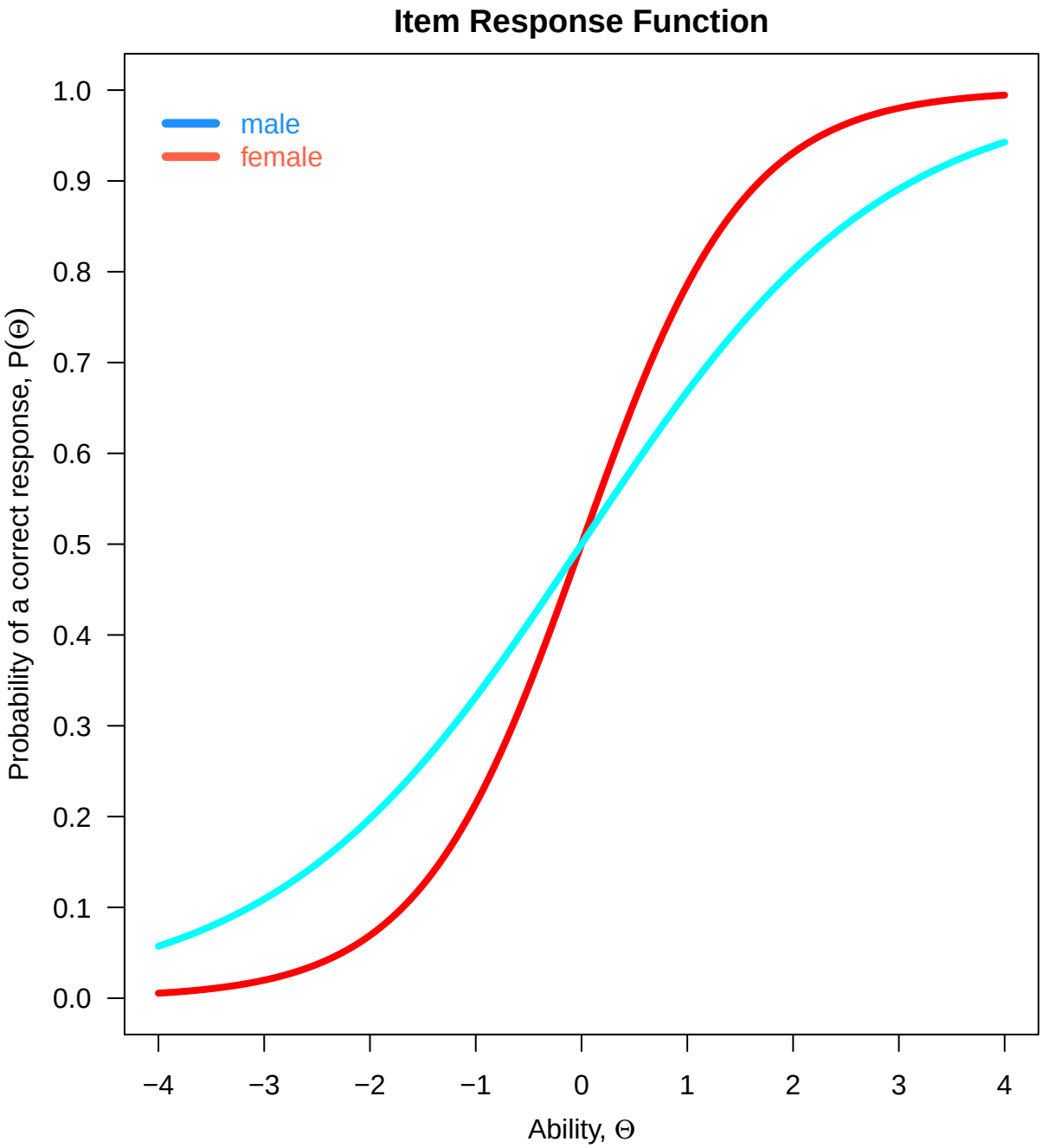

Item Response Function
male
female
Probability of a correct response, P(Θ)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−4
−3
−2
−1
0
1
2
3
4
Ability, Θ

Figure 24:

```r
str(dif.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
test.data <- dif.data[,-1]

# Add a column of raw scores
dif.data$scores <- rowSums(test.data)
summary(dif.data$scores)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##     3.0    11.0    14.0    14.8    19.0    25.0
```

```r
# Add a column of decile factors
quantile(dif.data$scores, probs = seq(0.25, 1, by = 0.25))
```

```
##  25%  50%  75% 100%
##   11   14   19   25
```

```r
dif.data$deciles <- cut(dif.data$scores,
                        breaks = c(-Inf, quantile(dif.data$scores,
                                                  probs = seq(0.25, 1, by = 0.25)),
                                   Inf))
summary(dif.data$deciles)
```

```
## (-Inf,11]   (11,14]   (14,19]   (19,25] (25, Inf]
##       537       342       466       375         0
```

```r
difresults <- difR::difMH(dif.data[, 1:26],
                          group = "Gender",
                          focal.name = 1,
                          alpha = 0.01)

difresults
```

```
##
## Detection of Differential Item Functioning using Mantel-Haenszel method
## with continuity correction and without item purification
##
## Results based on asymptotic inference
##
## No set of anchor items was provided
##
## Mantel-Haenszel Chi-square statistic:
##
##     Stat.    P-value
## V1    1.6940   0.1931
```

```
## V2    0.6207   0.4308
## V3    1.0023   0.3168
## V4    0.0426   0.8364
## V5    0.0019   0.9653
## V6    0.8315   0.3618
## V7    3.1715   0.0749 .
## V8    2.5334   0.1115
## V9    0.0035   0.9531
## V10   4.3188   0.0377 *
## V11 102.7551   0.0000 ***
## V12   0.0832   0.7730
## V13   3.3598   0.0668 .
## V14   1.8200   0.1773
## V15   0.0532   0.8176
## V16   0.0414   0.8388
## V17   0.1296   0.7189
## V18   0.7640   0.3821
## V19   0.0238   0.8773
## V20   0.0033   0.9544
## V21   0.8746   0.3497
## V22   1.2326   0.2669
## V23   0.3873   0.5337
## V24   0.8383   0.3599
## V25   6.7209   0.0095 **
##
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Detection threshold: 6.6349 (significance level: 0.01)
##
## Items detected as DIF items:
##
##  V11
##  V25
##
##
## Effect size (ETS Delta scale):
```

```
##
## Effect size code:
##  'A': negligible effect
##  'B': moderate effect
##  'C': large effect
##
##     alphaMH deltaMH
## V1   1.1980 -0.4246 A
## V2   1.1048 -0.2342 A
## V3   0.8805  0.2990 A
## V4   1.0325 -0.0753 A
## V5   0.9987  0.0031 A
## V6   1.6431 -1.1670 B
## V7   1.2160 -0.4596 A
## V8   1.1955 -0.4196 A
## V9   1.0119 -0.0277 A
## V10  1.2798 -0.5798 A
## V11  0.3034  2.8025 C
## V12  1.0408 -0.0941 A
## V13  1.2322 -0.4906 A
## V14  1.1818 -0.3926 A
## V15  1.0313 -0.0724 A
## V16  0.9719  0.0670 A
## V17  1.0461 -0.1059 A
## V18  0.8985  0.2515 A
## V19  1.0276 -0.0640 A
## V20  1.0131 -0.0307 A
## V21  1.1292 -0.2855 A
## V22  1.3152 -0.6439 A
## V23  1.0751 -0.1701 A
## V24  0.8065  0.5054 A
## V25  0.7244  0.7577 A
##
## Effect size codes: 0 'A' 1.0 'B' 1.5 'C'
##  (for absolute values of 'deltaMH')
##
```

```
## Output was not captured!
```

```r
plot(difresults)
```

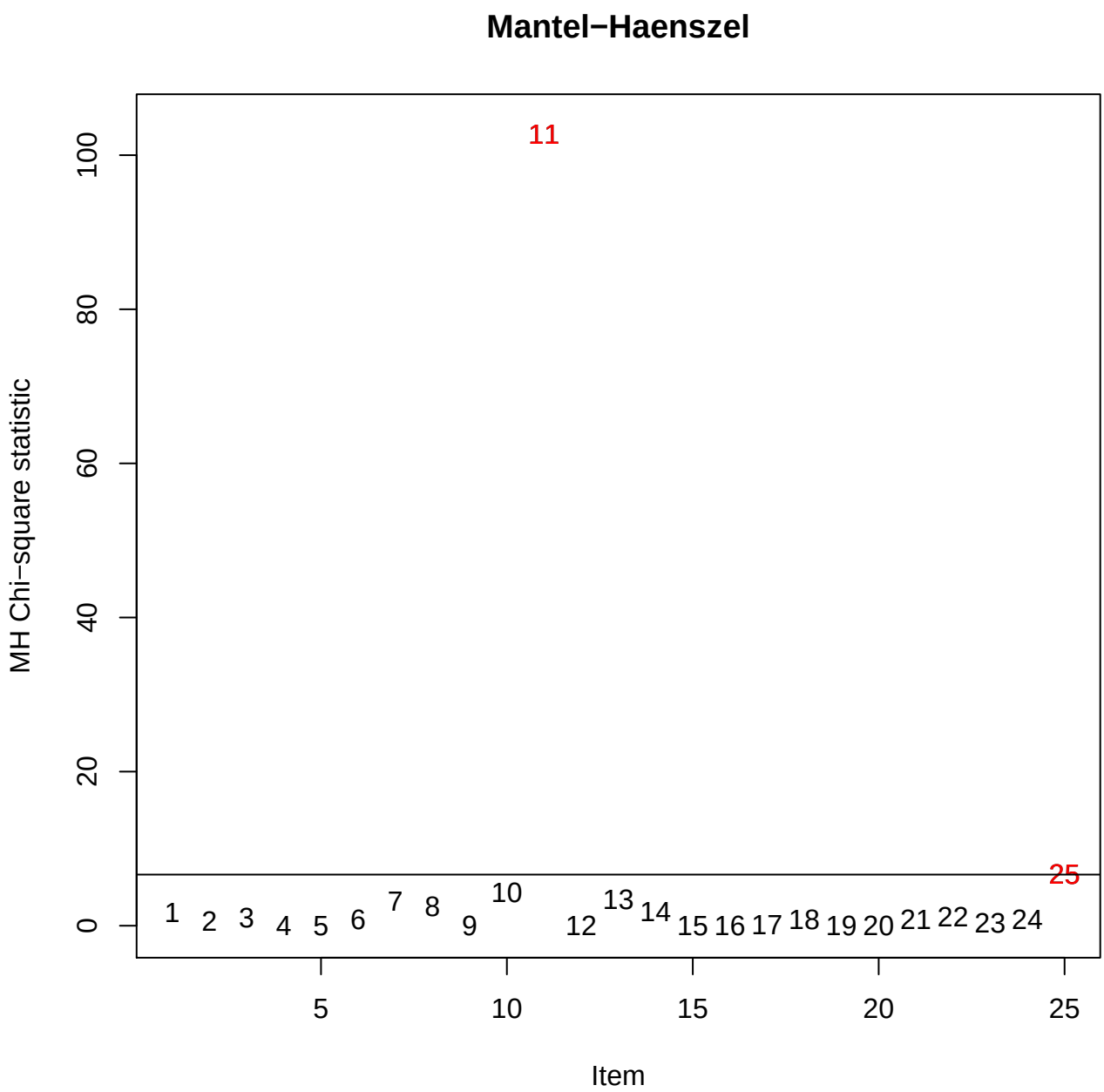


Figure 25:

```
## The plot was not captured!
```

1.4.4 Perform DIF Detection Using Breslow-Day Method

```r
difBD <- difR::difBD(dif.data[, 1:26],
                     group = "Gender",
                     focal.name = 2,
                     purify = FALSE,
                     alpha = 0.01)

difBD
```

```
##
## Detection of Differential Item Functioning using Breslow-Day method
## without item purification
##
## No set of anchor items was provided
##
## Breslow-Day statistic:
##
##     Stat.    df       P-value
## V1   13.9049  18.0000   0.7353
## V2   29.5451  20.0000   0.0776 .
## V3   21.5465  17.0000   0.2028
## V4   15.4914  18.0000   0.6280
## V5   19.5482  20.0000   0.4865
## V6    7.6583   7.0000   0.3637
## V7   19.1563  19.0000   0.4469
## V8   28.3599  19.0000   0.0767 .
## V9   17.8732  20.0000   0.5958
## V10  12.4255  18.0000   0.8245
## V11  37.9025  17.0000   0.0025 **
## V12  19.6895  19.0000   0.4135
## V13  19.0885  19.0000   0.4512
## V14  29.8974  19.0000   0.0531 .
## V15  15.7983  20.0000   0.7291
## V16  15.0052  18.0000   0.6616
## V17  24.9896  19.0000   0.1609
## V18  15.9941  18.0000   0.5930
## V19  10.0502  17.0000   0.9015
```

```
## V20  33.0567  20.0000   0.0333 *
## V21   9.4499  16.0000   0.8937
## V22  13.0381  12.0000   0.3663
## V23 126.4894  19.0000   0.0000 ***
## V24  14.9394  13.0000   0.3112
## V25  17.5889  16.0000   0.3485
##
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Significance level: 0.01
##
## Items detected as DIF items:
##
##  V11
##  V23
##
## Output was not captured!
```

```r
plot(difBD)
```

```
## The plot was not captured!
```

1.4.5 Perform DIF Detection Using Logistic regression Method

```r
logistic.dif <- difR::difLogistic(dif.data[c(1:26)],
                                  group = "Gender",
                                  focal.name = 2,
                                  purify = FALSE)

logistic.dif
```

```
##
## Detection of both types of Differential Item Functioning
## using Logistic regression method, without item purification
## and with LRT DIF statistic
```

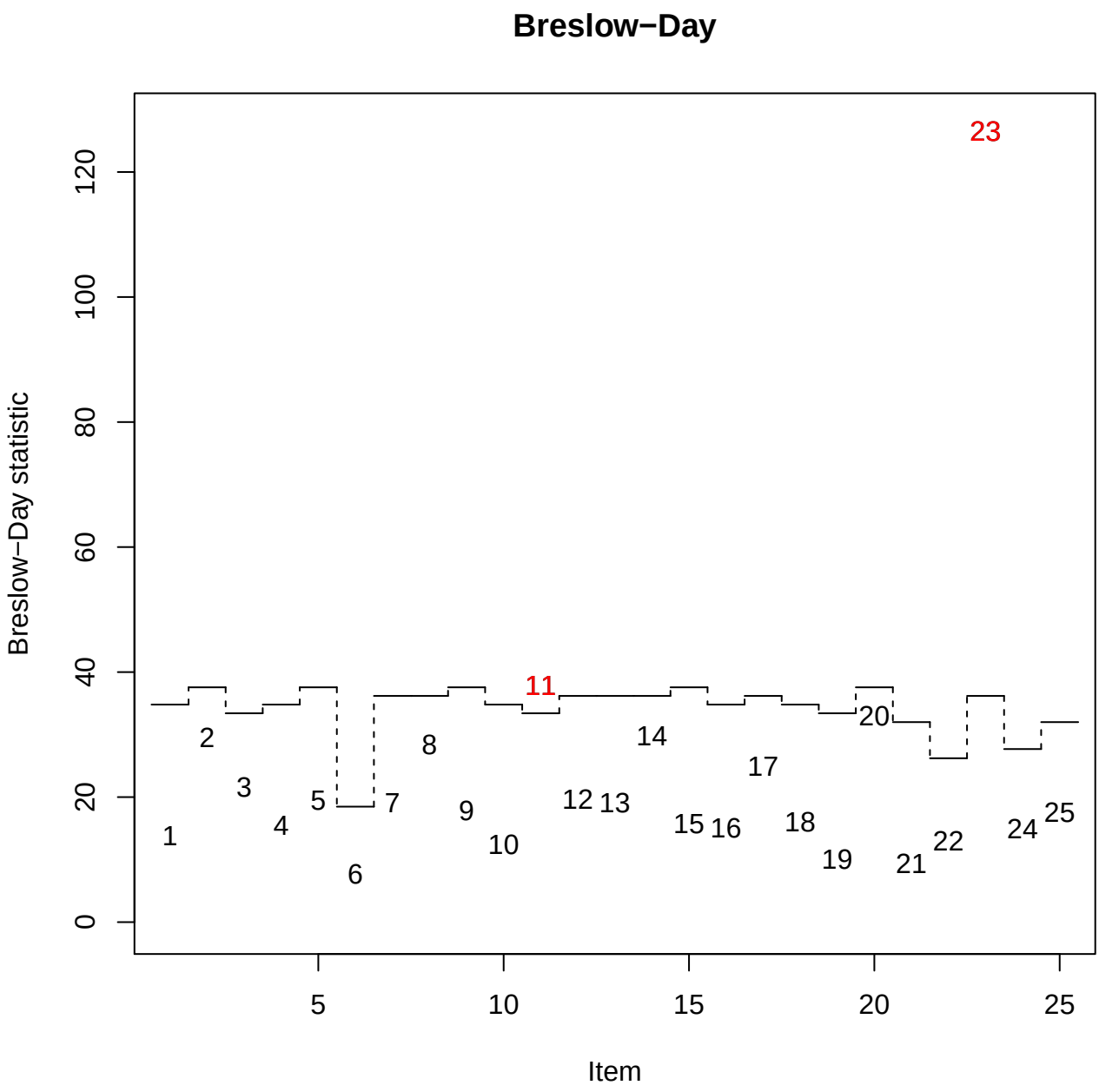

Breslow−Day
Breslow−Day statistic
Item
0
20
40
60
80
100
120
5
10
15
20
25
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Figure 26:

```
##
## Matching variable: test score
##
## No set of anchor items was provided
##
## Logistic regression DIF statistic:
##
##     Stat.    P-value
## V1    3.2193   0.2000
## V2    2.0670   0.3558
## V3    1.6569   0.4367
## V4    0.0922   0.9549
## V5    4.0223   0.1338
## V6    1.8693   0.3927
## V7    3.9788   0.1368
## V8    5.1314   0.0769 .
## V9    0.6488   0.7230
## V10   5.8281   0.0543 .
## V11 109.9090   0.0000 ***
## V12   0.3015   0.8601
## V13   6.3557   0.0417 *
## V14   8.4932   0.0143 *
## V15   1.8673   0.3931
## V16   1.2384   0.5384
## V17   2.8499   0.2405
## V18   1.4416   0.4864
## V19   0.0354   0.9824
## V20   1.9955   0.3687
## V21   1.3598   0.5067
## V22   1.8036   0.4058
## V23 103.7505   0.0000 ***
## V24   1.2068   0.5470
## V25   6.1843   0.0454 *
##
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Detection threshold: 5.9915 (significance level: 0.05)
##
## Items detected as DIF items:
##
##  V11
##  V13
##  V14
##  V23
##  V25
##
##
## Effect size (Nagelkerke's R^2):
##
## Effect size code:
##  'A': negligible effect
##  'B': moderate effect
##  'C': large effect
##
##     R^2    ZT JG
## V1  0.0014 A  A
## V2  0.0010 A  A
## V3  0.0007 A  A
## V4  0.0000 A  A
## V5  0.0018 A  A
## V6  0.0082 A  A
## V7  0.0020 A  A
## V8  0.0026 A  A
## V9  0.0004 A  A
## V10 0.0026 A  A
## V11 0.0473 A  B
## V12 0.0001 A  A
## V13 0.0033 A  A
## V14 0.0037 A  A
## V15 0.0010 A  A
## V16 0.0006 A  A
## V17 0.0014 A  A
```

```
## V18 0.0006 A  A
## V19 0.0000 A  A
## V20 0.0009 A  A
## V21 0.0005 A  A
## V22 0.0027 A  A
## V23 0.0472 A  B
## V24 0.0016 A  A
## V25 0.0025 A  A
##
## Effect size codes:
##  Zumbo & Thomas (ZT): 0 'A' 0.13 'B' 0.26 'C' 1
##  Jodoin & Gierl (JG): 0 'A' 0.035 'B' 0.07 'C' 1
##
##  Output was not captured!
```

```r
plot(logistic.dif)
```

```
## The plot was not captured!
```

1.4.6 Cases Identified as DIF (using Breslow-Day method)

```r
# Males
male.data <- subset(dif.data, subset = Gender == 1)

aggregate.male <- aggregate(data.matrix(male.data),
                            by = list(male.data$scores),
                            FUN = mean,
                            simplify = TRUE)

thick.male <- aggregate(data.matrix(male.data),
                        by = list(male.data$deciles),
                        FUN = mean,
                        simplify = TRUE)

# Females
```

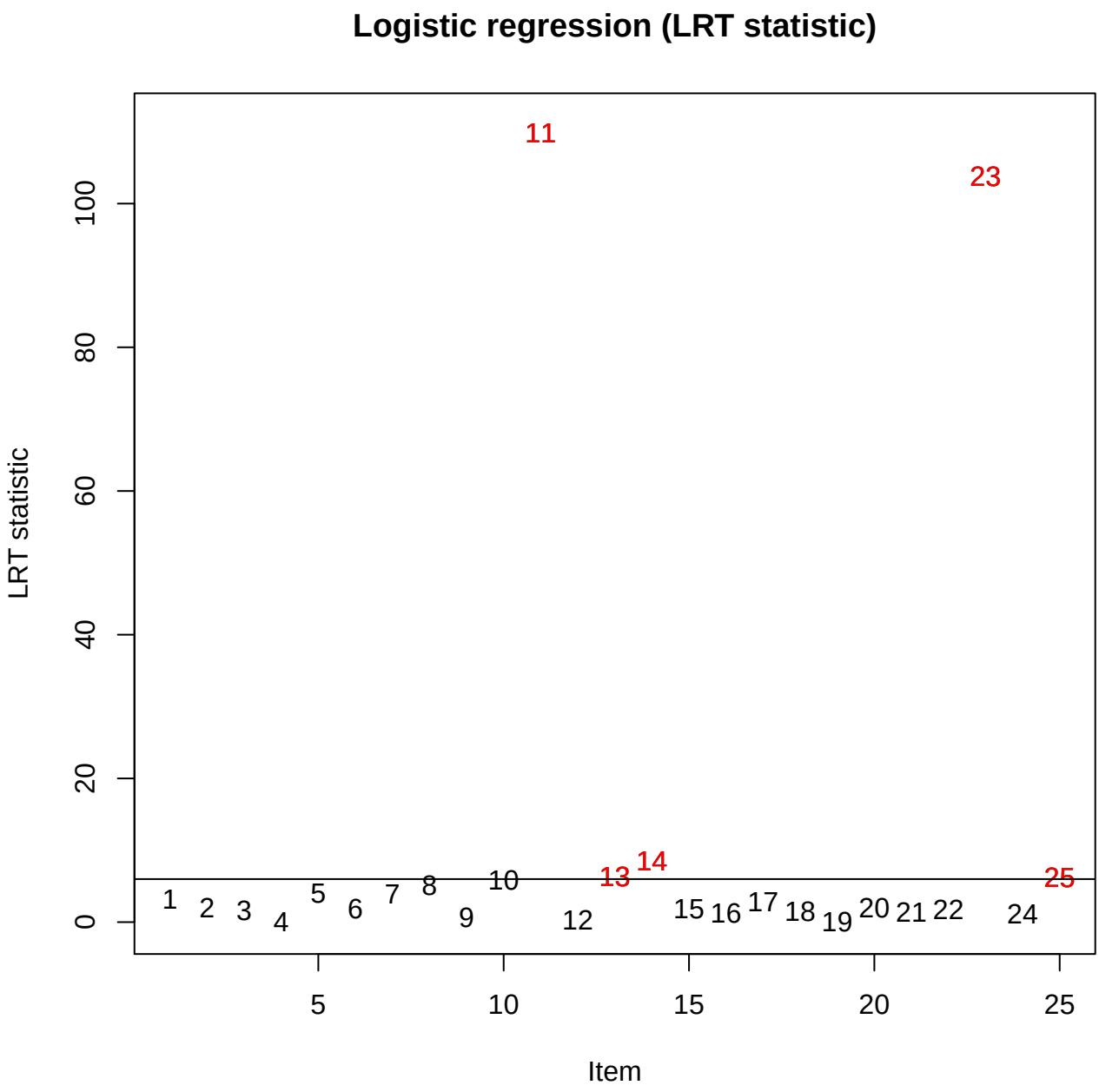

Logistic regression (LRT statistic)
LRT statistic
0
20
40
60
80
100
Item
5
10
15
20
25
11
23
14
13
25
1
2
3
4
5
6
7
8
9
10
12
15
16
17
18
19
20
21
22
24

Figure 27:

```
female.data <- subset(dif.data, subset = Gender == 2)

aggregate.female <- aggregate(data.matrix(female.data),
                              by = list(female.data$scores),
                              FUN = mean,
                              simplify = TRUE)

thick.female <- aggregate(data.matrix(female.data),
                          by = list(female.data$deciles),
                          FUN = mean,
                          simplify = TRUE)
```

Item 11 (uniform DIF)

```
dif.plot <-
  function(group1score, group2score, group1item, group2item){

    # graph boundaries
    xmin <- min(c(group1score, group2score))
    xmax <- max(c(group1score, group2score))
    ymin <- min(c(group1item, group2item))
    ymax <- max(c(group1item, group2item))

    # group 1
    plot(group1score,
         group1item,
         type = "l",
         xlim = c(xmin, xmax),
         ylim = c(ymin, ymax),
         col = adjustcolor("dodgerblue", alpha.f = .6),
         lwd = 6,
         axes = FALSE,
         xlab = "",
         ylab = "")

    # group 2
```

```
lines(group2score,
      group2item,
      type = "l",
      col = adjustcolor("tomato", alpha.f = .6),
      lwd = 6)

# x-axis
axis(1, pretty(c(xmin, xmax), 10))

mtext("Score",
      side = 1,
      col = "black",
      line = 2.5)

box()

# y-axis
axis(2, pretty(c(ymin, ymax), 10),
     las = 1,
     col = "black",
     col.axis = "black")

mtext(expression(paste("Probability of correct response, ", P(Theta))),
      side = 2,
      col = "black",
      line = 2.5)

# legend
legend(xmin, ymax,
       c("male", "female"),
       col = c("dodgerblue", "tomato"),
       text.col = c("dodgerblue", "tomato"),
       lty = 1,
       lwd = 5,
       bty = "n",
       merge = TRUE)
```

```
  }

dif.plot(thick.male$score, thick.female$score, thick.male$V11, thick.female$V11)
title(main = "Differential item functioning by gender\n (Item 11)")
```

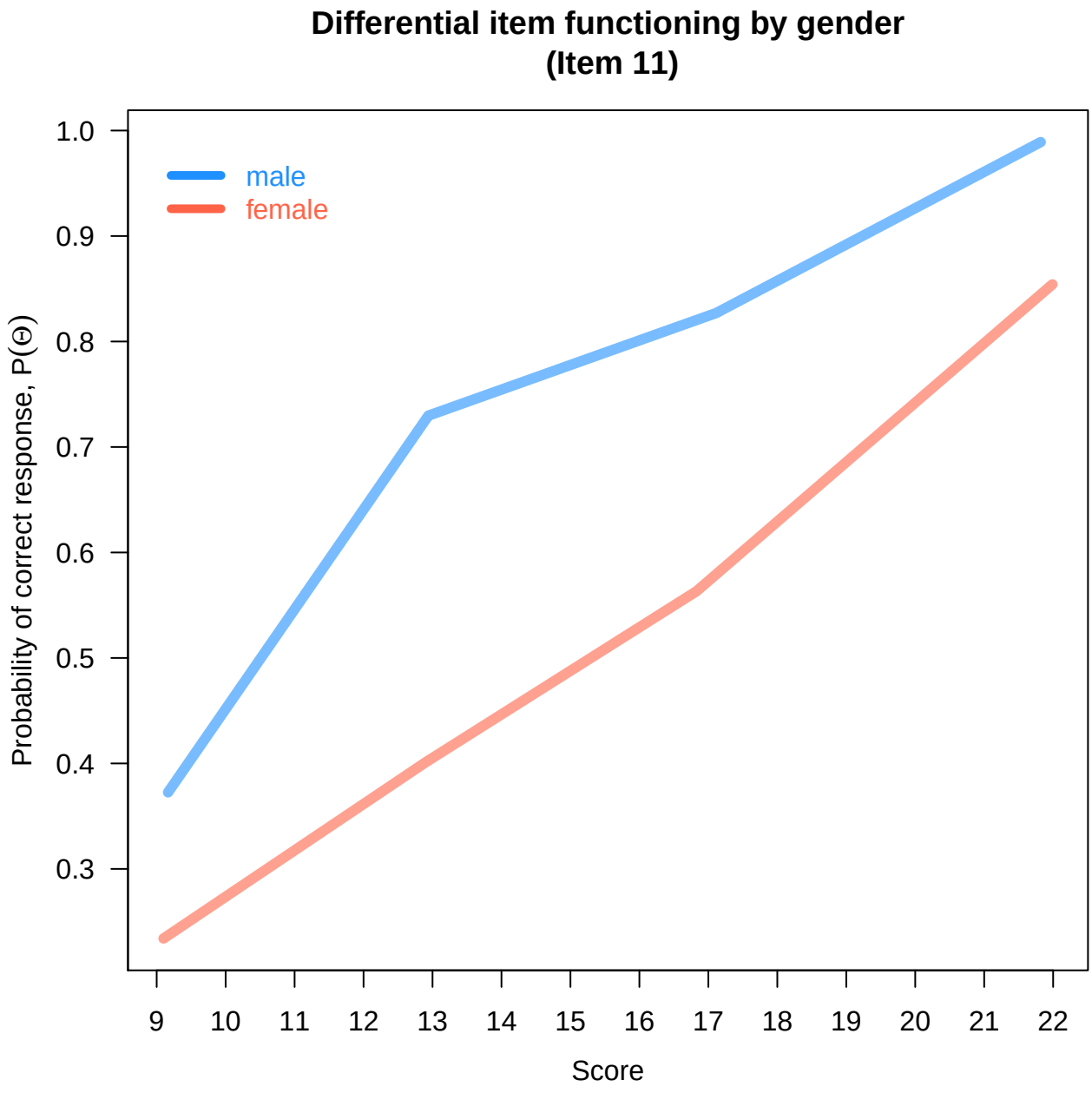


Figure 28:

Mantel-Haenszel chi-squared test with continuity correction

```r
mantelhaen.test(table(dif.data$Gender,dif.data$V11,dif.data$score))
```

```
##
##  Mantel-Haenszel chi-squared test with continuity correction
##
## data:  table(dif.data$Gender, dif.data$V11, dif.data$score)
## Mantel-Haenszel X-squared = 103, df = 1, p-value <2e-16
## alternative hypothesis: true common odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.23941 0.38461
## sample estimates:
## common odds ratio
##           0.30344
```

Woolf-test on Homogeneity of Odds Ratios

```r
vcd::woolf_test(table(dif.data$Gender, dif.data$V11, dif.data$deciles))
```

```
##
##  Woolf-test on Homogeneity of Odds Ratios (no 3-Way assoc.)
##
## data:  table(dif.data$Gender, dif.data$V11, dif.data$deciles)
## X-squared = 12.4, df = 4, p-value = 0.015
```

Item 23 (non-uniform DIF)

```r
dif.plot(thick.male$score, thick.female$score, thick.male$V23, thick.female$V23)
title(main = "Differential item functioning by gender\n (Item 23)")
```

```r
# Mantel-Haenszel chi-squared test with continuity correction
mantelhaen.test(table(dif.data$Gender, dif.data$V23, dif.data$score))
```

```
##
##  Mantel-Haenszel chi-squared test with continuity correction
##
## data:  table(dif.data$Gender, dif.data$V23, dif.data$score)
```

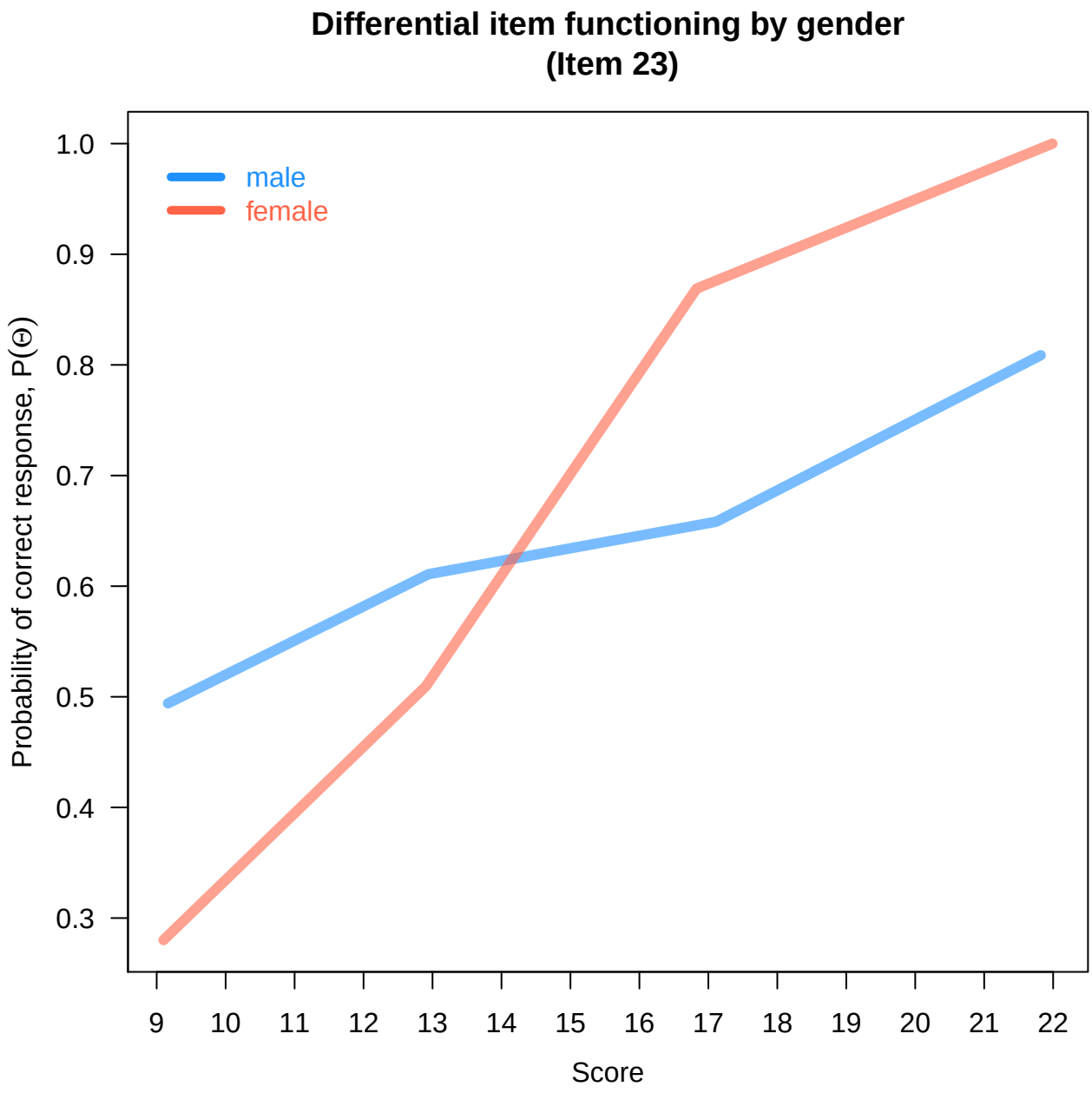

Differential item functioning by gender
(Item 23)
male
female
Probability of correct response, P(Θ)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
9
10
11
12
13
14
15
16
17
18
19
20
21
22
Score

Figure 29:

```
## Mantel-Haenszel X-squared = 0.387, df = 1, p-value = 0.53
## alternative hypothesis: true common odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.86699 1.33311
## sample estimates:
## common odds ratio
##            1.0751
```

```r
# Woolf-test on Homogeneity of Odds Ratios
vcd::woolf_test(table(dif.data$Gender, dif.data$V23, dif.data$deciles))
```

```
##
##  Woolf-test on Homogeneity of Odds Ratios (no 3-Way assoc.)
##
## data:  table(dif.data$Gender, dif.data$V23, dif.data$deciles)
## X-squared = 63.3, df = 4, p-value = 5.8e-13
```

1.5 The Rasch Model

The Rasch model (Rasch 1960) is mathematically equivalent to the 1PL IRT model, more generally a special case of a generalized linear model:

$$P(x = 1|\beta, \delta) = \frac{e^{(\beta - \delta_i)}}{1 + e(\beta - \delta_i)}$$

$$= ln\Big(\frac{P(\beta)}{1 - P(\beta)}\Big)$$

$$= \beta - \delta_i$$

where $P(x = 1|\beta, \delta)$ is the probability of correct response given latent trait ability, β, and item difficulty, δ.

Like the proportion correct responses in CTT (P) for the item difficulty, the correct number of responses is a sufficient statistic for β in the Rasch model. In other words, all examinees with the same score in the Rasch model have the same estimated ability (β) (Schumacker, 2004; Wright, 1997; Wright & Stone, 1979, p. 20). Whereas, in the 2PL and 3PL IRT models, examinees with the same

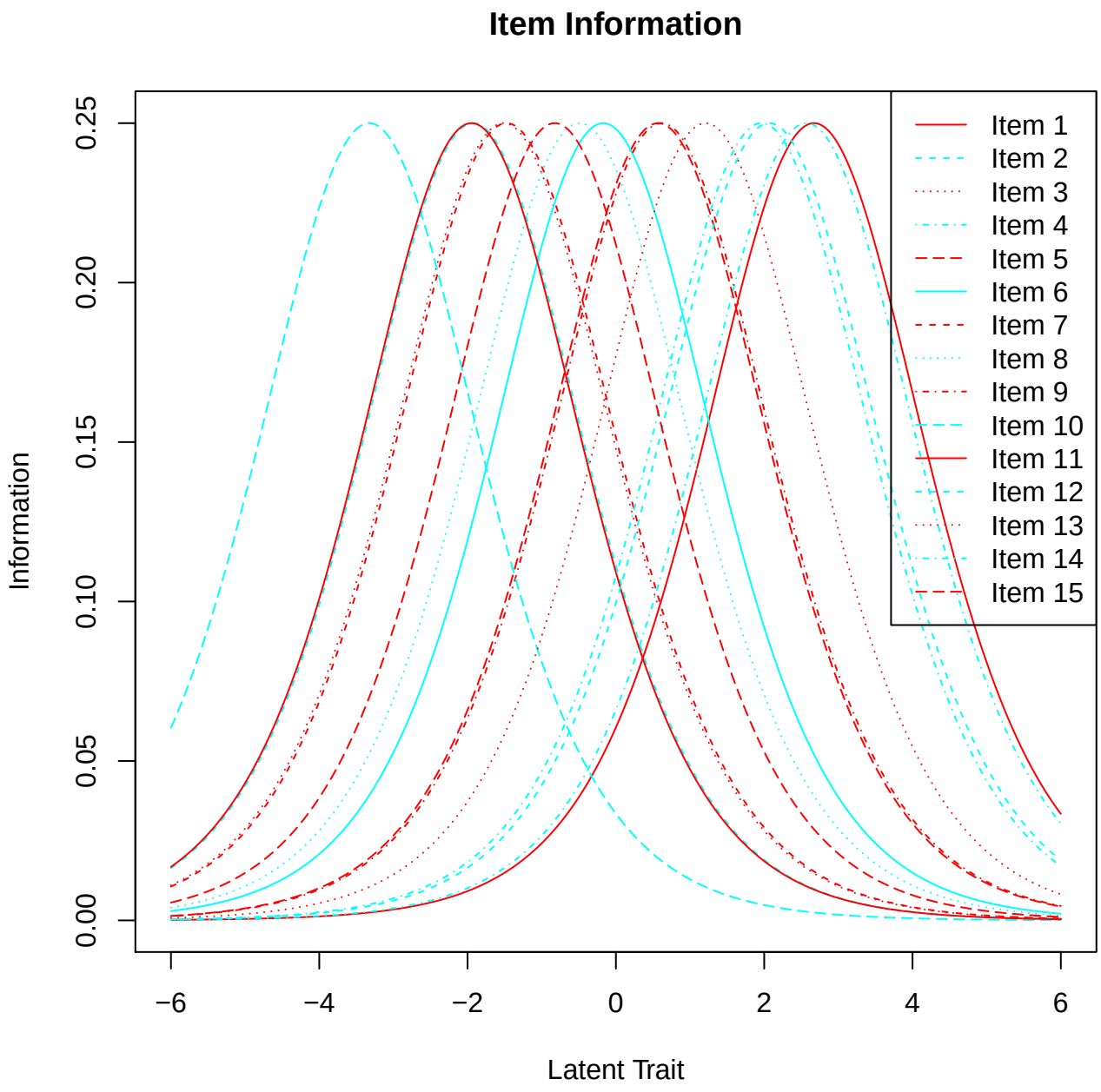
number of correct responses but with different patterns will have somewhat different estimates of β. Similarly, when the number of examinees responding correctly is the same for two items, then those items will have the same Rasch difficulty estimate, regardless of which examinees responded correctly (DeMars 2010).

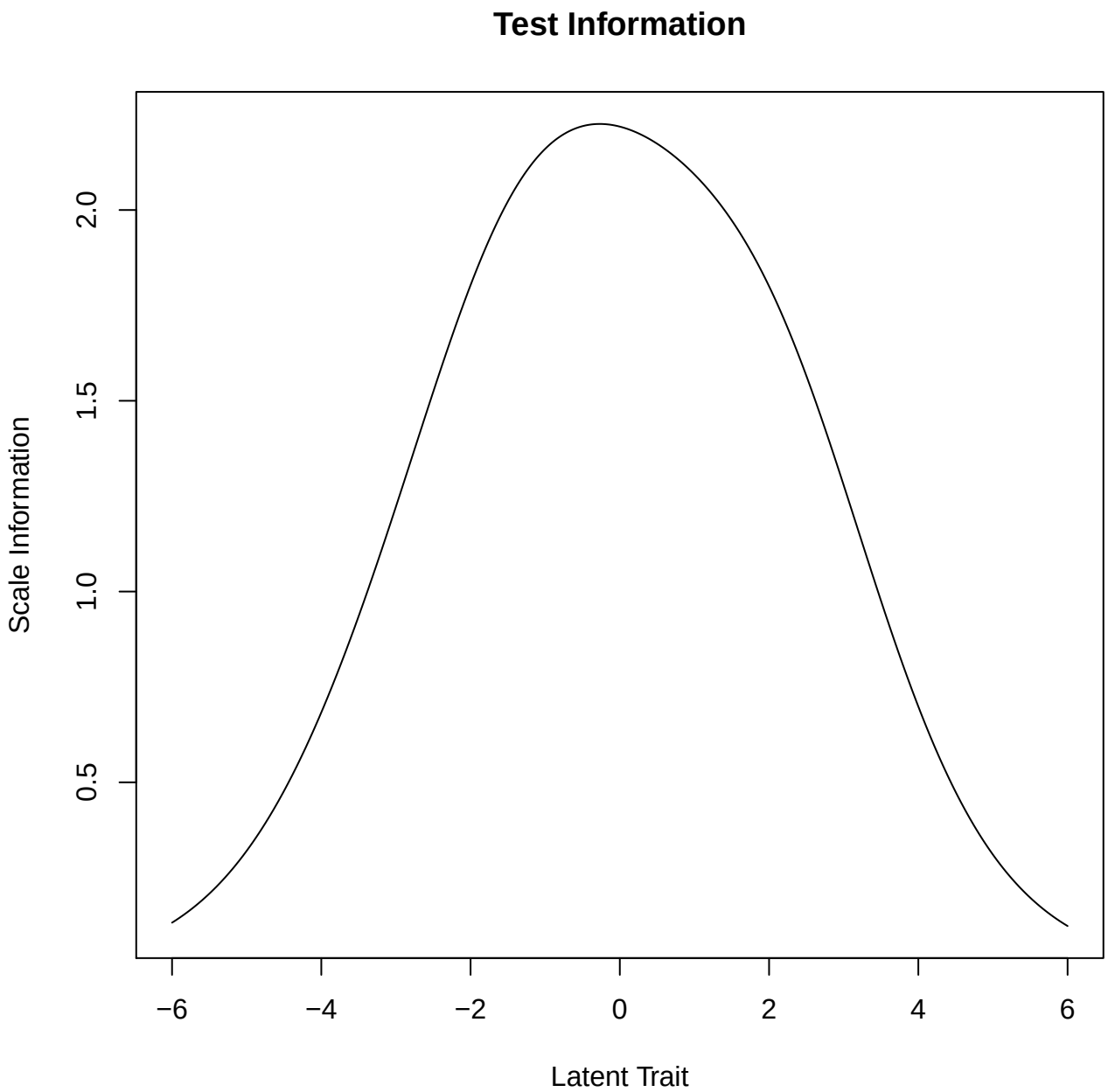
1.5.1 Estimation of Item Parameters

```
library(eRm)
Rasch <- eRm::RM(mytest, se = TRUE)
(delta <- Rasch$etapar) # item difficulty
```

```
##       V2       V3       V4       V5       V6       V7       V8       V9
## -1.92357 -1.51335  2.57022  0.57056 -0.17129 -1.47458 -0.48924  0.61552
##      V10      V11      V12      V13      V14      V15
## -3.32610 -1.94774  2.06894  1.20968  1.96351 -0.82398
```

```
# plot
eRm::plotINFO(Rasch, type = "item") # information plots
```


Item Information
Information
0.00
0.05
0.10
0.15
0.20
0.25
−6
−4
−2
0
2
4
6
Latent Trait
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9
Item 10
Item 11
Item 12
Item 13
Item 14
Item 15




```
eRm::plotjointICC(Rasch, legend = FALSE) # ICC curves
```

1.5.2 Maximum Likelihood Estimation of the Person Parameters

```
(beta <- eRm::person.parameter(Rasch))  # person parameters
```

```
##
## Person Parameters:
##
```

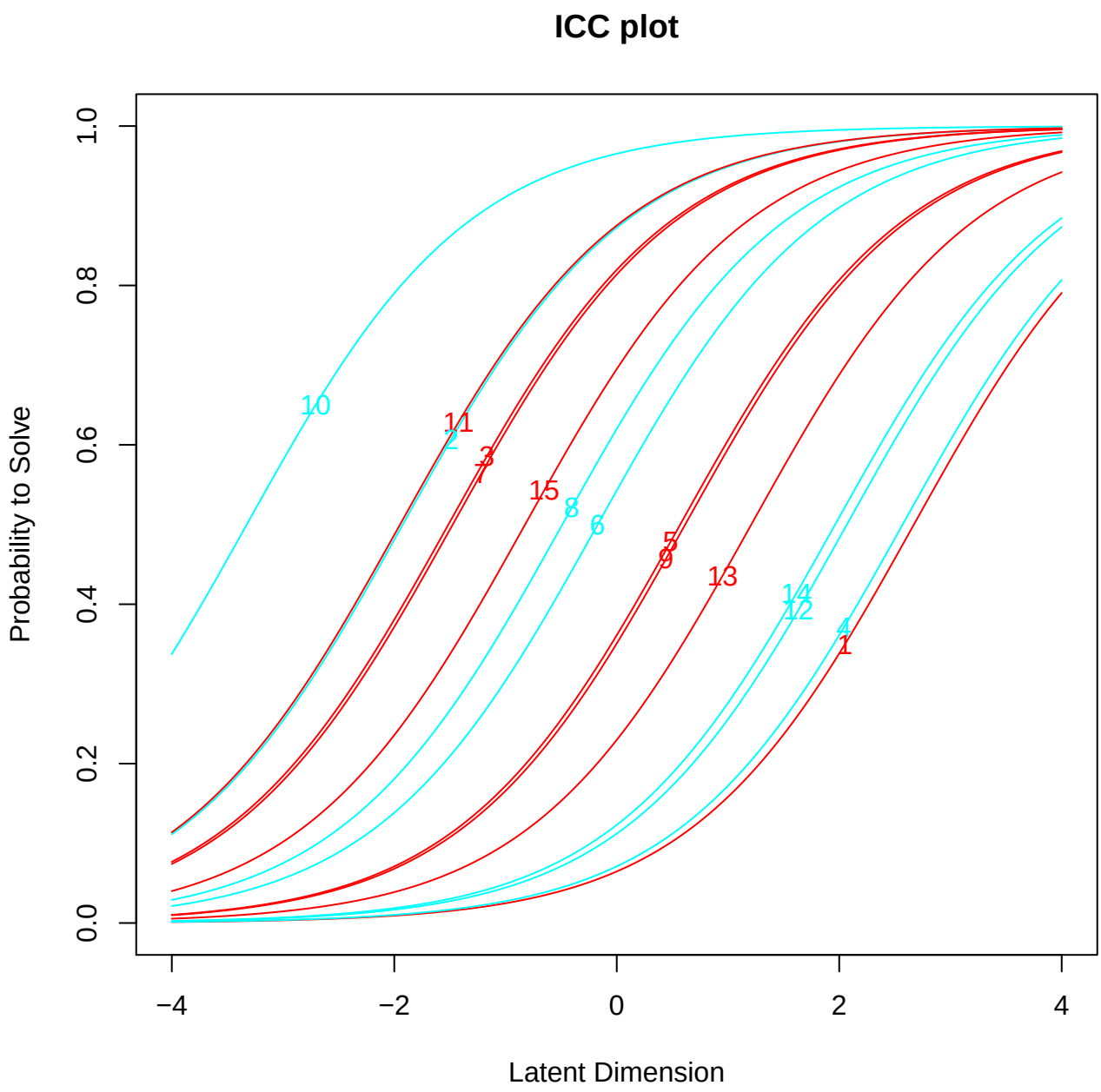

ICC plot
Probability to Solve
0.0
0.2
0.4
0.6
0.8
1.0
−4
−2
0
2
4
Latent Dimension
10
11
2
3
7
15
8
6
5
9
13
14
12
4
1

Figure 30:

```
##  Raw Score Estimate Std.Error
##          1 -3.77861   1.12536
##          2 -2.82610   0.86801
##          3 -2.16961   0.76400
##          4 -1.62902   0.71186
##          5 -1.14314   0.68526
##          6 -0.68297   0.67327
##          7 -0.23247   0.67034
##          8  0.21849   0.67365
##          9  0.67744   0.68213
##         10  1.15185   0.69655
##         11  1.65238   0.72052
##         12  2.19996   0.76393
##         13  2.84533   0.85410
##         14  3.75727   1.09934
```

```r
# log-likelihood of person parameter estimation
(log.likelihood <- stats::logLik(beta))
```

```
## Unconditional (joint) log Lik.: -86.828 (df=14)
```

```r
# Information Criteria
(eRm::IC(beta)) #log-likelihood, AIC, BIC
```

```
##
## Information Criteria:
##                        value npar    AIC    BIC   cAIC
## joint log-lik       -2638.1   28 5332.3 5447.3 5475.3
## marginal log-lik    -3158.7   14 6345.3 6402.9 6416.9
## conditional log-lik -2069.3   14 4166.6 4224.1 4238.1
```

1.5.3 Itemfit Statistics

```r
# itemfit statistics
(fit_of_items <- eRm::itemfit(beta))
```

```
##
## Itemfit Statistics:
##       Chisq  df p-value Outfit MSQ Infit MSQ Outfit t Infit t
## V1   226.26 449   1.000      0.503     0.726    -2.31   -3.47
## V2   274.08 449   1.000      0.609     0.763    -1.88   -2.62
## V3   234.45 449   1.000      0.521     0.761    -3.05   -3.12
## V4   349.65 449   1.000      0.777     0.858    -0.91   -1.78
## V5  2105.43 449   0.000      4.679     2.290    21.43   21.12
## V6   276.52 449   1.000      0.614     0.738    -4.52   -5.55
## V7   391.12 449   0.977      0.869     0.898    -0.68   -1.27
## V8   271.72 449   1.000      0.604     0.754    -4.13   -4.69
## V9   272.42 449   1.000      0.605     0.725    -4.82   -6.74
## V10  253.62 449   1.000      0.564     0.694    -0.92   -1.95
## V11  218.94 449   1.000      0.487     0.738    -2.63   -2.90
## V12  411.40 449   0.898      0.914     0.944    -0.41   -0.85
## V13  418.88 449   0.843      0.931     0.931    -0.55   -1.46
## V14  250.77 449   1.000      0.557     0.733    -2.96   -4.70
## V15  286.39 449   1.000      0.636     0.783    -3.16   -3.63
```

```r
# MSQ
MSQ <- cbind(rownames(fit_of_items),
             outfitMSQ = fit_of_items$i.outfitMSQ,
             infitMSQ = fit_of_items$i.infitMSQ)

# Sort decreasing out MSQ
head(MSQ[order(abs(fit_of_items$i.outfitMSQ), decreasing = TRUE),], 5)
```

```
##     outfitMSQ infitMSQ
## V5    4.67874  2.28981
## V13   0.93083  0.93076
## V12   0.91422  0.94421
## V7    0.86916  0.89793
## V4    0.77701  0.85780
```

```r
# Sort decreasing in MSQ
head(MSQ[order(abs(fit_of_items$i.infitMSQ), decreasing = TRUE),], 5)
```

```
##     outfitMSQ infitMSQ
## V5    4.67874  2.28981
## V12   0.91422  0.94421
## V13   0.93083  0.93076
## V7    0.86916  0.89793
## V4    0.77701  0.85780
```

1.5.4 Person Fit Statistics

```r
#Personfit
fit_of_persons <- eRm::personfit(beta)

person_fit <- data.frame(outMSQ = fit_of_persons$p.outfitMSQ,
                         inMSQ = fit_of_persons$p.infitMSQ,
                         outZ = fit_of_persons$p.outfitZ,
                         inZ = fit_of_persons$p.infitZ)

# pull out just those VERY largest values using MSQ and then sort
misfit_persons <- subset(person_fit,
                         subset = (person_fit$outMSQ > 4 |
                                   person_fit$inMSQ > 4 |
                                   abs(person_fit$outZ) > 4 |
                                   abs(person_fit$inZ) > 4))

# Sort decreasing out MSQ
misfit_persons[order(abs(misfit_persons$outMSQ), decreasing = TRUE),]
```

```
##      outMSQ   inMSQ   outZ      inZ
## P70  7.1720 4.25848 4.7749  5.27815
## P208 6.1596 0.89829 3.1767 -0.16385
## P111 5.9872 1.93888 1.9054  1.62972
## P15  5.2477 1.50072 1.7545  0.81773
## P65  5.2477 1.50072 1.7545  0.81773
## P123 5.2477 1.50072 1.7545  0.81773
## P172 5.2477 1.50072 1.7545  0.81773
## P210 5.2477 1.50072 1.7545  0.81773
```

```
## P249 5.2477 1.50072 1.7545  0.81773
## P243 4.1594 1.18583 2.8453  0.59988
```

```r
# Sort increasin in MSQ
misfit_persons[order(abs(misfit_persons$inMSQ), decreasing = TRUE),]
```

```
##      outMSQ   inMSQ   outZ      inZ
## P70  7.1720 4.25848 4.7749  5.27815
## P111 5.9872 1.93888 1.9054  1.62972
## P15  5.2477 1.50072 1.7545  0.81773
## P65  5.2477 1.50072 1.7545  0.81773
## P123 5.2477 1.50072 1.7545  0.81773
## P172 5.2477 1.50072 1.7545  0.81773
## P210 5.2477 1.50072 1.7545  0.81773
## P249 5.2477 1.50072 1.7545  0.81773
## P243 4.1594 1.18583 2.8453  0.59988
## P208 6.1596 0.89829 3.1767 -0.16385
```

1.5.5 Comparison of The Item Characteristic Curves for Items:

- V10 very easy item
- V6 average difficulty
- V5 average difficulty but negatively coded
- V1 very difficult item

```r
eRm::plotICC(Rasch,
             empICC = list("raw", type = "b", col = "tomato", lty = 1),
             item.subset = c(10, 5, 6, 1),
             ask = FALSE,
             empCI = list(gamma = 0.95, col = "dodgerblue", lty = 1),
             col = "green",
             lwd = 3)
```

1.5.6 Model Fit

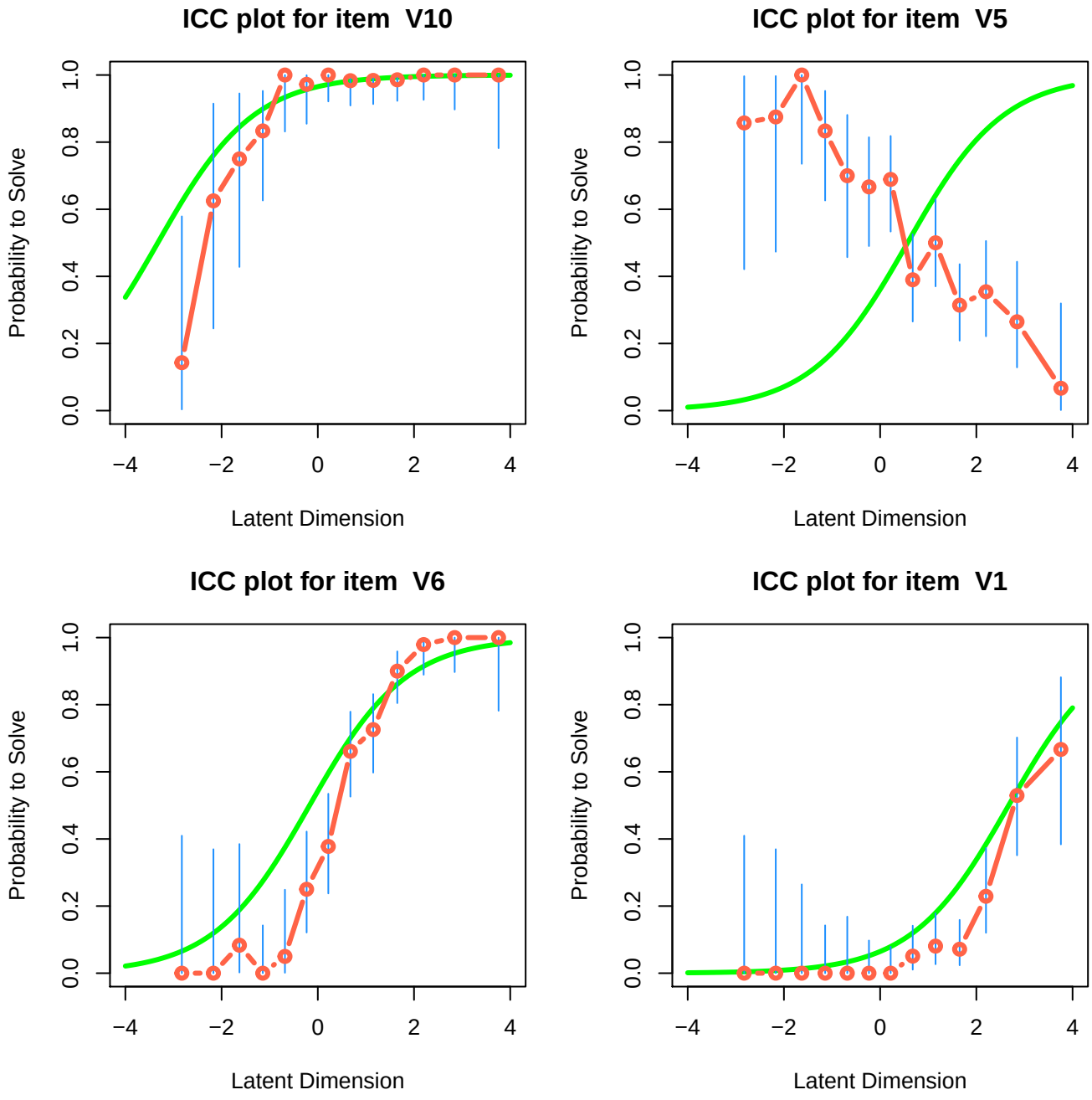

ICC plot for item V10
ICC plot for item V5
ICC plot for item V6
ICC plot for item V1
Probability to Solve
Latent Dimension
0.0
0.2
0.4
0.6
0.8
1.0
−4
−2
0
2
4

Figure 31:

```r
# Goodness-of-Fit Tests
(gof.res <- eRm::gofIRT(beta))
```

```
##
## Goodness-of-Fit Results:
## Collapsed Deviance = 956 (df = 210, p-value = 0)
## Pearson R2: 0.499
## Area Under ROC: 0.905
```

```r
summary(gof.res)
```

```
##
## Goodness-of-Fit Tests
##                        value    df p-value
## Collapsed Deviance  955.997   210       0
## Hosmer-Lemeshow      42.007     8       0
## Rost Deviance      1652.285 32753       1
## Casewise Deviance  5276.261  6722       1
##
## R-Squared Measures
## Pearson R2: 0.499
## Sum-of-Squares R2: 0.498
## McFadden R2: 0.754
##
## Classifier Results - Confusion Matrix (relative frequencies)
##          observed
## predicted     0     1
##         0 0.347 0.081
##         1 0.092 0.480
##
## Accuracy: 0.826
## Sensitivity: 0.855
## Specificity: 0.79
## Area under ROC: 0.905
## Gini coefficient: 0.809
```

```r
# LR-test on dichotomous Rasch model with user-defined split
(npersons <- nrow(mytest))
```

```
## [1] 450
```

```r
splitvec <- rep(1:2, each = (npersons/2))
LR <- eRm::LRtest(Rasch, splitcr = splitvec, se = TRUE)
summary(LR)
```

```
##
## Andersen LR-test:
## LR-value: 21.383
## Chi-square df: 14
## p-value:  0.092
##
##
## Subject Subgroup: splitvec 1:
## Log-likelihood:  -1087.3
##
## Beta Parameters:
##           beta V1 beta V2 beta V3  beta V4  beta V5 beta V6 beta V7
## Estimate -2.55876 1.87395 1.43133 -2.32592 -0.31564 0.21637 1.25568
## Std.Err.  0.20085 0.20627 0.18629  0.18872  0.14833 0.15345 0.17967
##          beta V8  beta V9 beta V10 beta V11 beta V12 beta V13 beta V14
## Estimate 0.46625 -0.64868  2.92533  1.74386  -2.0223 -0.96216  -1.8705
## Std.Err. 0.15782  0.14816  0.27771  0.19987   0.1756  0.15027   0.1701
##          beta V15
## Estimate  0.79121
## Std.Err.  0.16531
##
##
## Subject Subgroup: splitvec 2:
## Log-likelihood:  -971.31
##
## Beta Parameters:
##          beta V1 beta V2 beta V3  beta V4  beta V5 beta V6 beta V7 beta V8
```

```
## Estimate -2.8206 1.96040 1.58708 -2.86304 -0.85751 0.10634  1.7185 0.49627
## Std.Err.  0.2060 0.22305 0.20261  0.20831  0.15430 0.15859  0.2093 0.16530
##          beta V9 beta V10 beta V11 beta V12 beta V13 beta V14 beta V15
## Estimate -0.6045  3.99490  2.18111 -2.15117 -1.49850 -2.09111  0.84185
## Std.Err.  0.1538  0.44615  0.23737  0.17712  0.16101  0.17519  0.17394
```

```r
# Computation of Andersen's LR-test with mean split
(lrres.rasch <- LRtest(Rasch, splitcr = "mean"))
```

```
##
## Andersen LR-test:
## LR-value: 402.59
## Chi-square df: 14
## p-value:  0
```

```r
# We expect the difference to be close to zero
LR.matrix <- cbind(high = lrres.rasch$etalist$high,
                   low = lrres.rasch$etalist$low,
                   dif = lrres.rasch$etalist$high -
                         lrres.rasch$etalist$low)

LR.matrix
```

```
##         high       low      dif
## V2  -2.27315 -1.923722 -0.34943
## V3  -2.97297 -1.391662 -1.58131
## V4   2.89481  2.493805  0.40100
## V5   2.44486 -1.462141  3.90700
## V6  -0.36566  0.032572 -0.39824
## V7  -1.14709 -1.583065  0.43598
## V8  -1.14709 -0.226234 -0.92085
## V9   0.54898  1.104688 -0.55571
## V10 -2.97297 -3.434553  0.46158
## V11 -2.97297 -1.895872 -1.07710
## V12  2.48451  1.636600  0.84791
## V13  1.52177  1.104688  0.41708
## V14  2.12569  2.790355 -0.66466
```

```
## V15 -1.06352 -0.742050 -0.32147
```

```r
# potentially problematic items
# (the difference between high and low is larger than 1)
subset(LR.matrix, abs(LR.matrix[, "dif"]) > 1)
```

```
##        high     low     dif
## V3  -2.9730 -1.3917 -1.5813
## V5   2.4449 -1.4621  3.9070
## V11 -2.9730 -1.8959 -1.0771
```

1.5.7 Goodness of Fit Plot

```r
# color scheme
numitems <- ncol(mytest)
mycolors <- rainbow(numitems)

# Goodness of Fit Plot
eRm::plotGOF(lrres.rasch,
             set_par = TRUE,
             tlab = "number",
             col = mycolors,
             type = "p",
             ctrline = list(gamma = 0.95,
                            col = "black",
                            lty = 2,
                            cex = .5),
             conf = list(ia = FALSE,
                         col = mycolors,
                         lty = 3,
                         cex = .5))

# Item 5 tracing lines
abline(h = lrres.rasch$etalist$high[4],
       v = lrres.rasch$etalist$low[4],
```

```
        col = mycolors[5],
        lty = 3)
```

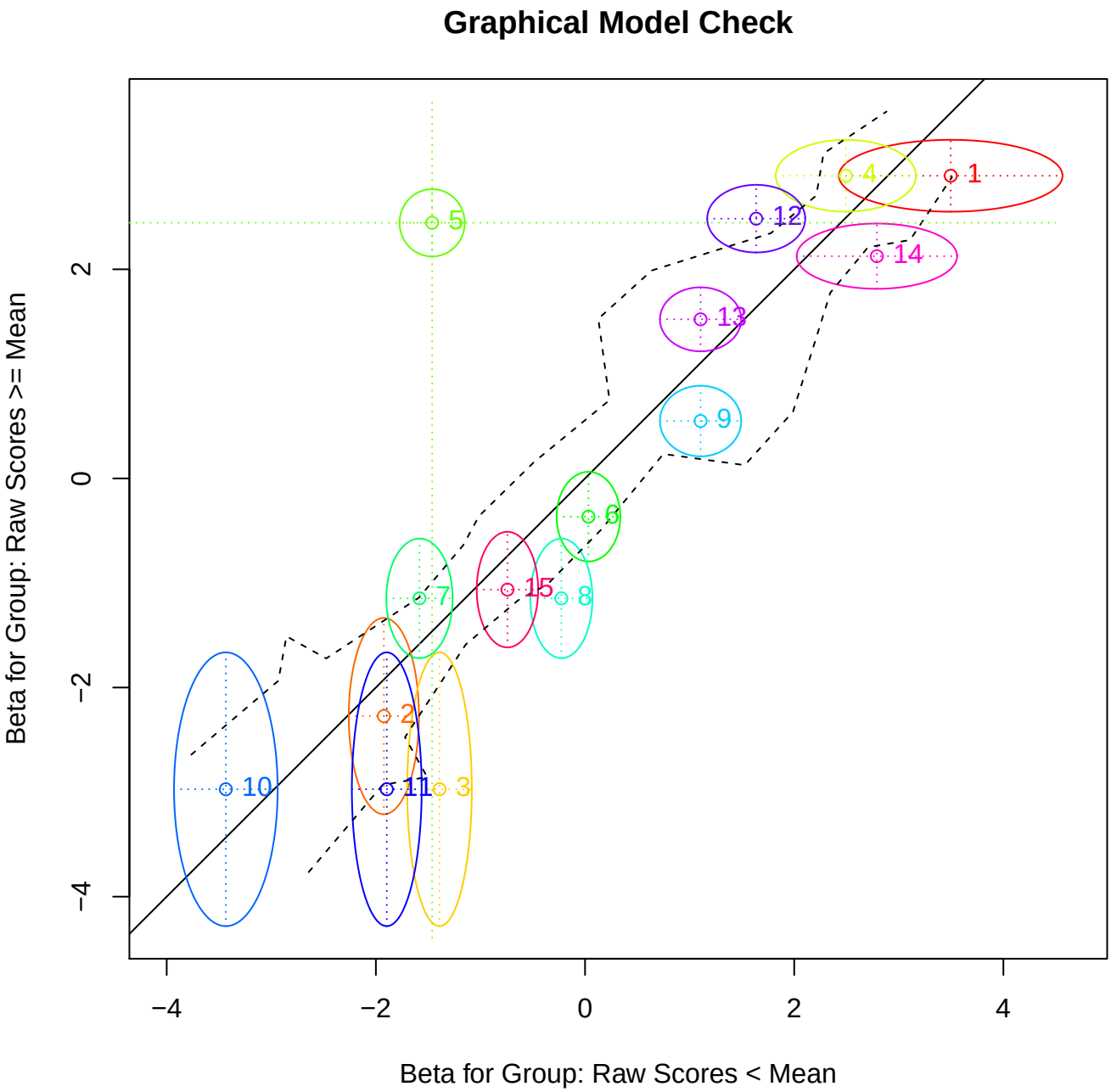


Figure 32:

Ideally, we would like to see a range of items difficulties over the latent trait continuum (-3, 3) without any gaps. Items' confidence ellipses should not significantly overlap (i.e. an items confidence ellipse should not include another items center). However, the confidence ellipses should include the black diagonal split line between groups; and their center should be within the black-dotted 95% confidence line.

For example, there is a large gap between items 6 and 9 (theta range about 0 to 1), and similary items 10 and 2. Confidence ellipses for items 2 and 11 include each others center, so they do about the same job as they cover the same region the of the latent trait continuum with item 2 doing a little bit better because its center is in the 95% confidence line and extending on both sides of the diagonal split line. Similarly items 9 and 13 have the same theta value. On the other hand, despite the fact that the confidence ellipse of item 1 includes the center of item 4, they cover different sections of the latent trait continuum (about theta 2.5 to 3.5). Also, there are issues with items 3, 5, 8, 9, 12 and 14 according to the above criteria, but item 5 clearly stands out. For the group, which has a raw score that is above the mean, the item-difficulty for 5 is about 2.5, where as for the group which has a raw score that is below the mean, item difficulty for it is -1.5. So the difficulty parameter of item 5 is clearly not invariant across the latent trait continuum. It may have been a negatively coded item as its ICC plot above also suggests.

1.6 Parameter Invariance

IRT parameters assumed to be invariant, that is IRT parameter metrics are interchangeable within a linear transformation. The degree to which item invariance is realized in practice is contingent on model-data fit, therefore can be used as part of model-data fit investigation (Ayala 2013, 61; DeMars 2010, 8). A strong Pearson correlation coefficient between item parameters that are calibrated from two random sets subsamples provides evidence for model data-fit.

$$\delta_i = c_1\delta_j + c_2$$

Function: Given a response matrix, split the data matrix into two random halves in terms of cases

```
SPLIT.CASES <-
  function(X, seed=NULL){
    # to produce the same instance of randomization process
    if (!is.null(seed)) {set.seed(seed)}

    X <- data.frame(X)

    # if n = 2x, then lengths Y1 = Y2
    # if n = 2x+1, then lenths Y1 = Y2+1
    n <- nrow(X)
```

```
    index <- sample(1:n, ceiling(n/2))
    Y1 <- X[index, ]
    Y2 <- X[-index, ]

    return(list(data.matrix(Y1), data.matrix(Y2)))
  } # end SPLIT.CASES

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

# returns a list of two response matrices
P <- SPLIT.CASES(mytest, seed = 1234)
```

Function: Given a response matrix and model (Rasch, 1PL, 2PL, or 3PL), computes item difficulty correlations and graphs item invariance plot.

```
invariance.plot <-
  function(test, model = NULL, seed = NULL) {
    source("SPLIT.CASES.R")

    # to produce the same instance of randomization process
    if (!is.null(seed)) {set.seed(seed)}

    mytest <- as.matrix(test)
    numitems <- ncol(mytest)

    # returns a list of two response matrices
    P <- SPLIT.CASES(mytest, seed = seed)

    if (model == "Rasch") {
      require(eRm)

      # Rasch model P
      PP <- eRm::RM(mytest, se = FALSE)

      # Rasch model P1
      P1 <- eRm::RM(P[[1]], se = FALSE)
```

```
    # Rasch model P2
    P2 <- eRm::RM(P[[2]], se = FALSE)

    # item difficulties
    deltas <- cbind(PP = PP$etapar, P1 = P1$etapar, P2 = P2$etapar)
  }
  else if (model == "1PL" | model == "2PL" | model == "3PL") {
    require(irtoys)

    # IRT P
    PP <- irtoys::est(mytest, model = model, engine = "ltm")

    # IRT P1
    P1 <- irtoys::est(P[[1]], model = model, engine = "ltm")

    # IRT P2
    P2 <- irtoys::est(P[[2]], model = model, engine = "ltm")

    # item difficulties
    deltas <- cbind(PP = PP$est[, 2], P1 = P1$est[, 2], P2 = P2$est[, 2])
  }
  else {return("Model has to be one of Rasch, 1PL, 2PL, or 3PL")}

  # Plot
  #-------------------------------------------------
  plot(deltas[,"PP"], deltas[,"P1"],
       pch = 16,
       cex = 2,
       lwd = .5,
       type = "p",
       col = adjustcolor(rainbow(nrow(deltas)), alpha.f = 0.5),
       ylab = "Subsamples Item Difficulties",
       xlab = "Whole Group Item Difficulties",
       main = "Item Invariance",
       xlim = c(-4, 4),
```

```
         ylim = c(-4, 4))

    points(deltas[,"PP"], deltas[,"P2"],
           pch = 1,
           cex = 2.1,
           lwd = 2,
           col = adjustcolor(rainbow(nrow(deltas)), alpha.f = 0.5))

    legend("topleft",
           c("Whole group vs subsample 1", "Whole group vs subsample 2"),
           col = "gray",
           bty = "n",
           text.col = "gray",
           pch = c(1, 16))

    return(cor(deltas))
  } # end invariance.plot

dump("invariance.plot", file = "invariance.plot.R")

invariance.plot(mytest, model = "Rasch", seed = 12345)
```

```
##         PP      P1      P2
## PP 1.00000 0.99637 0.99606
## P1 0.99637 1.00000 0.98492
## P2 0.99606 0.98492 1.00000
```

```
invariance.plot(mytest, model = "1PL", seed = 12345)
```

```
##         PP      P1      P2
## PP 1.00000 0.99649 0.99633
## P1 0.99649 1.00000 0.98569
## P2 0.99633 0.98569 1.00000
```

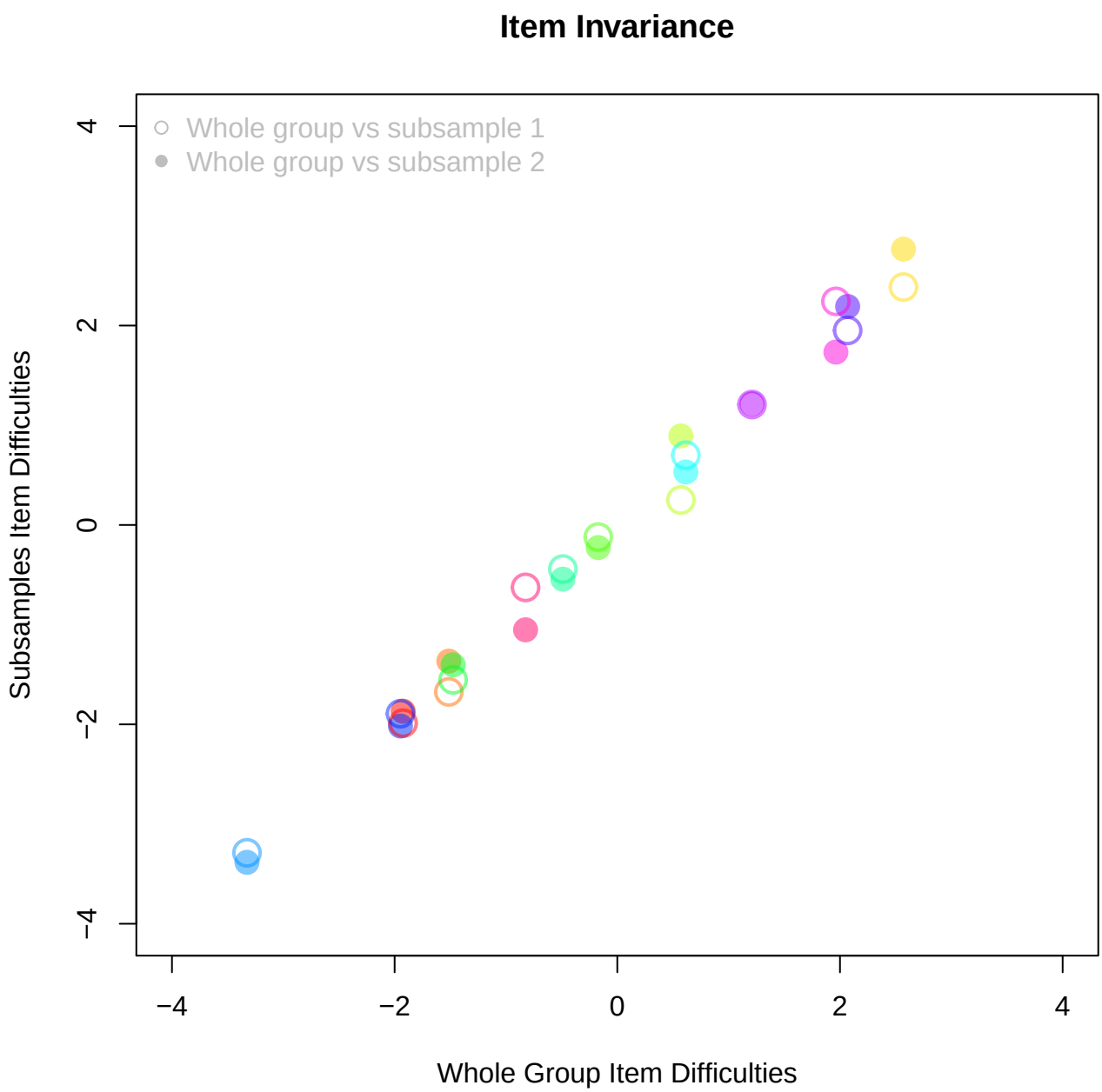

Item Invariance
Whole group vs subsample 1
Whole group vs subsample 2
Subsamples Item Difficulties
Whole Group Item Difficulties
−4
−2
0
2
4

Figure 33:

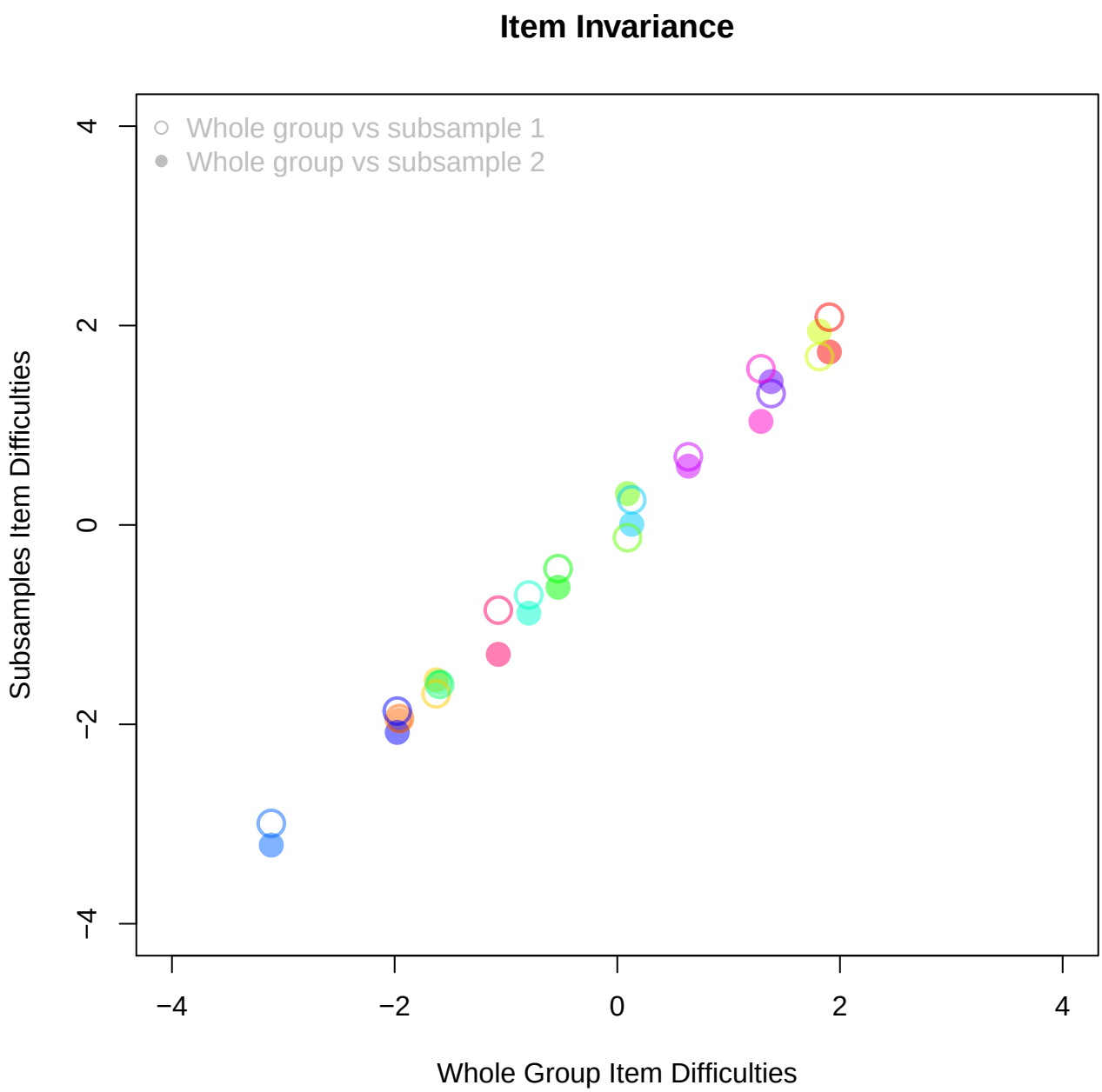

Item Invariance
Whole group vs subsample 1
Whole group vs subsample 2
Subsamples Item Difficulties
Whole Group Item Difficulties
−4
−2
0
2
4

Figure 34:

```r
invariance.plot(mytest, model = "2PL", seed = 12345)
```

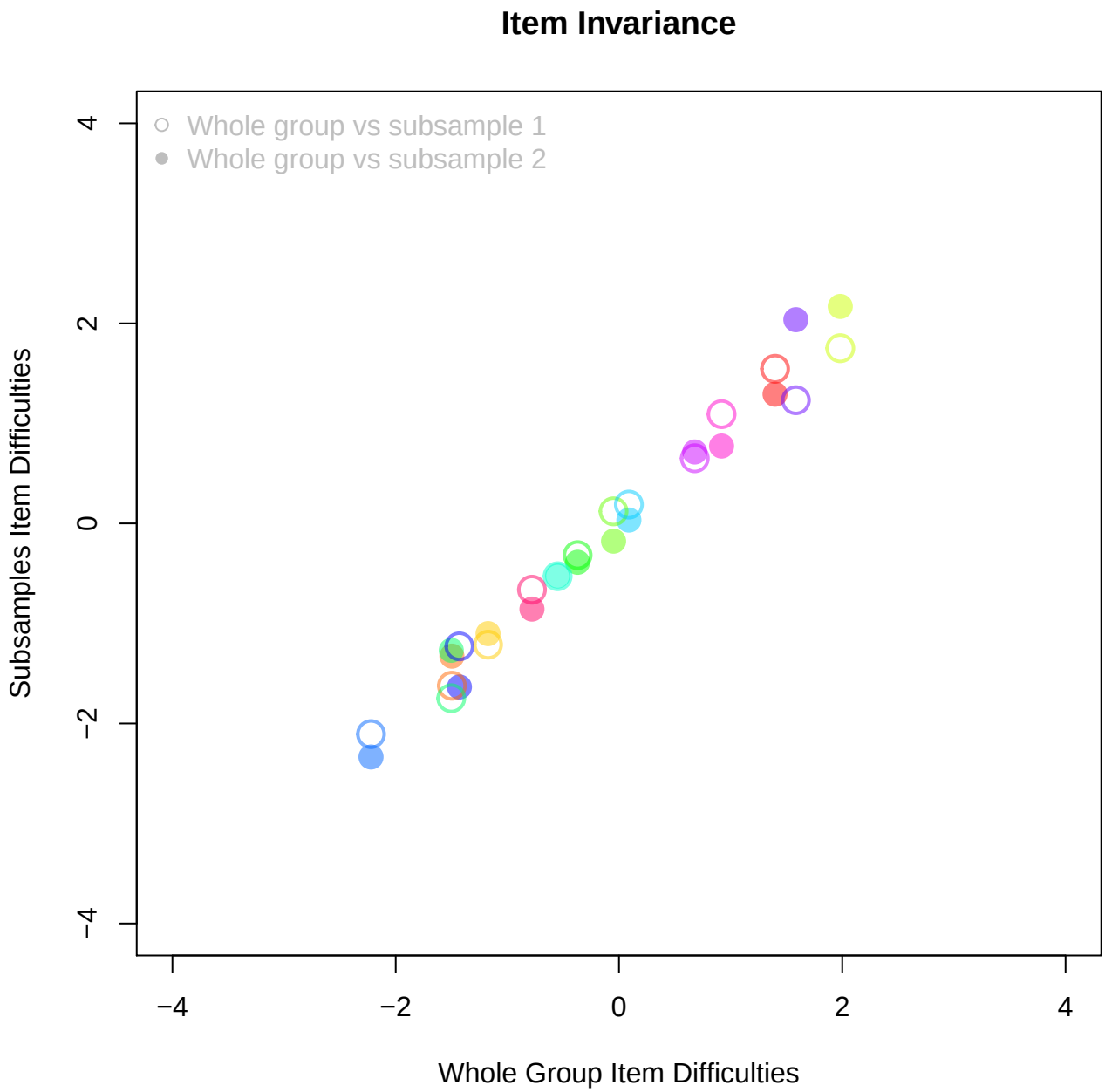


Figure 35:

```
##          PP      P1      P2
## PP 1.00000 0.99185 0.99101
## P1 0.99185 1.00000 0.96620
## P2 0.99101 0.96620 1.00000
```

The item difficulty (δ) estimates from both random subsamples of examinees (P1, P2) correlate very

strongly (r = .985) and with the estimates from the entire group (both r = .996). There is a strong evidence that the item difficulties are invariance across subsamples.

1.6.1 Root Mean Squared Difference (RMSD)

Correlation coefficients for the item parameter estimates do not take the interaction between item parameters into account. The root-mean-square difference (RMSD) computes the area between two ICCs and reflects the interaction between the item parameter estimates in 2PL and 3Pl models. Prior to RMSD computation, subsample metrics should be linked as they are scale dependent. $RMSD_i$ statistics for item i is given by

$$RMSD_j = \sqrt{\frac{\sum(p_{js} - p_{jt})^2}{n}}$$

where n is the number of Θ's used in computing p_{js} and p_{jt}'s. For example, if the range of Θ is (-4, 4) in 0.01 increments, then n = 801.

RMSD has a range of 0 to 1 (inclussive) with small values indicating good agreement between the two IRFs.

```
RMSD <-
  function(responses, seed = NULL, model = NULL){
    source("IRF.R")
    source("SPLIT.CASES.R")
    require(irtoys)

    # to produce the same instance of randomization process
    if (!is.null(seed)) {set.seed(seed)}

    # default model
    if (is.null(model)) {model <- "2PL"}

    resp <- as.matrix(responses)

    # CONSTANTS
    K <- ncol(resp) # Number of items
    N <- nrow(resp) # Number of examinees
```

```
    # returns a list of two response matrices
    P <- SPLIT.CASES(resp, seed = seed)

    rmsd <- rep(NA, K)

    # length of latent variable continium
    n <- length(seq(from = -4, to = 4, by = 0.01))

    # item parameter estimates
    P1 <- irtoys::est(resp = P[[1]],
                      model = model,
                      engine = "ltm")

    # probability of a correct response
    Pt <- IRF(parameter.matrix = P1$est)

    # item parameter estimates
    P2 <- irtoys::est(resp = P[[2]],
                      model = model,
                      engine = "ltm")

    # probability of a correct response
    Ps <- IRF(parameter.matrix = P2$est)

    # compute RMSD
    for (i in 1:K) {
      rmsd[i] <- sqrt(sum((Pt$probabilities[, i] -
                              Ps$probabilities[,i])^2)/n)
    }

    return(rmsd)
  }# end RMSD

dump("RMSD", file = "RMSD.R")
```

```r
rmsd <- RMSD(mytest, model = "2PL")
summary(rmsd)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 3.12e-05 4.25e-04 6.09e-04 8.88e-04 1.38e-03 2.35e-03
```

Largest RMSD value is less than 0.01, which is a small value. So, there isn't a significant difference between the two subsets of the 2PL model

References

Albano, A. 2015. *Observed-Score Linking and Equating.* R Package Version 2.0-3. 2014.

Ayala, R. J. de. 2013. *Theory and Practice of Item Response Theory Methodology in the Social Sciences.* New York, NY: Guilford Publications.

Braun, H. I., and P. W. Holland. 1982. *Observed-Score Test Equating: A Mathematical Analysis of Some ETS Equating Procedures. in P. W. Holland & d. B. Rubin (Eds.), Test Equating.* New York: Academic.

DeMars, C. 2010. *Item Response Theory.* New York: Oxford University Press.

Holland, P. W., and D. T. Thayer. 2000. "Univariate and Bivariate Loglinear Models for Discrete Test Score Distributions." *Journal of Educational and Behavioral Statistics* 25: 133–83.

Kolen, M. J. 1984. "Effectiveness of Analytic Smoothing in Equipercentile Equating." *Journal of Educational Statistics* 9: 25–44.

Kolen, M. J., and R. L. Brennan. 2013. *Test Equating, Scaling, and Linking.* 3rd ed. New York: Springer.

Livingston, & Kim, S. A. 2009. "The Circle-Arc Method for Equating in Small Samples." *Journal of Educational Measurement* 46: 330–43.

Rasch, G. 1960. *Probabilistic Models for Some Intelligence and Attainment Tests.* Copenhagen: Danish Institute for Educational Research.