Teoria da Resposta ao Item com uso do R

Adilson dos Anjos [1]

Dalton Francisco de Andrade [2]

João Pessoa, PB
30 de julho a 3 de agosto de 2012

[1]Professor do Departamento de Estatística da UFPR e Doutorando do Programa de Pós-graduação em Engenharia de Produção da UFSC - aanjos@ufpr.br

[2]Professor Voluntário do Programa de Pós-graduação em Engenharia de Produção da UFSC - dandrade@inf.ufsc.br

Prefácio

A ideia de elaborar esse texto surgiu durante o curso de Teoria da Resposta ao Item, ministrado no Programa de Pós-graduação em Engenharia de Produção da UFSC. O **R** por ser livre e por utilizar uma linguagem de programação permite ao usuário desenvolver suas próprias funções, de acordo com as suas necessidades. Além disso, possui uma variedade de métodos estatísticos implementados que podem ser úteis em análises de testes.

No primeiro capítulo, são apresentados alguns conceitos iniciais sobre Testes Clássicos, Teoria da Resposta ao Item e sobre o **R** . Não é a proposta do curso fundamentar os participantes na utilização de aspectos básicos do **R** . Presume-se que o participante já tenha conhecimentos básicos do software antes de iniciar a leitura desse texto.

No segundo capítulo, é apresentado, de forma breve, um exemplo da aplicação de alguns métodos utilizados na Teoria Clássica dos Testes, com dados provenientes de um questionário sobre a altura de respondentes.

No terceiro capítulo, são apresentados dois exemplos de utilização de modelos dicotômicos unidimensionais de dois e três parâmetros. Para o modelo de dois parâmetros são utilizados os dados do questionário sobre Altura. Para exemplificar a utilização de um modelo de três parâmetros foram utilizados dados do SARESP - Sistema de Avaliação de Rendimento Escolar do Estado de São Paulo, gentilmente fornecidos pela Secretaria da Educação do Estado de São Paulo..

No quarto capítulo, são utilizados dados do SARESP para mostrar uma

forma de equalização de testes.

No quinto capítulo, são apresentados alguns exemplos de simulação de respostas para modelos dicotômicos com uso do **R** .

Por fim, gostaríamos de agradecer à ABE a oportunidade de apresentar esse curso no 20^o SINAPE, ao Professor Masanao Ohira do Laboratório de Estatística Aplicada da UFSC (LEA), Juliana de Caldas Rosa e os professores: Pedro Alberto Barbetta, Paulo José Ogliari, Antonio Cezar Bornia e Heliton Ribeiro Tavares que colaboraram de alguma forma para a elaboração deste texto.

Adilson dos Anjos
Dalton Francisco de Andrade

Sumário

Capítulo 1

Introdução

O desenvolvimento de escalas apropriadas para medir características de indivíduos que não podem ser medidas diretamente, as quais são comumente denominadas de traço latente, tem tomado a atenção de pesquisadores das mais diferentes áreas do conhecimento. Exemplos de tais características: nível de qualidade de vida, proficiência em matemática, grau de depressão, usabilidade de sites de e-commerce na web, nível de raciocínio diagnóstico de profissionais de enfermagem etc.

Duas são as teorias utilizadas para este fim. A Teoria Clássica dos Testes – TCT, que utiliza o escore no teste como sua referência de medida, e a Teoria da Resposta ao Item – TRI, cujo foco principal, como bem diz o seu nome, é o item e não o teste como um todo. Ambas contemplam a análise de itens através das estimativas de seus parâmetros, e a análise do instrumento de medida como um todo. A TRI foi desenvolvida com o propósito de resolver um problema da TCT que é a dependência da medida de proficiência em relação ao teste aplicado e dos parâmetros dos itens em relação ao conjunto dos respondentes. Dentro do contexto da TRI, a medida de proficiência de um aluno não depende dos itens apresentados a ele, e os parâmetros de discriminação e de dificuldade do item não dependem do grupo de respondentes. Em outras palavras, um item mede determinado conhecimento, independentemente de quem o está respondendo, e a proficiência de um aluno não depende dos itens que estão

sendo apresentados a ele.

A aplicação destas teorias, em particular a TRI, exige a utilização de recursos computacionais específicos que estão disponibilizados em vários programas. O nosso foco, neste trabalho, será a apresentação dos recursos disponíveis no **R** .

Maiores detalhes sobre estas duas teorias podem ser encontrados em: Gulliksen (1950), Lord e Novick (1968), Lord (1980), Vianna (1987), Pasquali (2003), Andrade, Tavares e Valle (2000), Ayala (2009), Baker e Kim (2004) e Embretson e Reise (2000).

1.1 Teoria clássica dos testes (TCT)

Na teoria clássica dos testes, além do número total de acertos podem ser utilizadas algumas medidas para se avaliar a qualidade do instrumento de medida. Algumas dessas medidas, que podem ser obtidas com o uso do **R** , são: o coeficiente de correlação ponto-bisserial, o coeficiente de correlação bisserial e o coeficiente alfa de Cronbach.

1.1.1 Coeficiente de correlação ponto-bisserial

O coeficiente de correlação ponto-bisserial (ρ_{pb}) é a correlação de Pearson entre uma variável dicotômica e o escore do teste e é definido por:

$$\rho_{pb} = \frac{\bar{X}_A - \bar{X}_T}{S_T}\sqrt{\frac{p}{1-p}} \tag{1.1}$$

em que,

$\bar{X}_A$ é a média dos escores dos respondentes que acertaram o item;
$\bar{X}_T$ é a média global dos escores do teste;
S_T é o desvio padrão do teste;
p é a proporção de respondentes que acertaram o item.

1.1.2 Coeficiente de correlação bisserial

O coeficiente de correlação bisserial (ρ_b) é uma medida de associação entre uma variável dicotomizada e uma variável contínua, e é definido por:

$$\rho_b = \rho_{pb}\frac{\sqrt{p(1-p)}}{h(p)} \tag{1.2}$$

em que,

ρ_{pb} é a correlação ponto-bisserial;
p é a proporção de respondentes que acertaram o item;
$h(p)$ é o valor da densidade da distribuição normal padrão no ponto em que a área da curva à esquerda deste ponto é igual a p.

1.1.3 Coeficiente alfa de Cronbach

O coeficiente alfa de Cronbach é utilizado para medir a consistência interna do instrumento de medida, e é definido por:

$$\alpha = \frac{n}{n-1}(1 - \frac{\sum s_i^2}{s_T^2}) \tag{1.3}$$

em que,

n é o número de itens;
$\sum s_i^2$ é a soma das variâncias dos n itens;
s_T^2 é a variância global dos escores dos testes.

Esse coeficiente varia de 0 a 1. Quanto mais próximo de 0 menor a consistência e quanto mais próximo de 1 maior a consistência do teste.

1.2 Teoria da Resposta ao Item

Nesse texto, serão apresentadas algumas análises, considerando alguns modelos logísticos unidimensionais da TRI. Entre esses modelos estão os modelos de 1, 2 e 3 parâmetros. O modelo de 1 parâmetro é também referido como modelo de Rasch.

1.2.1 Modelo de 3 parâmetros

O modelo logístico de 3 parâmetros é definido por:

$$P(U_{ij} = 1|\theta_j, a_i, b_i, c_i) = c_i + (1 - c_i)\frac{e^{a_i(\theta_j - b_i)}}{1 + e^{a_i(\theta_j - b_i)}} \tag{1.4}$$

em que,

$P(U_{ij} = 1|\theta_j, a_i, b_i, c_i)$ é a probabilidade do indivíduo j com habilidade θ_j acertar o item i;
b_i é o parâmetro de dificuldade (ou de posição) do item i, medido na mesma escala de habilidade;
a_i é o parâmetro de discriminação (ou inclinação) do item i, com valor proporcional à inclinação da Curva Característica do Item no ponto b_i;
c_i é o parâmetro do item que representa a probabilidade de indivíduos com baixa habilidade responderem corretamente o item i (também chamado de probabilidade de acerto casual).

1.2.2 Modelo de 2 parâmetros

O modelo de 2 parâmetros é semelhante ao modelo de 3 parâmetros, mas não inclui o parâmetro de acerto casual no modelo.

$$P(U_{ij} = 1|\theta_j, a_i, b_i) = \frac{e^{a_i(\theta_j - b_i)}}{1 + e^{a_i(\theta_j - b_i)}} \tag{1.5}$$

1.2.3 Função de informação do item

A função de informação do item (*item information function*) permite analisar o quanto um item contém de informação sobre a medida de habilidade. Ela indica a quantidade de informação que um item apresenta dentro da escala de habilidade. A função de informação do item é definida como:

$$I_i(\theta) = \frac{[\frac{d}{d\theta}P_i(\theta)]^2}{P_i(\theta)Q_i(\theta)} \tag{1.6}$$

em que,

$I_i(\theta)$ é a informação fornecida pelo item i no nível de habilidade θ;
$P_i(\theta) = P(X_{ij} = 1|\theta)$;
$Q_i(\theta) = 1 - P_i(\theta)$.

Para um modelo logístico unidimensional de 3 parâmetros, a função de informação do item pode ser escrita como:

$$I_i(\theta) = D^2 a_i^2 \frac{Q_i(\theta)}{P_i(\theta)}[\frac{P_i(\theta) - c_i}{1 - c_i}]^2 \tag{1.7}$$

A Equação 1.7 apresenta como os parâmetros dos itens se relacionam com a quantidade de informação. A informação é maior quando:

1. b_i se aproxima de (θ);
2. quanto maior for a_i;
3. quanto mais c_i se aproximar de 0.

Para um modelo logístico de 2 parâmetros a equação 1.7 pode ser escrita da seguinte forma:

$$I(\theta) = a_i^2 P_i(\theta)(Q_i(\theta))$$

E para um modelo com 1 parâmetro,

$$I(\theta) = P_i(\theta)(Q_i(\theta))$$

1.2.4 Função de informação do teste

Dada a independência entre os itens, a função de informação do teste (*teste information function*) é a soma das informações fornecidas por cada item, que foram calibrados em uma mesma escala. A função de informação do teste é escrita como:

$$I(\theta) = \sum_{i=1}^{I} I_i(\theta)$$

1.2.5 Equalização

Equalização é o procedimento para 'ajustar' a medida de habilidade (θ) entre grupos de indivíduos submetidos a diferentes testes com itens em comum para uma mesma métrica (KOLEN; BRENNAN, 2010). Basicamente, o objetivo da equalização é tornar a medida de habilidade comparável.

Existem vários métodos utilizados para a equalização. No **R** estão implementados os métodos de equalização a *posteriori*, ou seja, após a calibração em separado dos itens nos diferentes grupos de respondentes.

Com dois grupos, o objetivo é encontrar um conjunto de constantes para transformar a escala de um grupo ($G1$) na mesma métrica do outro ($G2$). Assim, é possível obter-se θ_{G1} da seguinte forma:

$$\theta_{G1} = A\theta_{G2} + B \tag{1.8}$$

em que,

θ_{G1} é a habilidade do Grupo 1 (transformada);

θ_{G2} é a habilidade do Grupo 2;
A e B são constantes a serem estimadas.

Portanto, o objetivo da equalização a *posteriori* é encontrar A e B, de modo a transformar a escala de habilidade de um grupo para que essa possa ser comparada com a do outro.

Os métodos Média/média e Média/desvio são métodos onde as estimativas de A e B podem ser obtidas por meio de regressões lineares simples tendo em vista que as estimativas são obtidas por fórmulas específicas e não por mínimos quadrados. As estimativas de A e B são obtidas a partir de itens comuns entre os testes.

Os métodos de Haebara, e Stocking e Lord são baseados nas curvas características dos itens e são procedimentos computacionalmente iterativos.

Os quatro métodos implementados no pacote `plink` do **R** são:

1. Média/média (mean/mean):

$$A = \frac{\mu(a_{G2})}{\mu(a_{G1})}$$

e

$$B = \mu(b_{G1}) - A\mu(b_{G2})$$

2. Média/desvio (mean/sigma):

$$A = \frac{\sigma(a_{G2})}{\sigma(a_{G1})}$$

e

$$B = \mu(b_{G1}) - A\mu(b_{G2})$$

3. Haebara:

No método proposto por Haebara, obtém-se a soma do quadrado das diferenças entre a curva característica do item para cada respondente i:

$$Hdif(\theta_i) = \sum_{j:v} \left[p_{ij}(\theta_{Ji}; \hat{a}_{Jj}, \hat{b}_{Jj}, \hat{c}_{Jj}) - p_{ij}(\theta_{Ji}; \frac{\hat{a}_{Ij}}{A}, A\hat{b}_{Ji} + B, \hat{c}_{Ji}) \right]^2$$

em que,

$j : v$ são os itens em comum;

O processo de estimação consiste em encontrar A e B que minimiza o seguinte critério:

$$H_{crit} = \sum_i Hdif(\theta_i).$$

4. Stocking e Lord

   No método proposto por Stocking e Lord, obtém-se o quadrado das diferenças das somas entre a curva característica do item para cada respondente i:

$$SLdif(\theta_i) = \left[\sum_{j:v} p_{ij}(\theta_{Ji}; \hat{a}_{Jj}, \hat{b}_{Jj}, \hat{c}_{Jj}) - \sum_{j:v} p_{ij}(\theta_{Ji}; \frac{\hat{a}_{Ij}}{A}, A\hat{b}_{Ji} + B, \hat{c}_{Ji}) \right]^2$$

   O processo de estimação consiste em encontrar A e B que minimiza o seguinte critério:

$$SL_{crit} = \sum_i SLdiff(\theta_i).$$

Mais detalhes sobre a teoria, outros modelos, métodos de estimação podem ser vistos em (ANDRADE; TAVARES; VALLE, 2000), disponível em `http://www.inf.ufsc.br/~dandrade/TRI/LivroTRI_pdf.zip`.

1.3 Programa R

O **R** pode ser obtido no seguinte endereço: `http://cran.r-project.org/`. Existem versões do **R** para os sistemas operacionais Windows, Linux e Mac. Escolha a versão, baixe o arquivo de instalação e siga as instruções.

Na página do **R** você poderá encontrar dezenas de documentos sobre como utilizá-lo. Na internet também existem muitos materiais disponíveis em centenas de páginas.

1.3.1 Recursos do R para Psicometria

O **R** possui milhares de pacotes (*packages*) disponíveis. Alguns desses pacotes foram agrupados em função de áreas em comum. Esses agrupamentos são chamados de *Task Views* e estão disponíveis no site do **R** (`http://cran.r-project.org/`).

Em `http://cran-r.c3sl.ufpr.br/web/views/` há um conjunto de pacotes organizados na área de Psicometria chamado *Psychometrics* que pode ser acessado em `http://cran-r.c3sl.ufpr.br/web/views/Psychometrics.html`

Se for do seu interesse, é possível baixar todos os pacotes listados em Psicometria de uma só vez. Primeiro instale o pacote `ctv`. Em seguida, instale os pacotes da área de interesse.

```
> install.packages("ctv")
> library(ctv)
> install.views("Psychometrics")
```

Para instalar um pacote, utilize a função `install.packages('nomedopacote')` (com aspas).

Para utilizar o pacote, utilize a função `library(nomedopacote)` (sem aspas).

Em vários pacotes do **R** existem conjuntos de dados (datasets) disponíveis, que são utilizados nos exemplos de utilização de funções. Para saber quais os *datasets* instalados em seu computador utilize `data()`. Para utilizar dados de algum pacote, digite `data(nomedodataset)`.

Nesse texto, foram utilizados os seguintes pacotes do **R** :

1. `irtoys` (PARTCHEV, 2010);
2. `ltm` (RIZOPOULOS, 2006)
3. `Deducer` (FELLOWS, 2012)
4. `plink` (WEEKS, 2010)
5. `CTT` (WILLSE; SHU, 2008)

As análises foram realizadas com a seguinte versão do **R** :

```
[1] "R version 2.14.2 (2012-02-29)"
```

Sugere-se a utilização do software RStudio como interface do software **R** . Entre na página `www.rstudio.org` e baixe a versão compatível com seu sistema operacional.

1.4 Arquivos de dados utilizados

Neste texto, serão utilizados 2 conjuntos de dados:

1. Dados **Altura**

   Descrição: Questionário com 14 itens e altura em metros de 211 respondentes. Respostas dicotômicas.
   Utilização: Teoria clássica dos testes e modelo logístico unidimensional de 2 parâmetros.

Disponível em:
`http://www.ufpr.br/~aanjos/TRI/sinape/dados/altura211.dat`
Fonte: Dalton Francisco de Andrade e Antonio Cezar Bornia (Laboratório de custos e medidas da UFSC)

2. Dados **SARESP**

   Descrição: Uma amostra de 3 testes (manhã, tarde e noite) de Língua Portuguesa aplicados para alunos do terceiro ano do ensino médio em 2007.
   Utilização: Modelo logístico unidimensional de 3 parâmetros e equalização a *posteriori* de dois grupos.
   Disponível em:
   `http://www.ufpr.br/~aanjos/TRI/sinape/dados/saresp.dat`
   Fonte: Secretaria Estadual da Educação de São Paulo.

Capítulo 2

Teoria Clássica dos Testes

Neste capítulo, será mostrado como realizar uma análise clássica utilizando algumas funções dos pacotes `ltm` e `CTT` do **R** .

2.1 Exemplo: Dados Altura

O instrumento de medida apresentado na Tabela 2.1 refere-se ao questionário sobre Altura com 14 itens. O objetivo desse questionário é obter, com um modelo da TRI, uma estimativa da altura das pessoas em função das suas respostas.

Esse questionário foi respondido por 211 pessoas, que também forneceram a informação sobre sua altura em metros (Item 15).

Tabela 2.1: Questionário com itens para estimar a altura de pessoas.

Item	**Descrição (pergunta): Assinale 1 para 'sim' e 0 para 'não'.**
1	Na cama, eu frequentemente sinto frio nos pés.
2	Eu frequentemente desço as escadas de dois em dois degraus.
3	Eu acho que me daria bem em um time de basquete.
4	Como policial eu impressionaria muito.
5	Na maioria dos carros eu me sinto desconfortável.
6	Eu literalmente olho para meus colegas de cima para baixo
7	Você é capaz de pegar um objeto no alto de um armário sem usar escada?
8	Você abaixa quando vai passar por uma porta?
9	Você consegue guardar a bagagem no porta-malas do avião?
10	Você regula o banco do carro para trás?
11	Normalmente, quando você está andando de carona, lhe oferecem o banco da frente?
12	Quando você e várias outras pessoas vão tirar fotos, formando-se três fileiras, onde ninguém ficará agachado, você costuma ficar atrás?
13	Você tem dificuldade para se acomodar no ônibus?
14	Em uma fila, por ordem de tamanho, você é sempre colocado atrás?
15	Qual a sua altura em metros?

2.1.1 Leitura do arquivo

O arquivo `Altura211.dat` possui o seguinte formato:

```
  11,8101011110111010
  21,6400000000100000
  31,8000110010110101
  41,7801111010110101
  51,6600000010110000
  61,6700000010110000
.............................
2061,6510000000000000
2071,9200010011111111
2081,6310000000000000
2091,6010000000000000
2101,5700000000000000
2111,5800000000000000
```

A primeira coluna fornece o número do respondente e pode ter até 3 algarismos. Em seguida, vem a altura em metros com 4 campos, e os últimos 14 dígitos são as respostas aos 14 itens do questionário. Um arquivo nesse formato pode ser lido no **R** com a função `read.fwf()`. Observe, ainda, que não há um cabeçalho para as colunas e que utilizou-se ',' como separador decimal.

Os dados podem ser obtidos diretamente do site com os seguinte comandos:

```
> altura<-read.fwf(
+ 'http://www.ufpr.br/~aanjos/TRI/sinape/dados/altura211.dat',
+ widths=c(3,4,rep(1,14)),header=FALSE,dec=',')
```

A função `read.fwf()` com o argumento `widths=c(3,4,rep(1,14))` lê o arquivo considerando as 3 primeiras colunas como identificadores, em seguida a altura, com 4 campos, e cada uma das colunas seguintes como sendo um

item. O argumento `header=FALSE` indica que não há cabeçalho e o argumento `dec=','` indica que os decimais são separados por vírgula.

Parte dos dados pode ser visualizada com as funções `head()` e `tail()`:

```
> head(altura)
```

```
  V1  V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16
1  1 1.8  0  1  0  1  1  1  1   0   1   1   1   0   1   0
2  2 1.6  0  0  0  0  0  0  0   0   1   0   0   0   0   0
3  3 1.8  0  0  1  1  0  0  1   0   1   1   0   1   0   1
4  4 1.8  0  1  1  1  1  0  1   0   1   1   0   1   0   1
5  5 1.7  0  0  0  0  0  0  1   0   1   1   0   0   0   0
6  6 1.7  0  0  0  0  0  0  1   0   1   1   0   0   0   0
```

```
> tail(altura)
```

```
     V1  V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16
206 206 1.6  1  0  0  0  0  0  0   0   0   0   0   0   0   0
207 207 1.9  0  0  0  1  0  0  1   1   1   1   1   1   1   1
208 208 1.6  1  0  0  0  0  0  0   0   0   0   0   0   0   0
209 209 1.6  1  0  0  0  0  0  0   0   0   0   0   0   0   0
210 210 1.6  0  0  0  0  0  0  0   0   0   0   0   0   0   0
211 211 1.6  0  0  0  0  0  0  0   0   0   0   0   0   0   0
```

A função `colnames()` pode ser utilizada para colocar nomes nas colunas:

```
> colnames(altura)<-c('id','altura',paste('i',1:14,sep=""))
> # insere nomes nas colunas
> head(altura) # ver os 6 primeiros registros
```

```
  id altura i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14
1  1    1.8  0  1  0  1  1  1  1  0  1   1   1   0   1   0
2  2    1.6  0  0  0  0  0  0  0  0  1   0   0   0   0   0
```

```
3  3    1.8  0  0  1  1  0  0  1  0  1   1   0   1   0   1
4  4    1.8  0  1  1  1  1  0  1  0  1   1   0   1   0   1
5  5    1.7  0  0  0  0  0  0  1  0  1   1   0   0   0   0
6  6    1.7  0  0  0  0  0  0  1  0  1   1   0   0   0   0
```

O objeto `altura` é um `data.frame`:

```
> class(altura) # tipo de objeto
```

```
[1] "data.frame"
```

2.1.2 TCT com pacote ltm

Existem várias estatísticas que podem ser utilizadas para examinar um conjunto de respostas de um teste. Por exemplo, a correlação bisserial e o coeficiente alfa de Cronbach.

A função `descript()` do pacote `ltm` aplicada aos dados do objeto `altura` fornece os seguintes resultados:

```
> library(ltm)
```

Observe que no objeto `altura` os itens estão nas colunas 3 a 16. Veja, também, que o objeto `altura.desc` contém mais informações:

```
> altura.itens<-altura[,3:16] # utilizando apenas as colunas de respostas
> altura.desc<-descript(altura.itens)
> names(altura.desc)
```

```
 [1] "sample"    "perc"      "items"     "pw.ass"
 [5] "n.print"   "name"      "missin"    "data"
 [9] "bisCorr"   "ExBisCorr" "alpha"
```

```
> altura.desc


Descriptive statistics for the 'altura.itens' data-set

Sample:
 14 items and 211 sample units; 0 missing values

Proportions for each level of response:
       0     1 logit
i1  0.68 0.322 -0.74
i2  0.76 0.242 -1.14
i3  0.79 0.208 -1.33
i4  0.73 0.265 -1.02
i5  0.81 0.194 -1.42
i6  0.80 0.204 -1.36
i7  0.47 0.531  0.12
i8  0.92 0.081 -2.43
i9  0.33 0.673  0.72
i10 0.53 0.474 -0.10
i11 0.60 0.403 -0.39
i12 0.59 0.412 -0.35
i13 0.73 0.275 -0.97
i14 0.68 0.318 -0.77


Frequencies of total scores:
      0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
Freq 20 21 28 32 17 21 13 15 10 10  6  9  5  3  1


Point Biserial correlation with Total Score:
    Included Excluded
i1      0.25     0.11
i2      0.48     0.37
i3      0.54     0.45
```

```
i4      0.43     0.31
i5      0.62     0.54
i6      0.61     0.53
i7      0.70     0.61
i8      0.34     0.27
i9      0.47     0.35
i10     0.71     0.63
i11     0.54     0.43
i12     0.66     0.57
i13     0.63     0.54
i14     0.73     0.65


Cronbach's alpha:
              value
All Items      0.83
Excluding i1   0.84
Excluding i2   0.82
Excluding i3   0.82
Excluding i4   0.83
Excluding i5   0.81
Excluding i6   0.81
Excluding i7   0.80
Excluding i8   0.83
Excluding i9   0.82
Excluding i10  0.80
Excluding i11  0.82
Excluding i12  0.81
Excluding i13  0.81
Excluding i14  0.80


Pairwise Associations:
   Item i Item j p.value
1       1      2   0.982
```

```
2       8      9   0.975
3       1     11   0.740
4       1     10   0.707
5       1      5   0.632
6       1      9   0.585
7       2     11   0.522
8       1      4   0.395
9       1      7   0.315
10      1      8   0.274
```

Os resultados mostram a correlação ponto bisserial, o coeficiente alfa de Cronbach e as associações entre itens, além da frequência de respostas de cada item.

A correlação bisserial indica que os itens apresentam uma correlação alta com o escore total, exceto os itens 1 e 8 que apresentam uma correlação baixa. Também, não há itens que apresentam correlação negativa, indicando que não há itens com problemas de entendimento.

O coeficiente alfa de Cronbach para cada um dos itens individualmente, e considerando todos os itens simultaneamente, é sempre maior do 0,8, indicando que existe boa fidedignidade do instrumento.

2.1.3 Gráficos

Para avaliar o comportamento dos itens, podem ser apresentados alguns gráficos. Com os resultados da função `descript()` pode-se pedir para construir um gráfico com o uso da função `plot()`. Nesse gráfico (Figura 2.1) é apresentado o total dos *escores* em relação à proporção de respostas 1 (corretas) dos itens 1, 8 e 9, por exemplo. A opção `includeFirstLast=TRUE` indica que todos os escores devem ser inseridos no gráfico.

Por exemplo, para o item 9, dentre aqueles que obtiveram um escore 2 (soma das respostas 1 no questionário), cerca de 45% responderam sim para esse item. Na Figura 2.1, pode-se observar que o item 9 foi o que apresentou em média, a maior proporção de respostas corretas e o item 8 foi o que apresentou

a menor proporção de respostas corretas, enquanto o item 1 obteve valores intermediários. Veja as questões associadas com cada item na Tabela 2.1.

```
> plot(altura.desc,items=c(1,8:9),type="b",
+      includeFirstLast=TRUE,pch=c('1','8','9'))
```

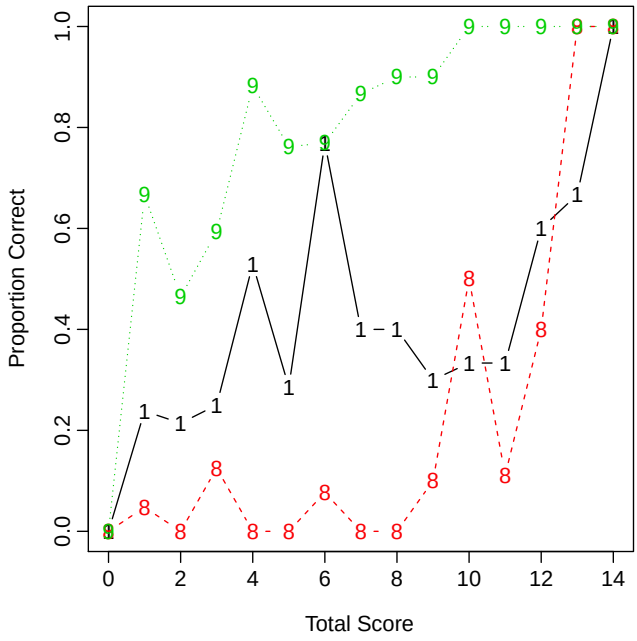

Figura 2.1: Gráfico do total dos escores e proporção de acertos para os itens 1, 8 e 9 do questionário sobre altura.

Experimente construir o gráfico com todos os itens:

```
> plot(altura.desc,type='b',includeFirstLast=TRUE)
```

2.1.4 Correlação bisserial

A correlação bisserial entre o escore total e um item pode ser obtida com a função `biserial.cor()`: Por exemplo, para o item 1:

```
> biserial.cor(rowSums(altura.itens), altura.itens[[1]])

[1] -0.25
```

Por padrão, a função `biserial.cor()` utiliza o valor 0 como referência. Para utilizar o valor 1 utilize o argumento `level=2`:

```
> biserial.cor(rowSums(altura.itens), altura.itens[[1]],level=2)

[1] 0.25
```

Compare com a saída da função `descript()` e observe o sinal da correlação. O valor é o mesmo, mas o sinal é diferente. Isso se deve ao fato de que a função considera o primeiro nível das respostas para obter o valor da correlação.

2.1.5 Coeficiente alfa de Cronbach

A função `cronbach.alpha()` fornece o valor do coeficiente alfa de Cronbach entre os itens.

```
> cronbach.alpha(altura.itens)

Cronbach's alpha for the 'altura.itens' data-set

Items: 14
Sample units: 211
alpha: 0.83
```

Para obter o mesmo resultado da função `descript()`, utilize da seguinte forma:

```
> cronbach.alpha(altura.itens[-1])  # exclui o item 1

Cronbach's alpha for the 'altura.itens[-1]' data-set

Items: 13
Sample units: 211
alpha: 0.84
```

2.1.6 TCT com pacote CTT

Além do pacote `ltm`, há também o pacote `CTT` que pode ser utilizado para obtenção de algumas estatísticas de interesse na análise clássica de testes.

Inicialmente, carregue o pacote `CTT`:

```
> library(CTT)
```

A função `reliability()` do pacote `CTT` pode ser utilizada para obter o coeficiente `Alpha de Cronbach` e outras estatísticas:

```
> altura.reliab<-reliability(altura.itens)
> names(altura.reliab)

[1] "N_item"           "N_person"         "alpha"
[4] "scale.mean"       "scale.sd"         "alpha.if.deleted"
[7] "pbis"             "item.mean"
```

Por exemplo, pode-se ter interesse na correlação ponto bisserial:

```
> altura.reliab$pbis
```

```
 [1] 0.11 0.37 0.45 0.31 0.54 0.53 0.61 0.27 0.35 0.63 0.43
[12] 0.57 0.54 0.65
```

Observe que essa correlação refere-se ao valor com a exclusão do item, na ordem em que são apresentados. Compare com os resultados da função `descript()`.

Para ver mais resultados da função `reliability()` use:

```
> str(altura.reliab)
```

```
List of 8
 $ N_item          : int 14
 $ N_person        : int 211
 $ alpha           : num 0.826
 $ scale.mean      : num 4.6
 $ scale.sd        : num 3.46
 $ alpha.if.deleted: num [1:14(1d)] 0.839 0.821 0.816 0.825 0.81 ...
 $ pbis            : num [1:14(1d)] 0.114 0.375 0.455 0.314 0.544 ...
 $ item.mean       : Named num [1:14] 0.322 0.242 0.209 0.265 0.194 ...
  ..- attr(*, "names")= chr [1:14] "i1" "i2" "i3" "i4" ...
 - attr(*, "class")= chr "reliability"
```

Capítulo 3

Teoria da Resposta ao Item

3.1 Exemplo: modelo de 2 parâmetros (Altura)

Nesse exemplo, serão utilizados os dados do questionário com itens sobre altura apresentado no Capítulo 2, Tabela 2.1. As análises serão apresentadas com uso dos pacotes `irtoys` e `ltm`.

3.1.1 Análise pelo pacote irtoys

O pacote `irtoys` (PARTCHEV, 2010) pode ser carregado da seguinte maneira:

```
> library(irtoys)
```

Calibração:

O pacote `irtoys` possui uma função chamada `est()` que é utilizada para a calibração dos itens de um teste. Para mais informações sobre a função `est()` veja o arquivo de ajuda do pacote: `help(est)`.

```
> library(irtoys)
```

Essa função apresenta mais opções do que a apresentada no pacote `ltm` (mas utiliza recursos do pacote `ltm`).

```
> altura.par<-est(altura.itens, model="2PL",
+                 engine="ltm",nqp= 20, logistic=TRUE)
```

O objeto `altura.par` contém os resultados da função `est` aplicada nos itens sobre altura. Nesse objeto são armazenados os resultados da calibração de um modelo com 2 parâmetros (a e b). A opção `nqp=20` indica o número de pontos de quadratura utilizados na estimação, e `logistic=TRUE` (padrão) indica que a função retornará os parâmetros na escala logística.

A opção `engine='ltm'` indica que a função `est` utilizará os recursos do pacote `ltm`. Outras opções são o 'BILOG' e 'ICL'. Observe que utilizou-se a opção `engine='ltm'`. Nesse caso, somente as opções listadas no exemplo são funcionais.

O objeto `altura.par` é uma matriz onde a primeira coluna contém os valores do parâmetro a (discriminação) de cada item e na segunda coluna o parâmetro b (dificuldade) de cada item:

```
> altura.par
```

```
    [,1]   [,2] [,3]
i1  0.30  2.519    0
i2  1.18  1.224    0
i3  1.47  1.244    0
i4  0.89  1.330    0
i5  2.21  1.091    0
i6  1.80  1.143    0
i7  4.09 -0.103    0
i8  1.11  2.642    0
i9  1.14 -0.788    0
i10 3.19  0.055    0
i11 1.16  0.427    0
```

```
i12 2.37  0.256    0
i13 1.74  0.829    0
i14 2.81  0.533    0
```

3.1.2 Gráficos com o pacote irtoys

O comportamento dos itens pode ser analisado por meio de gráficos, como a curva característica do item (Figura 3.1) e a curva de informação do item (Figura 3.2). Esses gráficos podem facilmente ser obtidos com algumas funções disponíveis no pacote `irtoys`.

A estimativa da habilidade $\hat{\theta}$ de cada um dos indivíduos pode ser obtida com a função `eap()`. O argumento `qu=normal.qu()` controla o número de pontos de quadratura no momento da estimação. É necessário fornecer o objeto que contém os parâmetros estimados dos itens. No exemplo, as estimativas dos parâmetros dos itens estão no objeto `altura.par`.

```
> altura.sco<-eap(altura.itens,altura.par,qu=normal.qu())
```

Os resultados da estimação da habilidade podem ser obtidos da seguinte maneira:

```
> head(altura.sco)
```

```
        est  sem  n
[1,]  0.884 0.35 14
[2,] -0.995 0.52 14
[3,]  0.721 0.32 14
[4,]  1.134 0.37 14
[5,]  0.016 0.28 14
[6,]  0.016 0.28 14
```

No objeto `altura.sco` a primeira coluna é a estimativa da habilidade de cada pessoa, a segunda coluna é o erro padrão e a terceira coluna o número de respostas de cada pessoa.

```
> plot(irf(altura.par),label=TRUE)
```

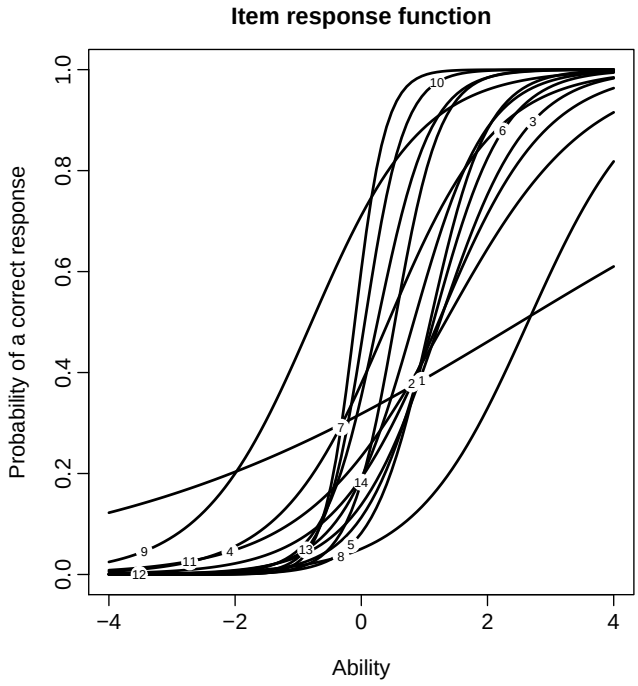

Figura 3.1: Curva característica dos itens para os dados sobre altura.

A curva de informação do teste (Figura 3.3) para os dados de Altura pode ser obtida com a função `tif` do pacote `irtoys`.

```
> plot(iif(altura.par),label=TRUE)
```

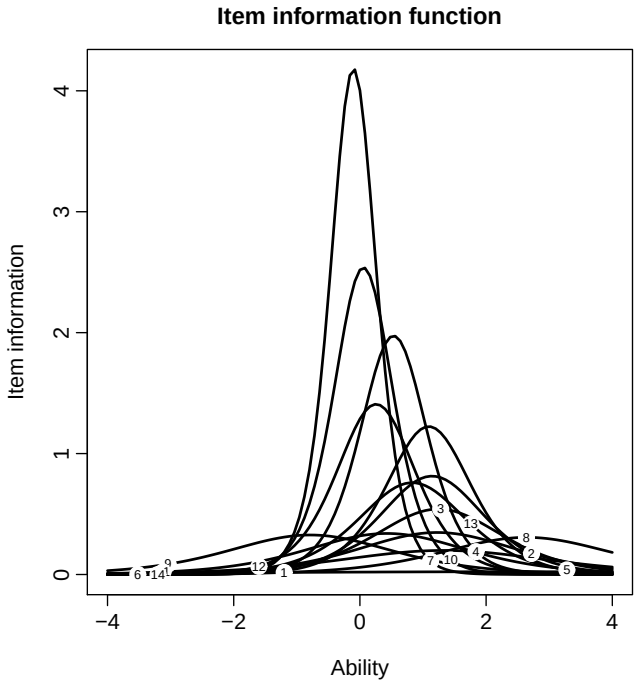

Figura 3.2: Curva de informação dos itens para os dados sobre altura.

```
> plot(tif(altura.par),label=TRUE)
```

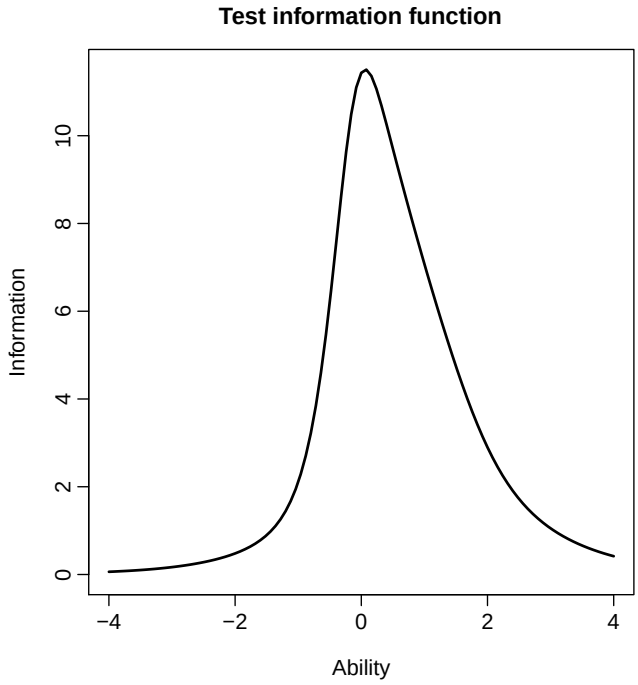

Figura 3.3: Curva de informação do teste para os dados sobre altura.

3.1.3 Estimando a altura

Dado um padrão de respostas, pode-se obter a altura em metros de uma pessoa. Um padrão de respostas poderia ser informado da seguinte maneira:

```
> adilson<-c(1,0,0,1,1,0,1,0,1,1,1,1,1,1)
> dalton<-c(0,0,0,0,0,0,1,0,1,1,0,1,0,0)
> resposta<-rbind(adilson,dalton)
> resposta
```

```
        [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
adilson    1    0    0    1    1    0    1    0    1     1
dalton     0    0    0    0    0    0    1    0    1     1
        [,11] [,12] [,13] [,14]
adilson     1     1     1     1
dalton      0     1     0     0
```

Estima-se o 'traço latente' de cada pessoa dado o seu padrão de respostas com a função `eap()`, utilizando os parâmetros estimados do modelo de 2 parâmetros armazenado no objeto `altura.par`:

```
> theta.resposta<-eap(resposta, altura.par,qu=normal.qu())
> theta.resposta
```

```
      est  sem  n
[1,] 1.21 0.38 14
[2,] 0.22 0.31 14
```

Como nesse exemplo o traço latente é conhecido (altura em metros) pode-se avaliar a correlação entre o traço latente estimado pelo conjunto de itens e a altura em metros (Figura 3.4). Considere que:

```
> x<-altura.sco[,1] # theta estimado de cada pessoa
> y<-altura$altura  # altura de cada pessoa
```

```
> plot(x,y)
```

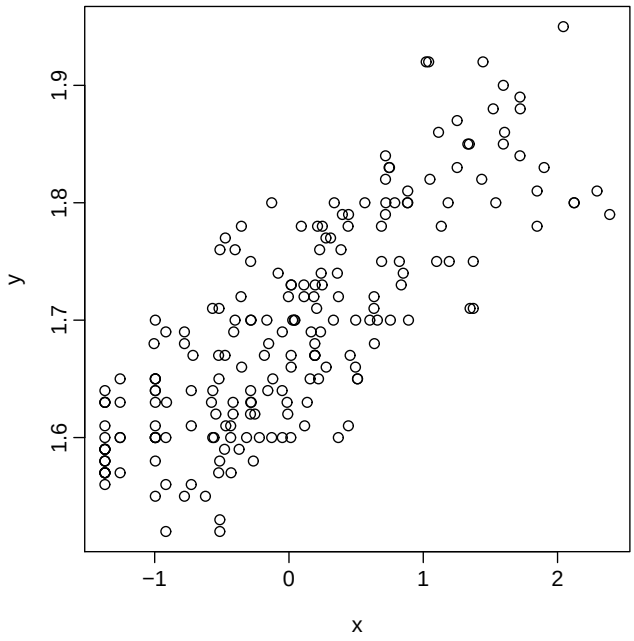

Figura 3.4: Gráfico de dispersão entre o traço latente estimado de cada pessoa (x) e sua altura em metros (y) .

Nesse exemplo, a correlação entre $\hat{\theta}$ (x) e *altura* (y) é 0.8003 (Figura 3.4).

Um modelo de regressão linear simples pode ser utilizado para obter a estimativa da altura em metros. No **R** isso pode ser realizado da seguinte maneira:

```
> altura.fit<-lm(y~x) # ajuste do modelo
> altura.fit

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     1.6919       0.0819
```

Com o modelo ajustado, pode-se então, obter a estimativa da altura para um $\hat{\theta}_j$.

```
> novo<-data.frame(x =theta.resposta[,1])
> novo  # theta de Adilson e Dalton

      x
1 1.212
2 0.221

> altura.pred <- predict(altura.fit, novo, interval="prediction")
> altura.pred[,1]

   1    2
1.79 1.71
```

As alturas verdadeiras são: Adilson<-1,80 m e Dalton<-1,73 m.

3.1.4 Análise pelo pacote ltm

No **R** o ajuste de modelos com 2 parâmetros pode ser realizado com a função `tpm()`. Deve-se redefinir o argumento `constraint` para que o parâmetro c seja zero. Nessa opção, o exemplo `const=cbind(1:14,1,0)` indica que para os 14 itens (1:14), o parâmetro c (1) será definido como 0 (0).

```
> altura.tpm<-tpm(altura.itens,const=cbind(1:14,1,0))
> altura.tpm

Call:
tpm(data = altura.itens, constraint = cbind(1:14, 1, 0))

Coefficients:
     Gussng  Dffclt  Dscrmn
i1        0   2.537    0.30
i2        0   1.221    1.16
i3        0   1.223    1.48
i4        0   1.345    0.86
i5        0   1.076    2.20
i6        0   1.124    1.81
i7        0  -0.114    4.15
i8        0   2.636    1.10
i9        0  -0.813    1.13
i10       0   0.035    3.12
i11       0   0.405    1.17
i12       0   0.234    2.40
i13       0   0.808    1.75
i14       0   0.511    2.83

Log.Lik: -1425
```

3.1.4.1 Gráficos com o pacote ltm

As curvas características dos itens, com as funções do pacote `ltm`, podem ser obtidas da seguinte maneira (Figura 3.5):

Outras curvas também podem ser obtidas. Por exemplo, pode-se ter interesse em avaliar apenas as curvas de alguns itens. Experimente a seguinte sintaxe:

```
> plot(altura.tpm,items=1:5)
```

```
> plot(altura.tpm,item=1:14,sub='Altura',legend=F)
```

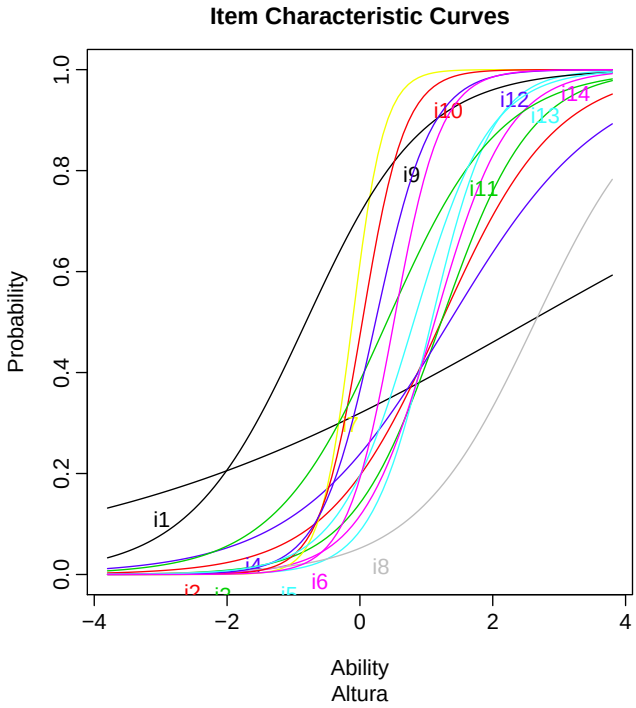

Figura 3.5: Curva característica dos itens para os dados sobre altura.

As curvas de informação dos itens podem ser obtidas da seguinte maneira (Figura 3.6):

A curva de informação do teste pode ser obtida utilizando o argumento `items=0` (Figura 3.7):

```
> plot(altura.tpm,type="IIC",items=1:5)
```

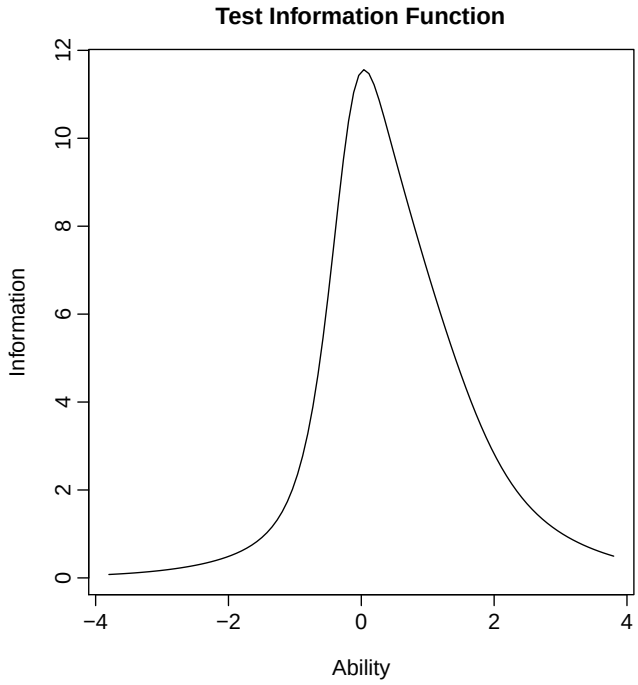

Figura 3.6: Curva de informação dos itens para dados sobre altura.

```
> plot(altura.tpm,type="IIC",items=0)
```

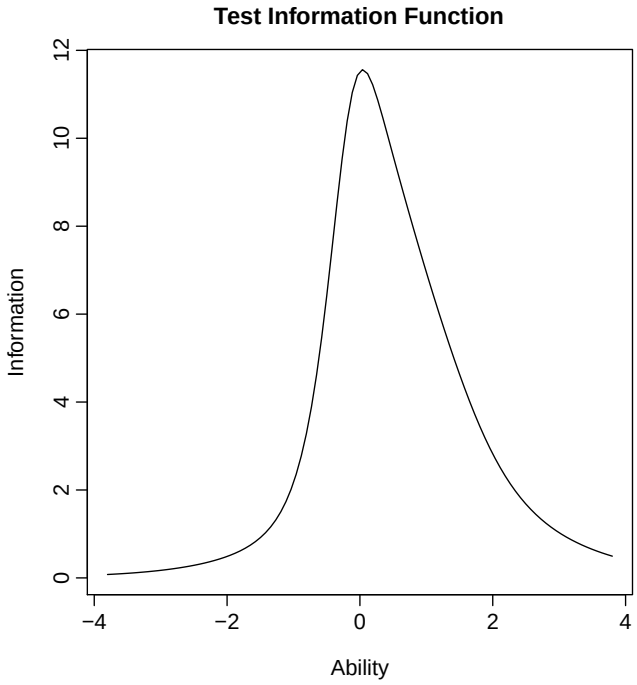

Figura 3.7: Curva de informação do teste para os dados sobre altura.

Estimação de θ

Para obter os valores de $\hat{\theta}$ basta aplicar a função `factor.scores()` sobre o objeto `altura.tpm`:

```
> altura.prof<-factor.scores(altura.tpm)
> head(altura.prof$score)
```

```
  i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14 Obs    Exp
1  0  0  0  0  0  0  0  0  0   0   0   0   0   0  20 18.170
2  0  0  0  0  0  0  0  0  0   0   0   0   1   1   1  0.028
3  0  0  0  0  0  0  0  0  0   0   0   1   0   0   1  0.908
4  0  0  0  0  0  0  0  0  0   0   0   1   1   0   1  0.080
5  0  0  0  0  0  0  0  0  0   0   1   1   0   0   1  0.270
6  0  0  0  0  0  0  0  0  1   0   0   0   0   0  14 11.917
     z1 se.z1
1 -1.19  0.63
2 -0.32  0.32
3 -0.59  0.40
4 -0.36  0.33
5 -0.43  0.35
6 -0.84  0.50
```

Nesse resultado, z_1 indica o valor estimado de θ.

3.2 Exemplo: modelo de 3 parâmetros (SARESP)

O conjunto de dados do SARESP corresponde a uma amostra de 3054 respondentes da prova de Língua Portuguesa aplicada, com 30 itens, no ano de 2007, a alunos do terceiro ano dos turnos matutino, vespertino e noturno de escolas públicas de SP.

Para esse exemplo será utilizado apenas o turno da manhã, que possui 1001 respondentes.

3.2.1 Leitura do arquivo

O arquivo original está no seguinte formato:

```
1                    ABDCAABCDCAADBDDABDDDCDCADADBC
2                    ADCDAADBCCCABABDBCABADCABBBDBC
3                    BDCACABACACABADDABDBADCABDBBBC
1 1 011001138433m07 ADDCAADBDBABDDABABCBCBDCCAADBC
1 1 011002964093m07 DBACAACDABACDBBAABDDBCAACDADBC
1 1 011004154243m07 ABDCAABCDDAADBDDABDADCDDADACBC
1 1 011005367283m07 DCDCACABADCCCCADCBCABCCCCAABCD
1 1 011007519633m07 DBDCABCDBAABDBDCCDBDDBADBBABBA
1 1 011008054863m07 DDBCAACDBAAACBBBCBDDDBADBCACBC
```

As três primeiras linhas são as respostas (gabarito) de cada um dos trinta itens. Cada linha corresponde a um turno. A primeira coluna indica o turno, a segunda coluna o teste, e a terceira indica o identificador de cada respondente. Por último, cada letra representa a resposta de cada item.

O arquivo com os dados pode ser lido no **R** com os seguintes comandos:

```
> saresp<-read.fwf(
+            'http://www.ufpr.br/~aanjos/TRI/sinape/dados/saresp.dat',
+             widths=c(1,-1,1,-1,12,3,-1,rep(1,30)),
+             header=FALSE,skip=3,na.strings=' ')
> colnames(saresp)<-c('grupo','escola','id','turno',
+                     paste('i',1:30,sep=""))
```

Os argumentos nessa função indicam que:

- `widths=c(1,-1,1,-1,12,3,-1,rep(1,30))`: leia a coluna 1 com um dígito, não leia a coluna 2, leia a coluna 3 com um dígito, não leia a coluna 4, leia os próximos 12 algarismos como sendo uma coluna, leia

os próximos 3 algarismos como um coluna, não leia a próxima coluna e, por último, considere cada uma das letras como uma coluna;

- `header=FALSE`: indica que as colunas do arquivo não possuem um cabeçalho (nomes nas colunas);
- `skip=3`: não leia as 3 primeiras linhas;
- `na.strings=' '`: considere o espaço em branco como *missing*.

Para ver os dados desse objeto digite:

```
> head(saresp)
```

O gabarito pode ser obtido de maneira semelhante:

```
> gabarito<-read.fwf('http://www.ufpr.br/~aanjos/TRI/dados/saresp.dat',
+ widths=c(-20,rep(1,30)),header=F,nr=3)
> colnames(gabarito)<-c(paste('i',1:30,sep=""))
> gabarito
```

```
  i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14 i15 i16
1  A  B  D  C  A  A  B  C  D   C   A   A   D   B   D   D
2  A  D  C  D  A  A  D  B  C   C   C   A   B   A   B   D
3  B  D  C  A  C  A  B  A  C   A   C   A   B   A   D   D
  i17 i18 i19 i20 i21 i22 i23 i24 i25 i26 i27 i28 i29 i30
1   A   B   D   D   D   C   D   C   A   D   A   D   B   C
2   B   C   A   B   A   D   C   A   B   B   B   D   B   C
3   A   B   D   B   A   D   C   A   B   D   B   B   B   C
```

Nessa opção, `nr=3` indica que apenas as 3 primeiras linhas devem se lidas.

Observe que as respostas e também o gabarito foram fornecidos em modo literal (letras). No **R** , as principais funções conhecidas para análise de dados da TRI requerem que as respostas estejam em um objeto do tipo `data.frame`

ou `matrix`, com números 0 e 1 (0 indica uma resposta incorreta e 1 uma resposta correta).

No pacote `Deducer`, há uma função chamada `recode.variables()` que pode ser utilizada para converter as respostas de letras para números.

Para instalar o pacote, utilize o comando `install.packages('Deducer')`.

Para obter o pacote, utilize:

```
> library(Deducer)
```

A recodificação pode ser realizada da seguinte maneira:

```
> dados<-recode.variables(saresp,"'A'->1; 'B'->2; 'C'->3; 'D'->4")
```

Observe que, por opção, foi criado um novo objeto chamado `dados`. Utilize a função `head()` para ver uma parte dos dados. Na recodificação não são mantidos os nomes das colunas do arquivo original. Por isso, utiliza-se a função `names()` para recolocar os nomes das colunas.

```
> names(dados)<-names(saresp) # colocar os nomes do arquivo original
```

Como, nesse exemplo, há o interesse apenas no período da manhã, foram extraídos os dados no seguinte objeto:

```
> manha<-subset(dados,grupo==1,select=names(dados))
> head(manha)
```

```
  grupo escola          id turno i1 i2 i3 i4 i5 i6 i7 i8 i9
1     1      1 11001138433   m07  1  4  4  3  1  1  4  2  4
2     1      1 11002964093   m07  4  2  1  3  1  1  3  4  1
3     1      1 11004154243   m07  1  2  4  3  1  1  2  3  4
4     1      1 11005367283   m07  4  3  4  3  1  3  1  2  1
```

```
5     1      1 11007519633   m07  4  2  4  3  1  2  3  4  2
6     1      1 11008054863   m07  4  4  2  3  1  1  3  4  2
  i10 i11 i12 i13 i14 i15 i16 i17 i18 i19 i20 i21 i22 i23
1   2   1   2   4   4   1   2   1   2   3   2   3   2   4
2   2   1   3   4   2   2   1   1   2   4   4   2   3   1
3   4   1   1   4   2   4   4   1   2   4   1   4   3   4
4   4   3   3   3   3   1   4   3   2   3   1   2   3   3
5   1   1   2   4   2   4   3   3   4   2   4   4   2   1
6   1   1   1   3   2   2   2   3   2   4   4   4   2   1
  i24 i25 i26 i27 i28 i29 i30
1   3   3   1   1   4   2   3
2   1   3   4   1   4   2   3
3   4   1   4   1   3   2   3
4   3   3   1   1   2   3   4
5   4   2   2   1   2   2   1
6   4   2   3   1   3   2   3
```

Também é necessário recodificar o gabarito:

```
> gab<-recode.variables(gabarito,"'A'->1; 'B'->2; 'C'->3; 'D'->4")
> colnames(gab)<-c(paste('i',1:30,sep=""))
> gab

  i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14 i15 i16
1  1  2  4  3  1  1  2  3  4   3   1   1   4   2   4   4
2  1  4  3  4  1  1  4  2  3   3   3   1   2   1   2   4
3  2  4  3  1  3  1  2  1  3   1   3   1   2   1   4   4
  i17 i18 i19 i20 i21 i22 i23 i24 i25 i26 i27 i28 i29 i30
1   1   2   4   4   4   3   4   3   1   4   1   4   2   3
2   2   3   1   2   1   4   3   1   2   2   2   4   2   3
3   1   2   4   2   1   4   3   1   2   4   2   2   2   3
```

Observe que, no objeto `dados`, as respostas estão codificadas com os valores 1, 2, 3 e 4. No pacote `ltm` há uma função chamada `multi.choice()` que transforma as respostas em 0 ou 1, de acordo com o gabarito.

O objetos `manha.NA` contém as respostas numéricas no formato 0 e 1.

```
> manha.NA<-mult.choice(manha[,5:34],as.numeric(gab[1,]))
```

```
> head(manha.NA)
```

```
  i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14 i15 i16
1  1  0  1  1  1  1  0  0  1   0   1   0   1   0   0   0
2  0  1  0  1  1  1  0  0  0   0   1   0   1   1   0   0
3  1  1  1  1  1  1  1  1  1   0   1   1   1   1   1   1
4  0  0  1  1  1  0  0  0  0   0   0   0   0   0   0   1
5  0  1  1  1  1  0  0  0  0   0   1   0   1   1   1   0
6  0  0  0  1  1  1  0  0  0   0   1   1   0   1   0   0
  i17 i18 i19 i20 i21 i22 i23 i24 i25 i26 i27 i28 i29 i30
1   1   1   0   0   0   0   1   1   0   0   1   1   1   1
2   1   1   1   1   0   1   0   0   0   1   1   1   1   1
3   1   1   1   0   1   1   1   0   1   1   1   0   1   1
4   0   1   0   0   0   1   0   1   0   0   1   0   0   0
5   0   0   0   1   1   0   0   0   0   0   1   0   1   0
6   0   1   1   1   1   0   0   0   0   0   1   0   1   1
```

3.2.2 Ausência de respostas

Observe que nesse conjunto de respostas existem 'missings' ou 'NA's':

```
> descript(manha.NA)$missin
```

```
     i1  i2  i3 i4 i5 i6  i7  i8  i9 i10 i11 i12 i13 i14
Freq  0 6.0 4.0  0  0  0 1.0 2.0 3.0 1.0   0   0 4.0 1.0
(%)   0 0.6 0.4  0  0  0 0.1 0.2 0.3 0.1   0   0 0.4 0.1
     i15 i16 i17 i18 i19 i20 i21 i22 i23 i24 i25 i26 i27
Freq 1.0 1.0   0   0 4.0 1.0   0 2.0 4.0 4.0 2.0 3.0 1.0
(%)  0.1 0.1   0   0 0.4 0.1   0 0.2 0.4 0.4 0.2 0.3 0.1
```

```
     i28 i29 i30
Freq 3.0 3.0 1.0
(%)  0.3 0.3 0.1
```

Quando um pacote não permite 'NA', as respostas faltantes devem ser codificadas como uma resposta incorreta.

No **R** , pode-se recodificar os 'NA's' da seguinte maneira:

```
> manha.f<-ifelse(is.na(manha.NA)==T,0,manha.NA)
```

Nas análises seguintes será considerado o conjunto de respostas sem 'NA's'.

3.2.3 Análise pelo pacote irtoys

A calibração de modelos com 3 parâmetros da TRI pode ser obtido da seguinte maneira, com a função `est()`:

```
> manha.f.par<-est(manha.f, model = "3PL", engine = "ltm",
+                  nqp = 20, logistic = TRUE)
```

O objeto `manha.f.par` contém os valores dos parâmetros a, b e c do modelo com 3 parâmetros. Para visualizar os primeiros parâmetros utilize:

```
> head(manha.f.par)
```

```
   [,1]  [,2]  [,3]
i1 0.66  1.99 0.253
i2 1.92  0.26 0.194
i3 1.75  1.11 0.159
i4 1.09  0.26 0.144
i5 0.94 -2.72 0.042
i6 0.57 -0.42 0.036
```

Para obtenção da curva característica de informação utiliza-se o objeto que contém o resultado da estimação dos parâmetros (Figura 3.8).

```
> plot(iif(manha.f.par))
```

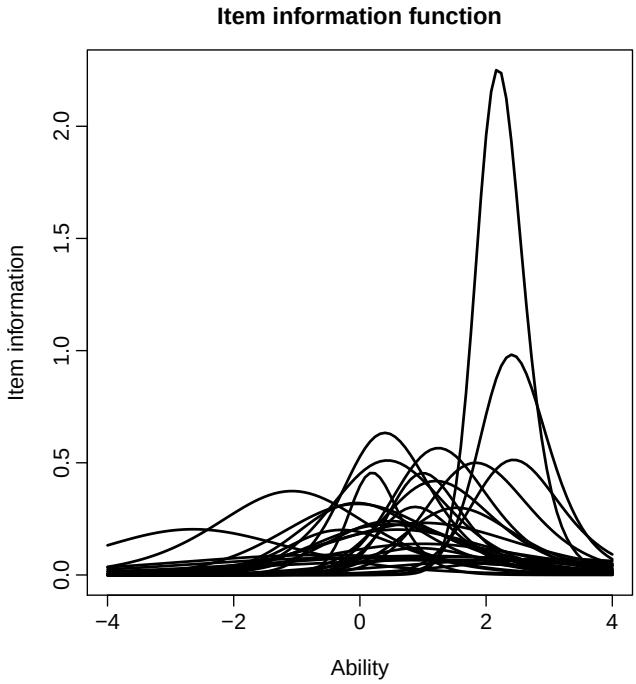

Figura 3.8: Curva característica de informação para os dados do SARESP 2007.

Na Figura 3.9, tem-se as curvas características dos itens para os dados do SARESP 2007.

```
> plot(irf(manha.f.par))
```

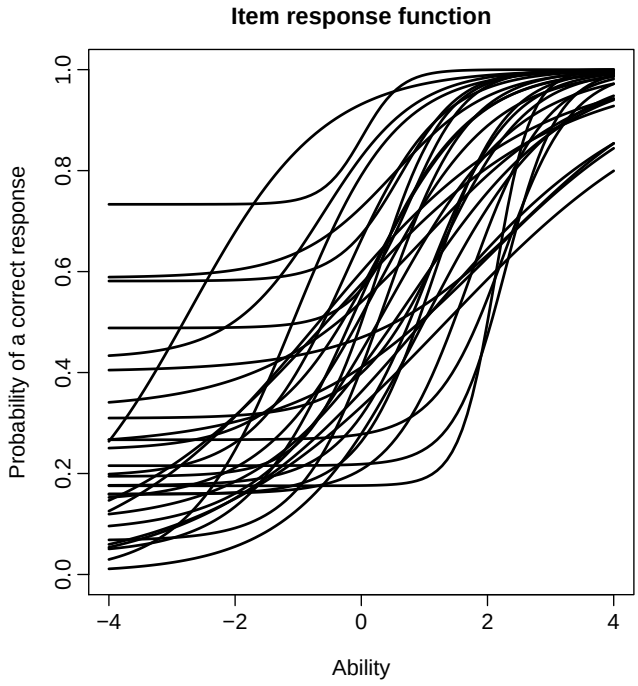

Figura 3.9: Curva característica dos itens do SARESP 2007.

3.2.4 Estimação da habilidade θ

Para obter $\hat{\theta}$ ou a habilidade de cada indivíduo, após a estimação dos parâmetros dos itens, pode-se utilizar a função `eap()`. É necessário fornecer o arquivo de respostas e o objeto com os parâmetros estimados do modelo:

```
> manha.f.sco<-eap(manha.f,manha.f.par,qu=normal.qu())
```

O objeto `manha.f.sco` contém o $\hat{\theta}$ de cada indivíduo.

```
> head(manha.f.sco)
```

```
        est  sem  n
[1,]  0.061 0.48 30
[2,]  0.588 0.41 30
[3,]  1.926 0.47 30
[4,] -1.711 0.60 30
[5,] -0.478 0.45 30
[6,] -0.290 0.45 30
```

E na Figura 3.10 tem-se a curva de informação do teste.

```
> plot(tif(manha.f.par))
```

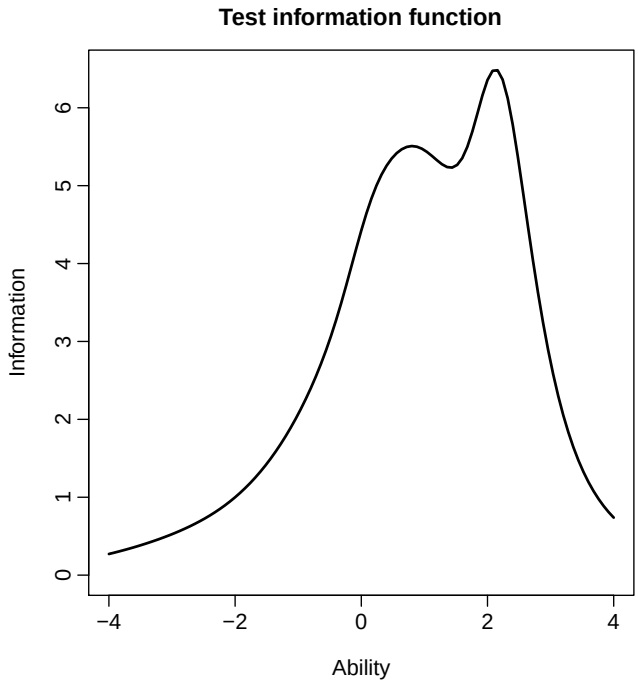

Figura 3.10: Curva de informação do teste do período da manhã para os dados do SARESP 2007.

A função `trf()` pode ser utilizada, também, para visualizar a relação entre a habilidade dos respondentes e o *escore*:

```
> trf(manha.f.par)

$x
  [1] -4.00 -3.92 -3.84 -3.76 -3.68 -3.60 -3.52 -3.44 -3.36
 [10] -3.28 -3.20 -3.12 -3.04 -2.96 -2.88 -2.80 -2.72 -2.64
 [19] -2.56 -2.48 -2.40 -2.32 -2.24 -2.16 -2.08 -2.00 -1.92
 [28] -1.84 -1.76 -1.68 -1.60 -1.52 -1.44 -1.36 -1.28 -1.20
 [37] -1.12 -1.04 -0.96 -0.88 -0.80 -0.72 -0.64 -0.56 -0.48
 [46] -0.40 -0.32 -0.24 -0.16 -0.08  0.00  0.08  0.16  0.24
 [55]  0.32  0.40  0.48  0.56  0.64  0.72  0.80  0.88  0.96
 [64]  1.04  1.12  1.20  1.28  1.36  1.44  1.52  1.60  1.68
 [73]  1.76  1.84  1.92  2.00  2.08  2.16  2.24  2.32  2.40
 [82]  2.48  2.56  2.64  2.72  2.80  2.88  2.96  3.04  3.12
 [91]  3.20  3.28  3.36  3.44  3.52  3.60  3.68  3.76  3.84
[100]  3.92  4.00

$f
  [1]  7.1  7.2  7.2  7.2  7.3  7.3  7.4  7.4  7.5  7.5  7.6
 [12]  7.7  7.7  7.8  7.9  8.0  8.0  8.1  8.2  8.3  8.4  8.5
 [23]  8.6  8.7  8.8  8.9  9.1  9.2  9.4  9.5  9.7  9.8 10.0
 [34] 10.2 10.4 10.6 10.8 11.0 11.2 11.5 11.7 12.0 12.2 12.5
 [45] 12.8 13.1 13.4 13.8 14.1 14.5 14.9 15.2 15.6 16.0 16.4
 [56] 16.8 17.2 17.6 18.0 18.4 18.8 19.2 19.6 20.0 20.4 20.8
 [67] 21.2 21.5 21.9 22.3 22.6 23.0 23.3 23.7 24.1 24.4 24.7
 [78] 25.1 25.4 25.7 26.0 26.3 26.5 26.8 27.0 27.2 27.4 27.6
 [89] 27.7 27.9 28.0 28.1 28.2 28.3 28.4 28.5 28.6 28.7 28.8
[100] 28.8 28.9

$ni
[1] 30

attr(,"class")
```

```
[1] "trf"
```

Aqui, x representa a habilidade estimada e, **f** o correspondente escore esperado.

3.2.5 Posicionamento dos respondentes

Um objeto final, com os escores, posição dos respondentes e número de acertos pode ser criado com os seguintes comandos:

```
> final.rank<-data.frame('escore'=manha.f.sco[,1],
+                        'posição'=rank(manha.f.sco[,1]),
+                        'acertos'=margin2table(manha.f)[-1002,31])
> head(final.rank)
```

```
  escore posição acertos
1  0.061     538      16
2  0.588     748      17
3  1.926     987      26
4 -1.711      15       8
5 -0.478     326      12
6 -0.290     398      13
```

Pode-se visualizar o resultado ordenado pelo número de acertos,

```
> final.acertos<- final.rank[order(final.rank$acertos),]
> head(final.acertos)
```

```
    escore posição acertos
25   -0.95     156       7
88   -1.94       2       7
128  -1.66      18       7
149  -1.60      24       7
```

```
183  -1.84       7       7
192  -1.60      23       7
```

```
> tail(final.acertos)
```

```
    escore posição acertos
422    2.5     999      27
803    2.3     997      27
859    2.2     993      27
229    2.6    1000      28
628    2.4     998      28
558    2.8    1001      29
```

ou pela posição (classificação) dos respondentes:

```
> final.escore<- final.rank[order(final.rank$escore),]
> head(final.escore)
```

```
    escore posição acertos
257   -1.9       1       7
88    -1.9       2       7
971   -1.9       3       7
315   -1.9       4       8
325   -1.9       5      10
395   -1.8       6       7
```

```
> tail(final.escore)
```

```
    escore posição acertos
409    2.3     996      27
803    2.3     997      27
628    2.4     998      28
422    2.5     999      27
229    2.6    1000      28
558    2.8    1001      29
```

3.2.6 Mudança de escala

Se for de interesse, pode-se alterar a escala do escore com a função `score.transform()` do pacote `CTT`. Por exemplo, pode-se mudar a escala da habilidade N(0,1) para uma escala N(500,10):

```
> novo.score<-score.transform(manha.f.sco[,1],
+ mu.new = 500, sd.new = 100, normalize = FALSE)
> round(head(novo.score$new,n=30),2)

 [1] 507 567 719 305 446 467 462 461 419 580 544 492 550 459
[15] 425 403 640 618 701 606 513 551 561 558 392 634 530 641
[29] 379 543
```

Agora, o novo escore é apresentado sem números negativos.

Observe o argumento `normalize`. Quando a opção for `TRUE`, os escores são padronizados de modo a garantir que a distribuição tenha a escala desejada.

3.2.7 Utilizando o pacote ltm

De forma semelhante, pode-se obter as estimativas dos parâmetros e da habilidade com uso do pacote `ltm` para os dados do SARESP 2007.

A estimativa dos parâmetros pode ser obtida com a função `tpm()` da seguinte maneira:

```
> manha.f.tpm<-tpm(manha.f)
> manha.f.tpm

Call:
tpm(data = manha.f)

Coefficients:
```

```
     Gussng  Dffclt  Dscrmn
i1    0.243   1.977    0.63
i2    0.191   0.253    1.90
i3    0.156   1.114    1.72
i4    0.142   0.253    1.08
i5    0.045  -2.729    0.94
i6    0.041  -0.402    0.57
i7    0.097   0.784    0.81
i8    0.002   1.351    0.52
i9    0.175   1.043    1.52
i10   0.309   1.292    1.47
i11   0.011  -0.638    0.60
i12   0.323   0.971    0.77
i13   0.039  -0.106    1.17
i14   0.243   0.233    1.23
i15   0.265   2.296    1.75
i16   0.158   1.708    1.61
i17   0.493   0.797    2.31
i18   0.731   0.001    3.14
i19   0.075   0.433    0.95
i20   0.591   0.533    1.32
i21   0.003   0.984    0.58
i22   0.173   2.196    2.83
i23   0.215   2.351    2.25
i24   0.402   2.587    0.80
i25   0.003   0.999    0.96
i26   0.192  -0.247    1.36
i27   0.578   0.606    1.99
i28   0.066   0.359    1.52
i29   0.003  -1.085    1.22
i30   0.426  -0.620    1.36

Log.Lik: -17785
```

O erro padrão das estimativas dos parâmetros pode ser obtido com a

função `summary()`:

```
> summary(manha.f.tpm)

Call:
tpm(data = manha.f)

Model Summary:
 log.Lik   AIC   BIC
  -17785 35750 36192

Coefficients:
            value std.err z.vals
Gussng.i1   0.243   0.164  1.480
Gussng.i2   0.191   0.072  2.646
Gussng.i3   0.156   0.040  3.940
Gussng.i4   0.142   0.130  1.096
Gussng.i5   0.045   1.094  0.041
Gussng.i6   0.041   0.556  0.073
Gussng.i7   0.097   0.162  0.603
Gussng.i8   0.002   0.026  0.076
Gussng.i9   0.175   0.042  4.190
Gussng.i10  0.309   0.049  6.248
Gussng.i11  0.011   0.246  0.045
Gussng.i12  0.323   0.152  2.129
Gussng.i13  0.039   0.183  0.215
Gussng.i14  0.243   0.120  2.032
Gussng.i15  0.265   0.028  9.436
Gussng.i16  0.158   0.033  4.841
Gussng.i17  0.493   0.041 11.968
Gussng.i18  0.731   0.042 17.547
Gussng.i19  0.075   0.153  0.492
Gussng.i20  0.591   0.076  7.786
Gussng.i21  0.003   0.034  0.077
Gussng.i22  0.173   0.017 10.387
```

```
Gussng.i23  0.215   0.018 12.048
Gussng.i24  0.402   0.070  5.732
Gussng.i25  0.003   0.061  0.053
Gussng.i26  0.192   0.138  1.390
Gussng.i27  0.578   0.051 11.361
Gussng.i28  0.066   0.068  0.979
Gussng.i29  0.003   0.044  0.061
Gussng.i30  0.426   0.184  2.317
Dffclt.i1   1.977   0.564  3.508
Dffclt.i2   0.253   0.163  1.552
Dffclt.i3   1.114   0.113  9.866
Dffclt.i4   0.253   0.357  0.708
Dffclt.i5  -2.729   1.887 -1.446
Dffclt.i6  -0.402   2.158 -0.186
Dffclt.i7   0.784   0.493  1.591
Dffclt.i8   1.351   0.256  5.281
Dffclt.i9   1.043   0.126  8.274
Dffclt.i10  1.292   0.169  7.650
Dffclt.i11 -0.638   0.875 -0.729
Dffclt.i12  0.971   0.634  1.532
Dffclt.i13 -0.106   0.419 -0.253
Dffclt.i14  0.233   0.344  0.678
Dffclt.i15  2.296   0.371  6.181
Dffclt.i16  1.708   0.168 10.183
Dffclt.i17  0.797   0.148  5.379
Dffclt.i18  0.001   0.204  0.007
Dffclt.i19  0.433   0.420  1.032
Dffclt.i20  0.533   0.389  1.370
Dffclt.i21  0.984   0.211  4.661
Dffclt.i22  2.196   0.224  9.821
Dffclt.i23  2.351   0.303  7.755
Dffclt.i24  2.587   0.716  3.615
Dffclt.i25  0.999   0.169  5.927
Dffclt.i26 -0.247   0.336 -0.735
Dffclt.i27  0.606   0.218  2.779
```

```
Dffclt.i28  0.359   0.150  2.389
Dffclt.i29 -1.085   0.131 -8.308
Dffclt.i30 -0.620   0.572 -1.083
Dscrmn.i1   0.632   0.443  1.425
Dscrmn.i2   1.895   0.422  4.486
Dscrmn.i3   1.718   0.421  4.078
Dscrmn.i4   1.082   0.280  3.860
Dscrmn.i5   0.938   0.235  3.989
Dscrmn.i6   0.570   0.304  1.875
Dscrmn.i7   0.811   0.301  2.693
Dscrmn.i8   0.520   0.089  5.868
Dscrmn.i9   1.522   0.332  4.581
Dscrmn.i10  1.468   0.492  2.982
Dscrmn.i11  0.603   0.144  4.175
Dscrmn.i12  0.771   0.373  2.066
Dscrmn.i13  1.168   0.301  3.876
Dscrmn.i14  1.228   0.335  3.666
Dscrmn.i15  1.747   0.982  1.780
Dscrmn.i16  1.607   0.511  3.143
Dscrmn.i17  2.315   0.807  2.867
Dscrmn.i18  3.140   1.152  2.725
Dscrmn.i19  0.955   0.288  3.318
Dscrmn.i20  1.324   0.526  2.515
Dscrmn.i21  0.584   0.093  6.301
Dscrmn.i22  2.833   1.424  1.989
Dscrmn.i23  2.250   0.973  2.311
Dscrmn.i24  0.796   0.557  1.430
Dscrmn.i25  0.964   0.193  4.982
Dscrmn.i26  1.360   0.304  4.478
Dscrmn.i27  1.991   0.721  2.762
Dscrmn.i28  1.515   0.284  5.329
Dscrmn.i29  1.220   0.135  9.052
Dscrmn.i30  1.363   0.387  3.525

Integration:
```

```
method: Gauss-Hermite
quadrature points: 21

Optimization:
Optimizer: optim (BFGS)
Convergence: 0
max(|grad|): 0.022
```

As curvas características dos itens podem ser obtidas da seguinte maneira (3.11):

```
> plot(manha.f.tpm,legend=F)
```

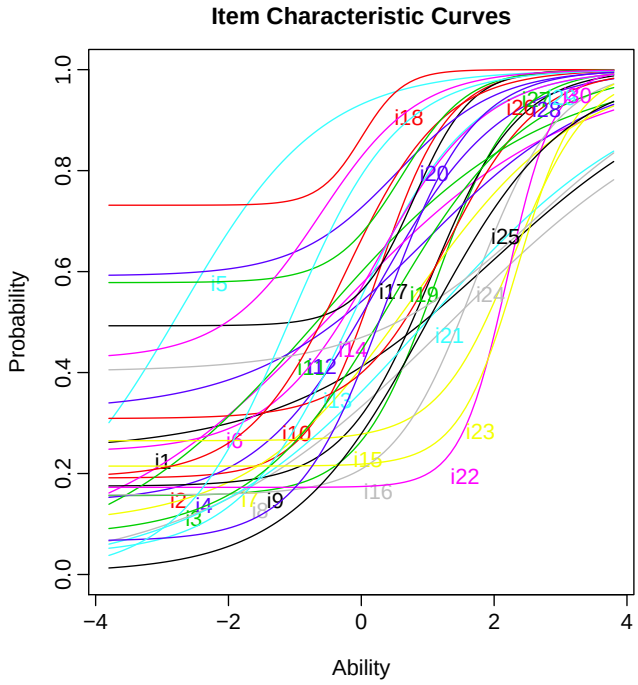

Figura 3.11: Curva característica dos itens para os dados do SARESP 2007.

Outras curvas também podem ser obtidas com os seguintes comandos (Figura 3.12):

```
> par(mfrow=c(2,2))
> plot(manha.f.tpm,items=1:5)
> plot(manha.f.tpm,type="IIC",items=1:5)
> plot(manha.f.tpm,type="IIC",items=0)
> par(mfrow=c(1,1))
```

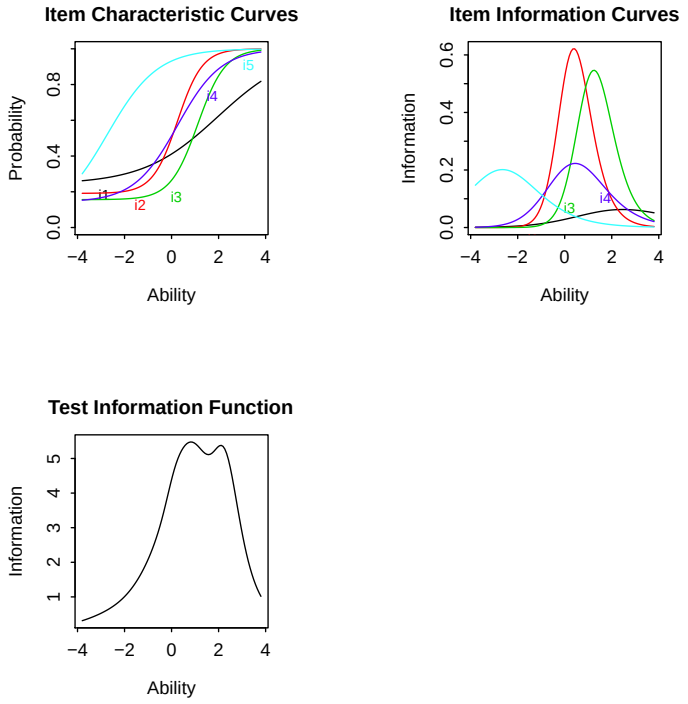

Figura 3.12: Curva característica dos itens sobre SARESP.

As estimativas de θ_j podem ser obtidas com a função `factor.scores()`:

```
> manha.f.prof<-factor.scores(manha.f.tpm)
```

A distribuição das estimativas da habilidade para os dados do SARESP 2007 pode ser obtida com a função `plot` sobre o objeto que contém as habilidades estimadas (Figura 3.13):

```
> plot(manha.f.prof)
```

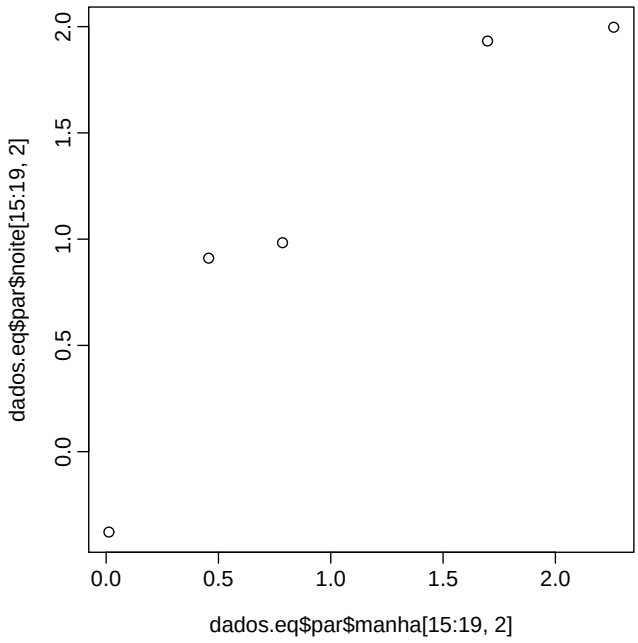

Figura 3.13: Gráfico da proficiência para os dados do SARESP 2007.

E na Figura 3.14 pode-se ver o posicionamento dos itens na escala da prova para o período matutino:

```
> plot(manha.f.prof,include.items=T)
```

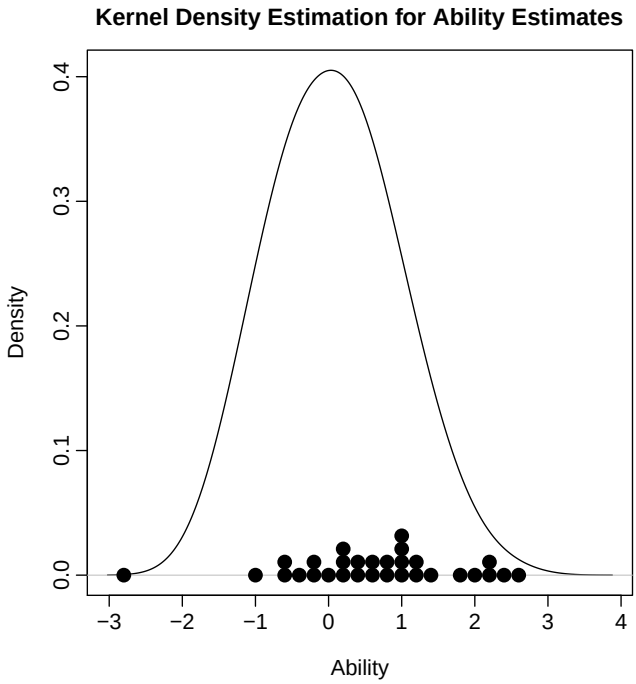

Figura 3.14: Posicionamento dos itens na prova de Língua Portuguesa do SARESP 2007.

Capítulo 4

Equalização

A equalização de testes de diferentes grupos pode ser realizada com a função `plink()` do pacote de mesmo nome.

A equalização envolve uma mudança de escala nas estimativas dos parâmetros dos itens que permite que a comparação entre os diferentes grupos seja realizada (ver Capítulo 1).

4.1 Exemplo: dados do SARESP

Os dados que serão utilizados são provenientes de dois testes aplicados em alunos de turnos diferentes, manhã e noite, do 3^o ano do ensino médio, no ano de 2007, no Estado de São Paulo.

Cada teste possui 30 itens (questões) dos quais 5 são comuns entre os dois testes. Os itens 15 a 19 são exatamente os mesmos aplicados aos dois grupos.

Esses itens em comum permitem que os parâmetros dos itens sejam equalizados em uma mesma escala, de forma que as proficiências entre os dois grupos possam ser comparadas.

4.2 Leitura do arquivo

Utilize as mesmas funções da seção 3.2.1 para ler o arquivo de dados do SARESP.

Como nesse exemplo há o interesse na equalização dos períodos manhã e noite, foram extraídos os dados nos seguintes objetos:

```
> manha<-subset(dados,grupo==1,select=names(dados))
> head(manha)
```

```
  grupo escola          id turno i1 i2 i3 i4 i5 i6 i7 i8 i9
1     1      1 11001138433   m07  1  4  4  3  1  1  4  2  4
2     1      1 11002964093   m07  4  2  1  3  1  1  3  4  1
3     1      1 11004154243   m07  1  2  4  3  1  1  2  3  4
4     1      1 11005367283   m07  4  3  4  3  1  3  1  2  1
5     1      1 11007519633   m07  4  2  4  3  1  2  3  4  2
6     1      1 11008054863   m07  4  4  2  3  1  1  3  4  2
  i10 i11 i12 i13 i14 i15 i16 i17 i18 i19 i20 i21 i22 i23
1   2   1   2   4   4   1   2   1   2   3   2   3   2   4
2   2   1   3   4   2   2   1   1   2   4   4   2   3   1
3   4   1   1   4   2   4   4   1   2   4   1   4   3   4
4   4   3   3   3   3   1   4   3   2   3   1   2   3   3
5   1   1   2   4   2   4   3   3   4   2   4   4   2   1
6   1   1   1   3   2   2   2   3   2   4   4   4   2   1
  i24 i25 i26 i27 i28 i29 i30
1   3   3   1   1   4   2   3
2   1   3   4   1   4   2   3
3   4   1   4   1   3   2   3
4   3   3   1   1   2   3   4
5   4   2   2   1   2   2   1
6   4   2   3   1   3   2   3
```

```
> noite<-subset(dados,grupo==3,select=names(dados))
> head(noite)
```

```
     grupo escola          id turno i1 i2 i3 i4 i5 i6 i7 i8
2054     3      2 11000234293   n07  2  1  3  1  4  1  2  1
2055     3      2 11007076383   n07  3  4  4  3  2  3  3  3
2056     3      2 11008005063   n07  2  4  3  1  3  1  2  1
2057     3      2 11014999823   n07  1  3  3  2  4  1  2  1
2058     3      2 11020696713   n07  2  4  3  2  1  4  1  1
2059     3      2 11020757283   n07  1  4  3  4  4  1  4  1
     i9 i10 i11 i12 i13 i14 i15 i16 i17 i18 i19 i20 i21 i22
2054  3   1   3   2   2   1   2   2   1   2   4   2   2   4
2055  2   1   1   1   2   1   2   3   3   2   4   2   1   1
2056  4   2   3   1   2   1   1   4   1   2   4   2   3   4
2057  4   2   3   1   2   1   1   1   3   2   4   2   1   4
2058  3   4   3   1   1   2   3   2   4   2   3   3   3   2
2059  4   2   3   1   3   1   4   1   3   2   2   4   1   4
     i23 i24 i25 i26 i27 i28 i29 i30
2054   3   3   2   2   2   2   2   3
2055   3   1   2   2   2   3   1   1
2056   3   1   2   4   2   3   4   2
2057   2   2   2   2   2   2   2   3
2058   3   1   4   3   2   1   3   4
2059   4   3   4   4   2   2   3   4
```

A recodificação do gabarito para os dados da manhã e noite podem ser realizadas com a função `mult.choice()` do pacote `ltm`:

```
> manha.NA<-mult.choice(manha[,5:34],as.numeric(gab[1,]))
> noite.NA<-mult.choice(noite[,5:34],as.numeric(gab[3,]))
```

Os objetos `manha.NA` e `noite.NA` contêm as respostas numéricas no formato 0 e 1.

```
> head(manha.NA)
```

```
  i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14 i15 i16 i17 i18
1  1  0  1  1  1  1  0  0  1   0   1   0   1   0   0   0   1   1
```

```
2  0  1  0  1  1  1  0  0  0   0   1   0   1   1   0   0   1   1
3  1  1  1  1  1  1  1  1  1   0   1   1   1   1   1   1   1   1
4  0  0  1  1  1  0  0  0  0   0   0   0   0   0   0   1   0   1
5  0  1  1  1  1  0  0  0  0   0   1   0   1   1   1   0   0   0
6  0  0  0  1  1  1  0  0  0   0   1   1   0   1   0   0   0   1
  i19 i20 i21 i22 i23 i24 i25 i26 i27 i28 i29 i30
1   0   0   0   0   1   1   0   0   1   1   1   1
2   1   1   0   1   0   0   0   1   1   1   1   1
3   1   0   1   1   1   0   1   1   1   0   1   1
4   0   0   0   1   0   1   0   0   1   0   0   0
5   0   1   1   0   0   0   0   0   1   0   1   0
6   1   1   1   0   0   0   0   0   1   0   1   1
```

```
> head(noite.NA)
```

```
     i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14 i15 i16 i17
2054  1  0  1  1  0  1  1  1  1   1   1   0   1   1   0   0   1
2055  0  1  0  0  0  0  0  0  0   1   0   1   1   1   0   0   0
2056  1  1  1  1  1  1  1  1  0   0   1   1   1   1   0   1   1
2057  0  0  1  0  0  1  1  1  0   0   1   1   1   1   0   0   0
2058  1  1  1  0  0  0  0  1  1   0   1   1   0   0   0   0   0
2059  0  1  1  0  0  1  0  1  0   0   1   1   0   1   1   0   0
     i18 i19 i20 i21 i22 i23 i24 i25 i26 i27 i28 i29 i30
2054   1   1   1   0   1   1   0   1   0   1   1   1   1
2055   1   1   1   1   0   1   1   1   0   1   0   0   0
2056   1   1   1   0   1   1   1   1   1   1   0   0   0
2057   1   1   1   1   1   0   0   1   0   1   1   1   1
2058   1   0   0   0   0   1   1   0   0   1   0   0   0
2059   1   0   0   1   1   0   0   0   1   1   1   0   0
```

4.3 Ausência de respostas

Observe que nesse conjunto de respostas existem ‘missings’ ou ‘NA’s’:

```
> descript(manha.NA)$missin
```

```
     i1  i2  i3 i4 i5 i6  i7  i8  i9 i10 i11 i12 i13 i14
Freq  0 6.0 4.0  0  0  0 1.0 2.0 3.0 1.0   0   0 4.0 1.0
(%)   0 0.6 0.4  0  0  0 0.1 0.2 0.3 0.1   0   0 0.4 0.1
     i15 i16 i17 i18 i19 i20 i21 i22 i23 i24 i25 i26 i27
Freq 1.0 1.0   0   0 4.0 1.0   0 2.0 4.0 4.0 2.0 3.0 1.0
(%)  0.1 0.1   0   0 0.4 0.1   0 0.2 0.4 0.4 0.2 0.3 0.1
     i28 i29 i30
Freq 3.0 3.0 1.0
(%)  0.3 0.3 0.1
```

```
> descript(noite.NA)$missin
```

```
      i1  i2  i3  i4  i5  i6  i7  i8  i9 i10 i11 i12 i13
Freq 1.0 4.0 2.0 3.0 2.0 1.0 5.0 3.0 3.0 2.0   0 1.0 3.0
(%)  0.1 0.4 0.2 0.3 0.2 0.1 0.5 0.3 0.3 0.2   0 0.1 0.3
     i14 i15 i16 i17 i18 i19 i20 i21 i22 i23 i24 i25 i26
Freq 6.0 3.0 1.0 1.0 1.0  10 7.0 3.0 7.0 5.0 6.0 3.0 3.0
(%)  0.6 0.3 0.1 0.1 0.1   1 0.7 0.3 0.7 0.5 0.6 0.3 0.3
     i27 i28 i29 i30
Freq 1.0 1.0 1.0   0
(%)  0.1 0.1 0.1   0
```

No **R** pode-se recodificar os 'NA's' da seguinte maneira:

```
> manha.f<-ifelse(is.na(manha.NA)==T,0,manha.NA)
> noite.f<-ifelse(is.na(noite.NA)==T,0,noite.NA)
```

Nas análises seguintes será considerado o conjunto de respostas sem 'NA's'.

4.4 Equalização com o pacote plink

No **R** a equalização (a *posteriori*) entre testes pode ser feita com o uso das funções do pacote `plink` (WEEKS, 2010). Veja também a função `sca()` do pacote `irtoys` (PARTCHEV, 2010).

```
> library(plink)
```

Nesse pacote podem ser utilizados os métodos apresentados na seção 1.2.5.

A equalização de testes, no pacote `plink`, envolve várias etapas. Deve-se fornecer as informações necessárias para que a equalização seja realizada de acordo com as características de cada teste:

1. Um objeto contendo os parâmetros dos itens de cada teste;
2. Um objeto especificando o número de categorias de resposta em cada teste;
3. Um objeto identificando o modelo da TRI associado com cada item;
4. Um objeto identificando os itens comuns entre testes ou grupos.

Estimação dos parâmetros dos itens

Utilizando a função `est()` do pacote `irtoys`, pode-se obter as estimativas dos parâmetros dos itens, considerando um modelo de 3 parâmetros:

```
> library(irtoys)
> manha.par<-est(manha.f, model="3PL", engine="ltm")
> head(manha.par)
```

```
   [,1]  [,2]  [,3]
i1 0.66  1.99 0.253
i2 1.92  0.26 0.194
```

```
i3 1.75  1.11 0.159
i4 1.09  0.26 0.144
i5 0.94 -2.72 0.042
i6 0.57 -0.42 0.036
```

```
> noite.par<-est(noite.f, model="3PL", engine="ltm")
> head(noite.par)
```

```
   [,1] [,2] [,3]
i1 0.57 2.13 0.06
i2 0.74 2.05 0.17
i3 1.82 0.86 0.52
i4 0.89 2.40 0.20
i5 1.35 2.48 0.17
i6 1.51 0.45 0.27
```

Para a equalização, é preciso criar uma lista com os parâmetros estimados de cada grupo e com os itens em comum:

```
> dados.eq<-list()
> dados.eq$par<-list()
> dados.eq$par$manha<-manha.par
> dados.eq$par$noite<-noite.par
```

Em seguida, informa-se quais são os itens em comum entre os dois testes:

```
> dados.eq$comum<-cbind(15:19,15:19)
> dados.eq
```

```
$par
$par$manha
    [,1]   [,2]   [,3]
i1  0.66  1.994 0.2535
```

```
i2  1.92  0.260 0.1943
i3  1.75  1.115 0.1585
i4  1.09  0.258 0.1445
i5  0.94 -2.723 0.0425
i6  0.57 -0.421 0.0356
i7  0.82  0.797 0.1025
i8  0.52  1.348 0.0016
i9  1.53  1.045 0.1762
i10 1.48  1.292 0.3096
i11 0.60 -0.641 0.0100
i12 0.78  0.987 0.3278
i13 1.17 -0.102 0.0408
i14 1.24  0.243 0.2465
i15 1.85  2.260 0.2669
i16 1.65  1.698 0.1596
i17 2.21  0.785 0.4883
i18 3.24  0.012 0.7332
i19 0.98  0.456 0.0843
i20 1.29  0.516 0.5879
i21 0.58  0.982 0.0022
i22 3.56  2.122 0.1757
i23 2.44  2.292 0.2155
i24 0.80  2.581 0.4017
i25 0.97  0.997 0.0033
i26 1.36 -0.242 0.1941
i27 2.04  0.618 0.5813
i28 1.53  0.361 0.0673
i29 1.23 -1.080 0.0027
i30 1.37 -0.612 0.4280

$par$noite
    [,1]   [,2]    [,3]
i1  0.57  2.134 0.06020
i2  0.74  2.048 0.17012
i3  1.82  0.855 0.52304
```

```
i4  0.89  2.398 0.19548
i5  1.35  2.484 0.16578
i6  1.51  0.454 0.26831
i7  1.12  0.804 0.35162
i8  1.15 -0.993 0.17308
i9  3.46  1.698 0.34631
i10 1.98  1.795 0.31069
i11 1.52  0.172 0.28763
i12 0.26  0.052 0.02737
i13 2.60  0.385 0.37827
i14 2.57  0.706 0.34819
i15 1.60  1.997 0.23580
i16 2.39  1.932 0.19656
i17 1.83  0.983 0.43326
i18 1.32 -0.378 0.51865
i19 1.30  0.910 0.23693
i20 1.24  0.062 0.16381
i21 0.78 -0.407 0.00055
i22 1.01  0.739 0.27781
i23 1.81  1.327 0.27460
i24 0.13  9.434 0.06572
i25 0.55 -0.203 0.00290
i26 1.02  1.578 0.25136
i27 1.19  1.010 0.12572
i28 0.65 -1.781 0.00436
i29 0.67  0.954 0.10916
i30 1.25  1.178 0.25225


$comum
     [,1] [,2]
[1,]   15   15
[2,]   16   16
[3,]   17   17
[4,]   18   18
```

```
[5,]   19   19
```

Especifica-se, também, o número de itens de cada teste e o tipo de teste. Como os testes são iguais, basta criar apenas um grupo de categorias, especificando o número de itens e o tipo de modelo que deve ser considerado. Neste exemplo, todos os itens são dicotômicos, por isso a opção `model='drm'`:

```
> pm<-as.poly.mod(30,model='drm')
> pm

An object of class "poly.mod"
Slot "model":
[1] "drm"

Slot "items":
$drm
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20 21 22 23 24 25 26 27 28 29 30
```

Por fim, cria-se um objeto com as informações necessárias para realizar a equalização com a função `as.irt.pars()`:

```
> testes<-as.irt.pars(dados.eq$par, dados.eq$comum,
+                  cat=list(rep(2,30),rep(2,30)),
+                  poly.mod=list(pm,pm),
+                  grp.names=c('manha','noite'))
> testes

An object of class "irt.pars"
Slot "pars":
$manha
      [,1]   [,2]   [,3]
 [1,] 0.66  1.994 0.2535
```

```
 [2,] 1.92  0.260 0.1943
 [3,] 1.75  1.115 0.1585
 [4,] 1.09  0.258 0.1445
 [5,] 0.94 -2.723 0.0425
 [6,] 0.57 -0.421 0.0356
 [7,] 0.82  0.797 0.1025
 [8,] 0.52  1.348 0.0016
 [9,] 1.53  1.045 0.1762
[10,] 1.48  1.292 0.3096
[11,] 0.60 -0.641 0.0100
[12,] 0.78  0.987 0.3278
[13,] 1.17 -0.102 0.0408
[14,] 1.24  0.243 0.2465
[15,] 1.85  2.260 0.2669
[16,] 1.65  1.698 0.1596
[17,] 2.21  0.785 0.4883
[18,] 3.24  0.012 0.7332
[19,] 0.98  0.456 0.0843
[20,] 1.29  0.516 0.5879
[21,] 0.58  0.982 0.0022
[22,] 3.56  2.122 0.1757
[23,] 2.44  2.292 0.2155
[24,] 0.80  2.581 0.4017
[25,] 0.97  0.997 0.0033
[26,] 1.36 -0.242 0.1941
[27,] 2.04  0.618 0.5813
[28,] 1.53  0.361 0.0673
[29,] 1.23 -1.080 0.0027
[30,] 1.37 -0.612 0.4280

$noite
      [,1]   [,2]    [,3]
 [1,] 0.57  2.134 0.06020
 [2,] 0.74  2.048 0.17012
 [3,] 1.82  0.855 0.52304
```

```
 [4,] 0.89  2.398 0.19548
 [5,] 1.35  2.484 0.16578
 [6,] 1.51  0.454 0.26831
 [7,] 1.12  0.804 0.35162
 [8,] 1.15 -0.993 0.17308
 [9,] 3.46  1.698 0.34631
[10,] 1.98  1.795 0.31069
[11,] 1.52  0.172 0.28763
[12,] 0.26  0.052 0.02737
[13,] 2.60  0.385 0.37827
[14,] 2.57  0.706 0.34819
[15,] 1.60  1.997 0.23580
[16,] 2.39  1.932 0.19656
[17,] 1.83  0.983 0.43326
[18,] 1.32 -0.378 0.51865
[19,] 1.30  0.910 0.23693
[20,] 1.24  0.062 0.16381
[21,] 0.78 -0.407 0.00055
[22,] 1.01  0.739 0.27781
[23,] 1.81  1.327 0.27460
[24,] 0.13  9.434 0.06572
[25,] 0.55 -0.203 0.00290
[26,] 1.02  1.578 0.25136
[27,] 1.19  1.010 0.12572
[28,] 0.65 -1.781 0.00436
[29,] 0.67  0.954 0.10916
[30,] 1.25  1.178 0.25225


Slot "cat":
$manha
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[29] 2 2

$noite
```

```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[29] 2 2


Slot "poly.mod":
$manha
An object of class "poly.mod"
Slot "model":
[1] "drm"

Slot "items":
$drm
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20 21 22 23 24 25 26 27 28 29 30



$noite
An object of class "poly.mod"
Slot "model":
[1] "drm"

Slot "items":
$drm
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20 21 22 23 24 25 26 27 28 29 30



Slot "common":
     manha noite
[1,]    15    15
[2,]    16    16
[3,]    17    17
```

```
[4,]    18    18
[5,]    19    19

Slot "location":
[1] FALSE FALSE

Slot "groups":
[1] 2

Slot "dimensions":
[1] 1 1
```

Um `summary()` do objeto `testes` fornece mais informações:

```
> summary(testes,TRUE)

-------- manha --------
Total Number of Items: 30

Number of Dichotomous Items: 30
Dichotomous Model: 3PL

Number of Polytomous Items: 0

-------- noite --------
Total Number of Items: 30

Number of Dichotomous Items: 30
Dichotomous Model: 3PL

Number of Polytomous Items: 0
```

A relação entre os parâmetros dos testes pode ser obtida com um dos quatro métodos implementados na função `plink`. Por exemplo, pode-se utilizar

o método MS: média/desvio padrão. Aqui, definiu-se o grupo de referência como o grupo 2 (noite) e a escala logística (D=1).

```
> testes.l<-plink(testes, rescale="MS", base.grp=2,D=1)
> summary(testes.l,descript=T)
```

```
-------  manha/noite*  -------
Linking Constants

                     A         B
Mean/Mean     1.176208 -0.137051
Mean/Sigma    1.050516 -0.006035
Haebara       0.961429  0.120083
Stocking-Lord 0.858941  0.315060
```

Compare os resultados com o resultado de um modelo de regressão linear simples entre os $\hat{b}_i$ comuns entre `manhã` e `noite`. Observe o gráfico de dispersão (Figura 4.1) e os valores estimados pela regressão.

```
> coef(lm(dados.eq$par$noite[15:19,2]~dados.eq$par$manha[15:19,2]))
```

```
                (Intercept) dados.eq$par$manha[15:19, 2]
                      0.071                        0.976
```

```
> plot(dados.eq$par$manha[15:19,2], dados.eq$par$noite[15:19,2])
```

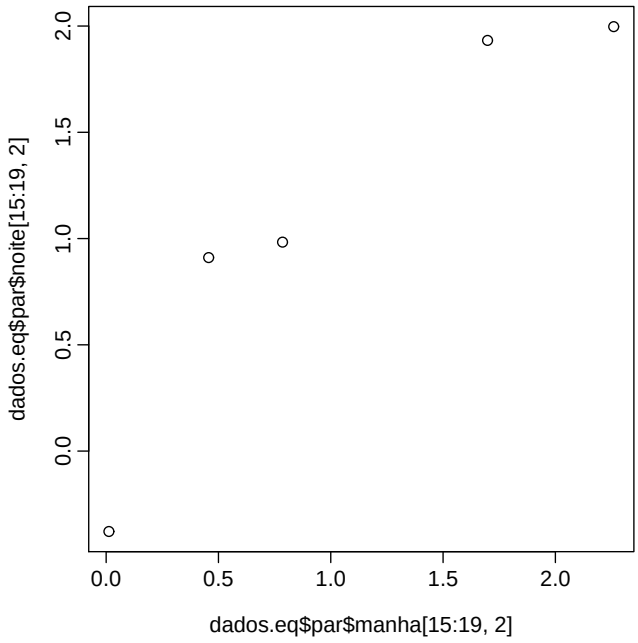

Figura 4.1: Gráfico de dispersão entre os parâmetros b_i dos itens em comum entre os períodos da manhã e noite, da prova de Língua Portuguesa do SARESP 2007.

Para finalizar a equalização, utiliza-se a função `link.pars()`:

```
> testes.l.pars<-link.pars(testes.l)
> testes.l.pars
```

```
$manha
      [,1]    [,2]   [,3]
 [1,] 0.63  2.0886 0.2535
 [2,] 1.83  0.2671 0.1943
 [3,] 1.67  1.1651 0.1585
 [4,] 1.04  0.2654 0.1445
 [5,] 0.90 -2.8664 0.0425
 [6,] 0.54 -0.4479 0.0356
 [7,] 0.78  0.8309 0.1025
 [8,] 0.50  1.4096 0.0016
 [9,] 1.46  1.0915 0.1762
[10,] 1.41  1.3511 0.3096
[11,] 0.57 -0.6790 0.0100
[12,] 0.75  1.0313 0.3278
[13,] 1.12 -0.1132 0.0408
[14,] 1.18  0.2493 0.2465
[15,] 1.76  2.3677 0.2669
[16,] 1.57  1.7778 0.1596
[17,] 2.11  0.8190 0.4883
[18,] 3.08  0.0069 0.7332
[19,] 0.93  0.4735 0.0843
[20,] 1.23  0.5364 0.5879
[21,] 0.56  1.0254 0.0022
[22,] 3.39  2.2228 0.1757
[23,] 2.32  2.4017 0.2155
[24,] 0.76  2.7050 0.4017
[25,] 0.92  1.0408 0.0033
[26,] 1.30 -0.2598 0.1941
[27,] 1.94  0.6431 0.5813
[28,] 1.45  0.3734 0.0673
[29,] 1.17 -1.1409 0.0027
[30,] 1.31 -0.6484 0.4280

$noite
      [,1]   [,2]    [,3]
```

```
 [1,] 0.57  2.134 0.06020
 [2,] 0.74  2.048 0.17012
 [3,] 1.82  0.855 0.52304
 [4,] 0.89  2.398 0.19548
 [5,] 1.35  2.484 0.16578
 [6,] 1.51  0.454 0.26831
 [7,] 1.12  0.804 0.35162
 [8,] 1.15 -0.993 0.17308
 [9,] 3.46  1.698 0.34631
[10,] 1.98  1.795 0.31069
[11,] 1.52  0.172 0.28763
[12,] 0.26  0.052 0.02737
[13,] 2.60  0.385 0.37827
[14,] 2.57  0.706 0.34819
[15,] 1.60  1.997 0.23580
[16,] 2.39  1.932 0.19656
[17,] 1.83  0.983 0.43326
[18,] 1.32 -0.378 0.51865
[19,] 1.30  0.910 0.23693
[20,] 1.24  0.062 0.16381
[21,] 0.78 -0.407 0.00055
[22,] 1.01  0.739 0.27781
[23,] 1.81  1.327 0.27460
[24,] 0.13  9.434 0.06572
[25,] 0.55 -0.203 0.00290
[26,] 1.02  1.578 0.25136
[27,] 1.19  1.010 0.12572
[28,] 0.65 -1.781 0.00436
[29,] 0.67  0.954 0.10916
[30,] 1.25  1.178 0.25225
```

Agora, os parâmetros dos dois testes estão equalizados, ou seja, podem ser utilizados para estimar a proficiência dos dois grupos, de modo que eles sejam comparáveis em uma mesma escala.

Capítulo 5

Simulação de respostas dicotômicas no R

Estudos de simulação podem ser úteis em vários aspectos, como avaliar métodos de estimação por exemplo. No **R** , existem várias funções que podem ser utilizadas para simulação de respostas segundo alguns modelos da TRI.

O objetivo deste capítulo é apresentar algumas dessas funções para geração de padrões de respostas segundo algum modelo. Serão utilizadas funções dos pacotes `irtoys` e `ltm`. O pacote `plink` também possui uma função para simulação de respostas mas não será apresentada nesse texto.

5.1 Simulação de respostas utilizando o pacote irtoys

Primeiro, carregue o pacote `irtoys` no **R** :

```
> library(irtoys)
```

Considere um modelo de 3 parâmetros. Inicialmente deve-se definir os valores dos parâmetros e o número de itens. Considere, por exemplo, 45 itens para uma prova.

O parâmetro de discriminação a será simulado, a partir de uma Distribuição Uniforme variando de 0,2 até 3 com a utilização da função `runif()`. Para cada simulação, pode-se definir uma *semente* para que o resultado da simulação possa ser repetido.

```
> set.seed(2345) # semente
> a<-runif(45,.2,3)
```

O parâmetro de dificuldade b será simulado considerando uma sequência de 45 números equidistantes entre os valores -2 e 2, utilizando-se a função `seq()`.

```
> b<-seq(-2,2,length=45)
```

O parâmetro c pode, também, ser baseado em uma distribuição uniforme ou em valores conhecidos a priori. Por exemplo, em um teste com itens com 4 alternativas, há uma chance de 25% de um aluno responder corretamente o item, ao acaso.

```
> set.seed(321)
> #c<-runif(45,.10,.25) ou
> c<-rep(.25,45)
```

Os valores dos parâmetros simulados podem ser agrupados em um `data.frame()` da seguinte maneira:

```
> pa<-cbind(a,b,c);pa
```

```
         a      b    c
 [1,] 0.53 -2.000 0.25
```

```
 [2,] 0.75 -1.909 0.25
 [3,] 2.18 -1.818 0.25
 [4,] 0.30 -1.727 0.25
 [5,] 1.53 -1.636 0.25
 [6,] 1.02 -1.545 0.25
 [7,] 1.89 -1.455 0.25
 [8,] 2.41 -1.364 0.25
 [9,] 1.34 -1.273 0.25
[10,] 2.19 -1.182 0.25
[11,] 0.65 -1.091 0.25
[12,] 1.16 -1.000 0.25
[13,] 0.43 -0.909 0.25
[14,] 0.62 -0.818 0.25
[15,] 1.35 -0.727 0.25
[16,] 1.13 -0.636 0.25
[17,] 1.92 -0.545 0.25
[18,] 1.42 -0.455 0.25
[19,] 2.09 -0.364 0.25
[20,] 0.73 -0.273 0.25
[21,] 2.94 -0.182 0.25
[22,] 1.59 -0.091 0.25
[23,] 2.49  0.000 0.25
[24,] 1.61  0.091 0.25
[25,] 1.18  0.182 0.25
[26,] 0.59  0.273 0.25
[27,] 2.70  0.364 0.25
[28,] 1.91  0.455 0.25
[29,] 1.82  0.545 0.25
[30,] 2.61  0.636 0.25
[31,] 0.67  0.727 0.25
[32,] 1.64  0.818 0.25
[33,] 2.90  0.909 0.25
[34,] 1.86  1.000 0.25
[35,] 0.38  1.091 0.25
[36,] 2.54  1.182 0.25
```

```
[37,] 1.79  1.273 0.25
[38,] 2.21  1.364 0.25
[39,] 0.46  1.455 0.25
[40,] 1.80  1.545 0.25
[41,] 2.36  1.636 0.25
[42,] 1.70  1.727 0.25
[43,] 1.66  1.818 0.25
[44,] 2.43  1.909 0.25
[45,] 1.06  2.000 0.25
```

É necessário, também, simular a proficiência dos indivíduos que respondem o teste. Por exemplo, pode-se simular a proficiência de 1000 pessoas, considerando que essa proficiência tenha uma distribuição simétrica. Nesse caso, pode-se utilizar a função `rnorm()` para gerar os valores da distribuição Normal com média 0 e desvio padrão 1.

```
> set.seed(1236)
> pf<-rnorm(1000)
```

No `irtoys` o padrão de respostas, em função dos parâmetros e da habilidade, pode ser obtido com a função `sim()`. É necessário fornecer os valores dos parâmetros dos itens e da proficiência dos indivíduos:

```
> dados.sim<-sim(ip=pa,x=pf)
```

Agora, o objeto `dados.sim` contém as respostas dos indivíduos para cada um dos 45 itens:

```
> head(dados.sim)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    0    1    1    1    1    1    1     1
[2,]    1    1    1    1    1    1    0    1    1     1
```

```
[3,]    1    1    1    1    1    0    1    1    0     1
[4,]    1    1    1    0    1    1    1    1    1     1
[5,]    1    1    1    1    1    0    1    0    1     0
[6,]    1    1    1    1    1    1    0    1    1     1
     [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19]
[1,]     0     1     0     0     1     0     0     1     0
[2,]     1     1     0     1     1     1     1     0     1
[3,]     1     1     1     1     1     1     1     0     0
[4,]     1     1     1     1     1     1     1     0     1
[5,]     0     1     1     0     0     0     1     0     0
[6,]     1     1     1     1     0     1     0     1     1
     [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28]
[1,]     1     0     0     1     1     0     1     0     1
[2,]     1     1     1     0     0     1     0     0     0
[3,]     1     0     0     0     1     1     1     0     1
[4,]     1     1     1     1     1     1     1     0     1
[5,]     0     0     1     0     0     1     1     1     0
[6,]     1     1     1     1     1     1     1     1     0
     [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
[1,]     1     0     0     1     0     0     1     0     0
[2,]     0     1     1     0     0     0     1     0     1
[3,]     1     1     0     0     0     1     1     0     0
[4,]     1     1     1     0     0     0     1     0     1
[5,]     0     0     1     0     0     0     1     0     0
[6,]     0     0     1     1     0     0     1     0     0
     [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45]
[1,]     1     0     0     0     1     0     1     0
[2,]     1     1     1     0     0     0     0     0
[3,]     0     0     0     0     1     0     1     0
[4,]     0     1     0     0     1     0     0     0
[5,]     0     0     0     0     0     0     0     1
[6,]     1     0     0     0     0     0     1     1
```

Para o padrão de respostas simulado, pode-se obter as estimativas dos parâmetros de um modelo com 3 parâmetros. Inicialmente, considere que o

valor do parâmetro c é conhecido e não será estimado:

```
> dados.tpm<-tpm(dados.sim,constraint = cbind(1:45, 1, 0.25))
> coef(dados.tpm)
```

```
        Gussng Dffclt Dscrmn
Item 1    0.25 -2.154   0.53
Item 2    0.25 -1.954   0.75
Item 3    0.25 -1.749   2.18
Item 4    0.25 -2.167   0.19
Item 5    0.25 -1.694   1.36
Item 6    0.25 -1.723   0.92
Item 7    0.25 -1.288   1.95
Item 8    0.25 -1.264   3.06
Item 9    0.25 -1.168   1.34
Item 10   0.25 -1.302   2.22
Item 11   0.25 -1.490   0.57
Item 12   0.25 -0.853   1.24
Item 13   0.25 -0.552   0.65
Item 14   0.25 -0.776   0.63
Item 15   0.25 -0.755   1.17
Item 16   0.25 -0.655   1.13
Item 17   0.25 -0.485   2.04
Item 18   0.25 -0.429   1.43
Item 19   0.25 -0.275   2.21
Item 20   0.25 -0.310   0.84
Item 21   0.25 -0.189   3.07
Item 22   0.25 -0.045   1.55
Item 23   0.25 -0.021   2.35
Item 24   0.25  0.037   1.40
Item 25   0.25  0.198   0.95
Item 26   0.25  0.277   0.60
Item 27   0.25  0.344   2.22
Item 28   0.25  0.608   2.41
Item 29   0.25  0.506   1.69
```

```
Item 30   0.25  0.650   2.26
Item 31   0.25  0.813   0.68
Item 32   0.25  0.719   1.59
Item 33   0.25  0.910   4.00
Item 34   0.25  1.006   1.96
Item 35   0.25  1.475   0.30
Item 36   0.25  1.167   1.90
Item 37   0.25  1.401   1.79
Item 38   0.25  1.412   2.24
Item 39   0.25  1.526   0.42
Item 40   0.25  1.726   1.83
Item 41   0.25  1.500   2.55
Item 42   0.25  1.664   2.36
Item 43   0.25  1.937   1.83
Item 44   0.25  1.991   1.51
Item 45   0.25  3.331   0.63
```

Agora, pode-se sugerir que a função também estime os valores de c, sem especificar `constraint`.

```
> dados.tpm<-tpm(dados.sim)
> coef(dados.tpm)

        Gussng Dffclt Dscrmn
Item 1  0.6701  0.360   1.07
Item 2  0.4981 -0.994   0.91
Item 3  0.4998 -1.355   2.61
Item 4  0.0564 -4.526   0.17
Item 5  0.2324 -1.736   1.33
Item 6  0.5425 -0.722   1.21
Item 7  0.3445 -1.124   2.09
Item 8  0.2489 -1.260   3.03
Item 9  0.4119 -0.772   1.58
Item 10 0.0954 -1.514   1.99
Item 11 0.0036 -2.406   0.52
```

```
Item 12 0.0012 -1.365   1.07
Item 13 0.0049 -1.491   0.55
Item 14 0.2796 -0.635   0.66
Item 15 0.4382 -0.209   1.55
Item 16 0.0012 -1.219   0.97
Item 17 0.2237 -0.522   1.97
Item 18 0.2585 -0.399   1.45
Item 19 0.2108 -0.333   2.09
Item 20 0.5035  0.620   1.67
Item 21 0.2428 -0.189   3.04
Item 22 0.3299  0.155   1.87
Item 23 0.2227 -0.058   2.24
Item 24 0.0989 -0.317   1.14
Item 25 0.3676  0.590   1.26
Item 26 0.4384  1.080   1.06
Item 27 0.3182  0.476   2.75
Item 28 0.2279  0.575   2.28
Item 29 0.1914  0.379   1.45
Item 30 0.2392  0.636   2.20
Item 31 0.0050 -0.250   0.49
Item 32 0.1830  0.570   1.39
Item 33 0.2368  0.893   3.85
Item 34 0.2474  1.002   1.96
Item 35 0.0306 -0.574   0.22
Item 36 0.3010  1.245   2.74
Item 37 0.2118  1.330   1.55
Item 38 0.2431  1.402   2.17
Item 39 0.3967  2.061   0.77
Item 40 0.2335  1.706   1.66
Item 41 0.2685  1.515   3.02
Item 42 0.2480  1.632   2.52
Item 43 0.2418  1.951   1.68
Item 44 0.2851  1.921   2.43
Item 45 0.2889  2.972   0.96
```

5.2 Simulação de respostas utilizando o pacote ltm

No pacote `ltm` existe a função `rmvlogis()` para simulação de padrões de resposta dicotômicos para modelos da TRI.

Neste texto, veremos como simular dados, considerando respostas dicotômicas para os modelos de 3 e 2 parâmetros e para o modelo Rasch (1 parâmetro).

A função `rmvlogis()` pode ser utilizada para simular qualquer um dos modelos de 1, 2 e 3 parâmetros. O número de parâmetros depende de como é especificado o argumento `theta` da função `rmvlogis()`.

```
rmvlogis(n, thetas, IRT = TRUE, link = c("logit", "probit"),
       distr = c("normal", "logistic", "log-normal", "uniform"),
       z.vals = NULL)
```

O argumento `theta` deve ser uma matriz, onde o número de linhas corresponde ao número de itens, e o número de colunas corresponde ao número de parâmetros. Esse argumento `theta` pode ser definido de várias maneiras, por exemplo:

1. Para um modelo de 3 parâmetros:

```
> set.seed(3)
> theta3<-cbind(.25,seq(-2,2,1),runif(5))
> theta3

     [,1] [,2] [,3]
[1,] 0.25   -2 0.17
[2,] 0.25   -1 0.81
[3,] 0.25    0 0.38
[4,] 0.25    1 0.33
[5,] 0.25    2 0.60
```

2. Para um modelo de 2 parâmetros:

```
> set.seed(2)
> theta2<-cbind(seq(-2,2,1),runif(5,.8,1.2))
> theta2

     [,1] [,2]
[1,]   -2 0.87
[2,]   -1 1.08
[3,]    0 1.03
[4,]    1 0.87
[5,]    2 1.18
```

3. Para um modelo de 1 parâmetro:

```
> theta1<-cbind(seq(-2,2,1),1)
> theta1

     [,1] [,2]
[1,]   -2    1
[2,]   -1    1
[3,]    0    1
[4,]    1    1
[5,]    2    1
```

Para simular dados considerando o modelo:

- Rasch utilize *theta1*;
- com 2 parâmetros utilize *theta2*;
- com 3 parâmetros utilize *theta3*.

Para um modelo de 3 parâmetros, as respostas de 10 respondentes podem ser obtidas da seguinte maneira.

```
> rmvlogis(10,theta3)
```

```
      [,1] [,2] [,3] [,4] [,5]
 [1,]    0    1    0    1    1
 [2,]    0    1    0    0    1
 [3,]    1    1    1    0    1
 [4,]    0    1    1    0    1
 [5,]    1    1    0    1    1
 [6,]    0    0    1    1    1
 [7,]    1    1    1    0    1
 [8,]    1    1    1    1    1
 [9,]    0    1    0    1    1
[10,]    1    1    1    1    1
```

Para um modelo de 2 parâmetros, as respostas de 10 respondentes podem ser obtidas da seguinte maneira.

```
> rmvlogis(10,theta2)
```

```
      [,1] [,2] [,3] [,4] [,5]
 [1,]    1    1    0    1    0
 [2,]    1    1    0    0    0
 [3,]    1    1    1    0    0
 [4,]    1    1    0    1    0
 [5,]    1    1    1    0    0
 [6,]    1    1    1    1    0
 [7,]    1    0    0    1    0
 [8,]    0    1    1    0    0
 [9,]    0    1    1    1    0
[10,]    1    0    1    0    0
```

E para um modelo de 1 parâmetro, as resposta de 10 respondentes podem ser obtidas da seguinte maneira.

```
> rmvlogis(10,theta1)
```

```
      [,1] [,2] [,3] [,4] [,5]
 [1,]    1    1    0    0    1
 [2,]    1    1    0    0    0
 [3,]    0    1    1    0    0
 [4,]    1    1    0    0    0
 [5,]    1    1    0    0    0
 [6,]    0    0    0    1    1
 [7,]    1    0    0    0    0
 [8,]    1    0    1    0    0
 [9,]    1    1    1    0    1
[10,]    1    1    1    0    1
```

5.3 Uma ilustração de simulação

Considere um conjunto de dados simulados com as seguintes características: 20 itens e 5000 respondentes. Observe nas linhas do programa os valores simulados para cada parâmetro e para a proficiência.

```
> a<-seq(.8,2,len=20)
> b<-seq(-2.0,2.0,length=20)
> c<-rep(.20,20)
> par<-cbind(a,b,c)
> set.seed(123)
> prof<-rnorm(5000,mean=0,sd=1.2) # 5000 respondentes
> dados<-sim(ip=par,x=prof)
```

No objeto `dados` estão as respostas simuladas por meio da função `sim()`:

```
> head(dados)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    0    1    1    1    1    0    1     1
[2,]    1    1    1    0    0    1    0    1    1     0
```

```
[3,]    1    1    1    1    1    1    1    1    1     1
[4,]    1    0    1    1    1    0    1    1    1     1
[5,]    1    1    1    1    1    1    1    1    1     0
[6,]    1    1    1    1    1    1    1    1    1     1
     [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19]
[1,]     0     1     1     0     0     0     0     0     0
[2,]     0     1     1     1     0     0     0     0     1
[3,]     1     1     1     1     1     1     1     1     0
[4,]     1     0     0     1     0     0     0     1     1
[5,]     1     1     0     0     1     0     0     0     0
[6,]     1     1     1     1     1     1     1     1     0
     [,20]
[1,]     1
[2,]     1
[3,]     1
[4,]     0
[5,]     0
[6,]     1
```

O objeto `par` contém os parâmetros que foram utilizados na simulação de respostas:

```
> par
```

```
         a     b   c
 [1,] 0.80 -2.00 0.2
 [2,] 0.86 -1.79 0.2
 [3,] 0.93 -1.58 0.2
 [4,] 0.99 -1.37 0.2
 [5,] 1.05 -1.16 0.2
 [6,] 1.12 -0.95 0.2
 [7,] 1.18 -0.74 0.2
 [8,] 1.24 -0.53 0.2
 [9,] 1.31 -0.32 0.2
[10,] 1.37 -0.11 0.2
```

```
[11,] 1.43  0.11 0.2
[12,] 1.49  0.32 0.2
[13,] 1.56  0.53 0.2
[14,] 1.62  0.74 0.2
[15,] 1.68  0.95 0.2
[16,] 1.75  1.16 0.2
[17,] 1.81  1.37 0.2
[18,] 1.87  1.58 0.2
[19,] 1.94  1.79 0.2
[20,] 2.00  2.00 0.2
```

Considere a estimação dos parâmetros (calibração) do modelo de 3 parâmetros:

```
> # Calibração
> dados.tpm<-tpm(dados)
> coef(dados.tpm)

        Gussng Dffclt Dscrmn
Item 1   0.346 -1.246   1.04
Item 2   0.113 -1.791   0.92
Item 3   0.233 -1.148   1.22
Item 4   0.257 -0.976   1.28
Item 5   0.318 -0.772   1.36
Item 6   0.247 -0.765   1.29
Item 7   0.089 -0.845   1.38
Item 8   0.260 -0.293   1.60
Item 9   0.168 -0.386   1.34
Item 10  0.194 -0.080   1.59
Item 11  0.182  0.024   1.80
Item 12  0.192  0.244   1.71
Item 13  0.204  0.420   1.89
Item 14  0.229  0.705   2.02
Item 15  0.166  0.772   1.73
Item 16  0.191  0.974   1.97
```

```
Item 17  0.196  1.138   2.33
Item 18  0.207  1.353   2.34
Item 19  0.184  1.496   2.12
Item 20  0.201  1.677   2.39
```

Compare os valores estimados com os valores simulados.

Na Figura 5.1 são apresentadas as curvas características dos itens.

```
> plot(dados.tpm)
```

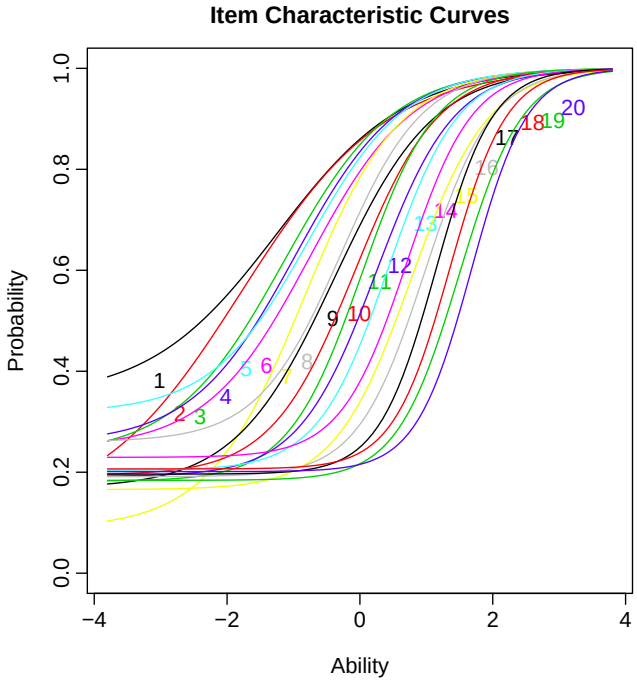

Figura 5.1: Curvas características dos itens.

A habilidade pode ser estimada com a função `factor.scores()`:

```
> theta<-factor.scores(dados.tpm, method = "EAP", prior = TRUE)
> head(theta$score.dat$z1,n=100) # 100 observações/indivíduos

  [1] -2.22 -2.35 -2.24 -2.25 -1.95 -1.99 -1.88 -2.02 -1.61
 [10] -1.89 -1.80 -1.89 -2.11 -1.92 -1.60 -1.75 -2.02 -2.03
 [19] -1.93 -1.87 -1.78 -1.91 -1.45 -1.20 -1.86 -1.73 -1.77
 [28] -1.68 -1.43 -1.64 -1.46 -1.30 -1.38 -2.03 -2.00 -1.76
 [37] -1.60 -1.92 -1.90 -1.60 -1.81 -1.74 -1.48 -1.53 -1.37
 [46] -1.82 -1.38 -1.19 -1.39 -1.31 -1.36 -0.67 -1.08 -1.47
 [55] -1.64 -1.61 -1.66 -1.68 -1.18 -1.55 -1.20 -1.33 -1.12
 [64] -1.02 -1.41 -1.46 -1.09 -0.92 -0.87 -0.55 -1.62 -1.43
 [73] -1.23 -1.42 -1.32 -0.20 -1.15 -0.87 -0.68 -0.56 -0.77
 [82] -0.67 -0.46 -0.42 -1.63 -1.59 -1.40 -1.30 -1.13 -1.61
 [91] -1.44 -1.44 -0.76 -1.30 -1.35 -1.01 -1.10 -1.12 -0.78
[100] -1.67
```

Na Figura 5.2 pode-se ver a distribuição dos valores de $\hat{\theta}$ com o posicionamento dos itens.

Suponha que 3 pessoas responderam o teste, cada uma com uma habilidade diferente, com os seguintes padrões de respostas:

```
> set.seed(1);A<-rbinom(20,1,.6)
> set.seed(2);B<-rbinom(20,1,.7)
> set.seed(3);C<-rbinom(20,1,.8)
> pessoas<-rbind(A,B,C);pessoas

  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
A    1    1    1    0    1    0    0    0    0     1     1
B    1    0    1    1    0    0    1    0    1     1     1
C    1    0    1    1    1    1    1    1    1     1     1
  [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
```

```
> par(mfrow=c(1,1))
> plot(theta,include.items=T)
```

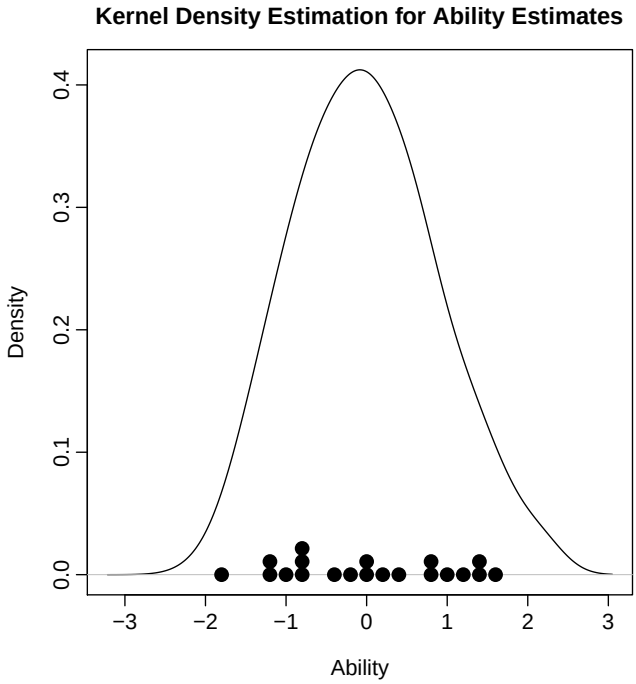

Figura 5.2: Gráfico das habilidades com os itens posicionados.

```
A     1     0     1     0     1     0     0     1     0
B     1     0     1     1     0     0     1     1     1
C     1     1     1     0     0     1     1     0     1
```

Estima-se a habilidade de cada pessoa dado o seu padrão de respostas com a função `factor.scores()`:

```
> theta.p<-factor.scores(dados.tpm, method = "EAP", resp.patterns=pessoas)
> theta.p


Call:
tpm(data = dados)

Scoring Method: Expected A Posteriori

Factor-Scores for specified response patterns:
  Item 1 Item 2 Item 3 Item 4 Item 5 Item 6 Item 7 Item 8
A      1      1      1      0      1      0      0      0
B      1      0      1      1      0      0      1      0
C      1      0      1      1      1      1      1      1
  Item 9 Item 10 Item 11 Item 12 Item 13 Item 14 Item 15
A      0       1       1       1       0       1       0
B      1       1       1       1       0       1       1
C      1       1       1       1       1       1       0
  Item 16 Item 17 Item 18 Item 19 Item 20 Obs   Exp     z1
A       1       0       0       1       0   0 0.001 -0.578
B       0       0       1       1       1   0 0.000 -0.059
C       0       1       1       0       1   0 0.105  1.157
  se.z1
A  0.61
B  0.44
C  0.40
```

Pode-se, a partir daí, posicionar as pessoas na escala de habilidade obtida (Figura 5.3):

```
> par(mfrow=c(1,1))
> plot(theta,include.items=T)
> text(theta.p$score.dat$z1,c(0.05,0.05,0.05),row.names(pessoas),col=2)
> abline(v=theta.p$score.dat$z1,col=2,lty=2)
```

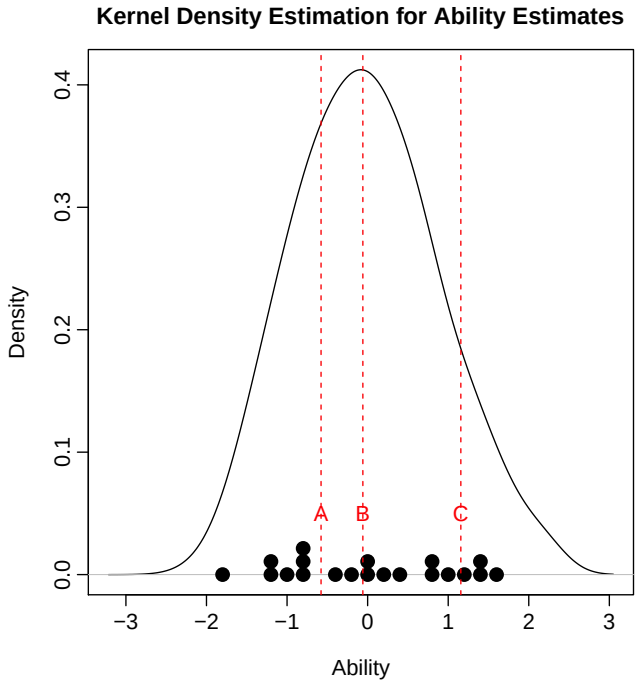

Figura 5.3: Gráfico das habilidades com os itens e pessoas posicionados.

Capítulo 6

Considerações gerais

Nesse curso, foram apresentados apenas problemas relacionados com modelos logísticos dicotômicos unidimensionais. Dentro dos pacotes apresentados nesse texto, ainda existem outras funções que podem ser utilizadas na análise de dados, segundo modelos da TRI, como por exemplo, a análise do funcionamento diferenciado do item (Diferential Item Function - DIF), modelos para dados politômicos e modelos multidimensionais. Na página do **R** , existem outros pacotes não apresentados nesse texto, que também podem ser utilizados. Por exemplo, o pacote `mirt` para análise de modelos multidimensionais.

Até o término desse texto, ainda havia algumas lacunas de funções não implementadas no **R** que poderiam ser objeto de estudos, como por exemplo, a ausência de funções para estimação de parâmetros em modelos nominais e equalização simultânea.

Espera-se que esse texto, apesar de introdutório, possa ser útil em cursos de Teoria da Resposta ao Item, principalmente com a utilização do **R** .

Referências Bibliográficas

ANDRADE, D. F. de; TAVARES, H. R.; VALLE, R. da C. *Teoria da resposta ao item: conceitos e aplicações*. São Paulo: ABE, 2000.

AYALA, R. J. de. *The Theory and Practice of Item Response Theory*. 1. ed. [S.l.]: The Guilford Press, 2009.

BAKER, F.; KIM, S. *Item response theory: Parameter estimation techniques*. [S.l.]: CRC, 2004.

EMBRETSON, S.; REISE, S. *Item response theory for psychologists*. [S.l.]: Lawrence Erlbaum, 2000.

FELLOWS, I. *Deducer: Deducer*. [S.l.], 2012. R package version 0.6-3. Disponível em: <http://CRAN.R-project.org/package=Deducer>.

GULLIKSEN, H. *Theory of Mental Tests*. [S.l.]: John Wiley & Sons Inc, 1950.

KOLEN, M. J.; BRENNAN, R. L. *Test Equating, Scaling, and Linking: Methods and Practices*. 2nd ed. 2004. ed. [S.l.]: Springer, 2010.

LORD, F. M. *Applications of Item Response Theory To Practical Testing Problems*. [S.l.]: Routledge, 1980.

LORD, F. M.; NOVICK, M. R. *Statistical theories of mental test scores*. [S.l.]: Addison-Wesley Pub. Co., 1968.

PARTCHEV, I. *irtoys: Simple interface to the estimation and plotting of IRT models*. [S.l.], 2010. R package version 0.1.3. Disponível em: <http://CRAN.R-project.org/package=irtoys>.

PASQUALI, L. *Psicometria*: teoria dos testes na psicologia e na educação. Petrópolis, RJ: Vozes, 2003.

R DEVELOPMENT CORE TEAM. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2012. Disponível em: <http://www.R-project.org/>.

RIZOPOULOS, D. ltm: An r package for latent variable modelling and item response theory analyses. *Journal of Statistical Software*, v. 17, n. 5, p. 1–25, 2006. Disponível em: <http://www.jstatsoft.org/v17/i05/>.

VIANNA, H. M. *Testes Em Educação*. [S.l.]: IBRASA, 1987.

WEEKS, J. P. plink: An R package for linking mixed-format tests using irt-based methods. *Journal of Statistical Software*, v. 35, n. 12, p. 1–33, 2010. Disponível em: <http://www.jstatsoft.org/v35/i12/>.

WILLSE, J. T.; SHU, Z. *CTT: Classical Test Theory Functions*. [S.l.], 2008. R package version 1.0.