Teoria da Resposta ao Item com uso do R

Adilson dos Anjos
Doutorando PPGEP
- UFSC -

Florianópolis, 2 de junho de 2011.

O objetivo dessa página é apresentar como utilizar o software **R** em análises da Teoria da Resposta ao Item (TRI).

Serão abordadas várias análises com exemplos resolvidos utilizando os diferentes pacotes disponíveis.

A versão do **R** utilizada foi:

```
> R.version$version.string
```

```
[1] "R version 2.13.0 (2011-04-13)"
```

1 Leitura de arquivos no R

Nessa primeira parte veremos como obter arquivos externos ao R. Existem variações nos tipos de arquivos de dados e como as respostas de questionários são inseridas neles. Serão abordadas três situações:

1. Arquivo de dados do BILOG com dados numéricos (0/1);
2. Arquivo de dados com respostas alfanuméricas (letras);
3. Arquivo de dados com extensão .xls.

1.1 Arquivo do BILOG

Inicialmente, vamos utilizar o arquivo de dados disponível no software BILOG: EXAMPL01.DAT.

Esse arquivo possui a seguinte estrutura:

```
1 242311431435242
2 243323413213131
3 142212441212312
4 341211323253521
5 343321411213113
6 341232413211114
```

Leitura do Arquivo no R:

```
> dados<-read.fwf('http://www.ufpr.br/~aanjos/TRI/dados/EXAMPL01.DAT',w=c(4,1,rep(
```

Para eliminar a coluna vazia basta fazer:

```
> dados<-read.fwf('http://www.ufpr.br/~aanjos/TRI/dados/EXAMPL01.DAT',w=c(4,1,rep(
> head(dados)
```

```
  V1 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17
1  1  2  4  2  3  1  1  4   3   1   4   3   5   2   4   2
2  2  2  4  3  3  2  3  4   1   3   2   1   3   1   3   1
3  3  1  4  2  2  1  2  4   4   1   2   1   2   3   1   2
4  4  3  4  1  2  1  1  3   2   3   2   5   3   5   2   1
5  5  3  4  3  3  2  1  4   1   1   2   1   3   1   1   3
6  6  3  4  1  2  3  2  4   1   3   2   1   1   1   1   4
```

O gabarito pode ser obtido de maneira semelhante:

```
> gab01<-read.fwf('http://www.ufpr.br/~aanjos/TRI/dados/EXAMPL01.KEY',w=c(3,2,rep(
> gab01
```

```
   V1 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17
1 KEY  3  4  1  4  2  1  3   2   3   4   4   1   4   1   3
```

As respostas podem ser recodificadas utilizando utilizando-se a função `mult.choice()` do pacote `ltm`:

Primeiro deve-se carregar o pacote `ltm`:

```
> library(ltm)
```

A função `mult.choice()` exige que o objeto que contém as resposta seja um vetro numérico. Na leitura do arquivo, a classe de `gab01` é data.frame, portanto, eve-se transformar esse objeto pra numérico. Ainda, no objeto que contém os dados a primeira coluna é o identificador. Logo, essa coluna não deve ser utilizada.

A recodificação dos dados pode ser realizada da seguinte forma:

```
> dados.resp<-mult.choice(dados[,2:16],as.numeric(gab01[,-1]))
```

Agora o objeto `dados.resp` contém dados na forma 0/1 que é o tipo de objeto requerido para a maioria das funções para análises em Teoria da Resposta ao Item com os pacotes do **R**.

```
> head(dados.resp)
```

```
     V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17
[1,]  0  1  0  0  0  1  0   0   0   1   0   0   0   0   0
[2,]  0  1  0  0  1  0  0   0   1   0   0   0   0   0   0
[3,]  0  1  0  0  0  0  0   0   0   0   0   0   0   1   0
[4,]  1  1  1  0  0  1  1   1   1   0   0   0   0   0   0
[5,]  1  1  0  0  1  1  0   0   0   0   0   0   0   1   1
[6,]  1  1  1  0  0  0  0   0   1   0   0   1   0   1   0
```

1.2 Leitura de dados - SARESP -

O arquivo com os dados está disponível com os seguintes comandos:

```
> saresp<-read.fwf('http://www.ufpr.br/~aanjos/TRI/dados/manha.dat',widths=c(12,3,
> colnames(saresp)<-c('id','turno',paste('i',1:30,sep=""))
```

Para ver os dados desse objeto digite:

```
> head(saresp)
> class(saresp)
```

O gabarito pode ser obtido de maneira semelhante:

```
> gabarito<-read.fwf('http://www.ufpr.br/~aanjos/TRI/dados/manha.dat',widths=c(-16
> gabarito
```

```
  V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16
1  A  B  D  C  A  A  B  C  D   C   A   A   D   B   D   D
  V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 V29 V30
1   A   B   D   D   D   C   D   C   A   D   A   D   B   C
```

```
> class(gabarito)
```

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

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 literais em números.

Para instalar opacote utilize o comando `install.packages('Deducer')` [1].

Para obter o pacote utilize:

```
> require(Deducer)

Please type JGR() to download/launch console. Launchers can also be obtained at htt



Note Non-JGR console detected:
        Deducer is best used from within JGR (http://jgr.markushelbig.org/).
                                                        To Bring up GUI dialogs, ty
```

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
```

Também, é necessário recodificar o gabarito:

```
> gab2<-recode.variables(gabarito,"'A'->1; 'B'->2; 'C'->3; 'D'->4")
```

[1] veja como instalar o pacote no Windows

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 respotas em 0 ou 1 de acordo com o gabarito.

```
> require(ltm)
```

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

```
> manha.NA<-mult.choice(dados[,3:32],as.numeric(gab2)) #ltm
```

```
> 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
[1,]   1   1   0   0   0   0   1   1   0   0   1   1   1
[2,]   1   1   1   1   0   1   0   0   0   1   1   1   1
[3,]   1   1   1   0   1   1   1   0   1   1   1   0   1
[4,]   0   1   0   0   0   1   0   1   0   0   1   0   0
[5,]   0   0   0   1   1   0   0   0   0   0   1   0   1
[6,]   0   1   1   1   1   0   0   0   0   0   1   0   1
     i30
[1,]   1
[2,]   1
[3,]   1
[4,]   0
[5,]   0
[6,]   1
```

1.3 Ausência de respostas

Observe que nesse conjunto de respostas existem 'missings' ou 'NA's'. Alguns pacotes conseguem realizar a estimação dos parâmetros considerando os 'missings' no arquivo

de respostas. 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<-ifelse(is.na(manha.NA)==T,0,manha.NA)
```

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

1.4 Leitura de dados a partir de uma planilha

So os dados estiverem em uma planilha, por exemplo, com a extensão `.xls`, pode-se utilizar o pacote `gdata` para leitura de dados.

Vamos carregr os dados de Satisfação (Disponível na página da disciplina TRI).

Primeiro é necessário carregar o pacote `gdata` (instale do site do R se necessário).

```
> require(gdata)
```

Obtendo o arquivo:

```
> satisf<-read.xls('http://www.ufpr.br/~aanjos/TRI/dados/DadossatisfacaoSil.xls')
> head(satisf)
```

```
  id.item X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14
1       1  1  1  1  1  1  1  0  0  1   0   1   1   0   1
2       2  1  1  1  1  1  1  0  1  1   1   1   1   1   1
3       3  1  1  1  1  1  0  1  1  1   1   1   1   0   0
4       4  0  0  1  1  1  1  1  1  1   1   0   1   1   1
5       5  1  1  1  1  1  1  0  1  1   1   1   1   0   1
6       6  1  1  1  1  1  1  0  1  1   1   1   1   1   1
  X15 X16 X17 X18 X19 X20 X21 X22 X23 X24 X25 X26 X27 X28
1   0   0   0   0   0   0   0   0   0   0   1   1   1   0
2   0   1   1   1   1   0   0   0   1   1   1   1   1   1
3   0   1   0   0   1   1   0   1   1   1   1   1   1   0
4   1   1   1   1   1   0   1   1   1   1   1   1   1   0
5   1   1   0   0   1   0   0   1   1   1   0   0   0   1
```

```
6   1   1   0   1   1   0   1   1   1   1   1   1   1   1
  X29 X30 X31 X32 X33 X34 X35
1   1   0   1   1   1   1   1
2   1   1   1   1   1   1   1
3   1   0   1   1   1   1   1
4   0   0   1   1   1   1   1
5   0   1   1   1   1   1   1
6   1   1   1   1   1   1   1
```

Observe que existem NA's nesse conjunto de observações.

1.5 Ler arquivos do SPSS

A função `read.spss()` do pacote `foreign` pode ser utilizada para leitura de arquivos do SPSS.

Para ler o arquico `Soldi` utilize a função da seguinte maneira:

```
> soldi<-read.spss('/home/adilson/Área de Trabalho/RTRI/dados/Soldi.sav',to.data.f
> head(soldi)

  Id Afeti_1 Afeti_2 Afeti_3 Afeti_4 Afeti_5 Afeti_6 Inst_1
1  1       0       0       0       0       0       0      1
2  2       0       1       0       0       1       0      1
3  3       0       1       0       0       0       0      0
4  4       0       0       0       0       0       0      1
5  5       1       1       0       0       0       1      0
6  6       0       0       0       0       0       0      1
  Inst_2 Inst_3 Inst_4 Inst_5 Inst_6 Norm_1 Norm_2 Norm_3
1      0      0      1      1      1      0      0      0
2      1      0      1      1      1      1      1      1
3      0      0      1      0      1      0      1      1
4      0      1      1      0      0      0      1      1
5      1      0      0      1      0      0      1      1
6      1      1      1      1      1      0      1      1
  Norm_4 Norm_5 Norm_6
1      0      0      0
2      0      1      0
```

```
3      0      0      0
4      1      1      0
5      1      1      1
6      0      1      0
```

1.6 Exercício

Ler no **R** os dados do questionário sobre Altura que estão disponíveis no seguinte endereço:

http://www.ufpr.br/~aanjos/TRI/dados/altura211.dat.

2 Testes clássicos

A função `descript()` dentro do pacote `ltm` realiza as principais análises clássicas.

```
> manha.desc<-descript(manha)
> names(manha.desc)

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

> manha.desc

Descriptive statistics for the 'manha' data-set

Sample:
 30 items and 1001 sample units; 0 missing values

Proportions for each level of response:
                     logit
i1  0.5754 0.4246 -0.3040
i2  0.4635 0.5365  0.1461
i3  0.6613 0.3387 -0.6693
i4  0.4755 0.5245  0.0980
```

```
i5  0.0929 0.9071  2.2786
i6  0.4286 0.5714  0.2877
i7  0.5734 0.4266 -0.2958
i8  0.6583 0.3417 -0.6559
i9  0.6234 0.3766 -0.5039
i10 0.5524 0.4476 -0.2106
i11 0.4076 0.5924  0.3739
i12 0.4466 0.5534  0.2146
i13 0.4565 0.5435  0.1743
i14 0.4196 0.5804  0.3245
i15 0.6953 0.3047 -0.8250
i16 0.7383 0.2617 -1.0370
i17 0.3736 0.6264  0.5167
i18 0.1339 0.8661  1.8672
i19 0.5415 0.4585 -0.1662
i20 0.2567 0.7433  1.0630
i21 0.6284 0.3716 -0.5252
i22 0.8012 0.1988 -1.3938
i23 0.7592 0.2408 -1.1485
i24 0.5165 0.4835 -0.0660
i25 0.6893 0.3107 -0.7969
i26 0.3536 0.6464  0.6030
i27 0.2847 0.7153  0.9212
i28 0.5544 0.4456 -0.2186
i29 0.2617 0.7383  1.0370
i30 0.1998 0.8002  1.3875


Frequencies of total scores:
     0 1 2 3 4 5 6  7  8  9 10 11 12 13 14 15 16 17 18 19
Freq 0 0 0 0 0 0 0 29 40 48 59 64 69 72 74 73 71 70 64 57
     20 21 22 23 24 25 26 27 28 29 30
Freq 51 45 35 28 21 14  9  5  2  1  0


Point Biserial correlation with Total Score:
    Included Excluded
i1    0.2386   0.1373
```

```
i2    0.4988   0.4137
i3    0.4107   0.3229
i4    0.4174   0.3250
i5    0.2499   0.1914
i6    0.2964   0.1973
i7    0.3649   0.2697
i8    0.2949   0.2001
i9    0.4071   0.3168
i10   0.3296   0.2317
i11   0.3358   0.2395
i12   0.2764   0.1759
i13   0.4695   0.3817
i14   0.4057   0.3136
i15   0.2025   0.1072
i16   0.3259   0.2397
i17   0.3433   0.2490
i18   0.2986   0.2313
i19   0.4090   0.3162
i20   0.2695   0.1813
i21   0.3289   0.2339
i22   0.1844   0.1017
i23   0.1609   0.0717
i24   0.1968   0.0930
i25   0.4048   0.3187
i26   0.4422   0.3559
i27   0.3310   0.2426
i28   0.5005   0.4159
i29   0.4089   0.3275
i30   0.3431   0.2657


Cronbach's alpha:
               value
All Items     0.7366
Excluding i1  0.7370
Excluding i2  0.7192
Excluding i3  0.7254
Excluding i4  0.7250
```

```
Excluding i5  0.7332
Excluding i6  0.7333
Excluding i7  0.7286
Excluding i8  0.7329
Excluding i9  0.7257
Excluding i10 0.7311
Excluding i11 0.7306
Excluding i12 0.7347
Excluding i13 0.7213
Excluding i14 0.7258
Excluding i15 0.7383
Excluding i16 0.7305
Excluding i17 0.7300
Excluding i18 0.7313
Excluding i19 0.7256
Excluding i20 0.7338
Excluding i21 0.7309
Excluding i22 0.7376
Excluding i23 0.7396
Excluding i24 0.7399
Excluding i25 0.7257
Excluding i26 0.7233
Excluding i27 0.7303
Excluding i28 0.7190
Excluding i29 0.7255
Excluding i30 0.7292
```

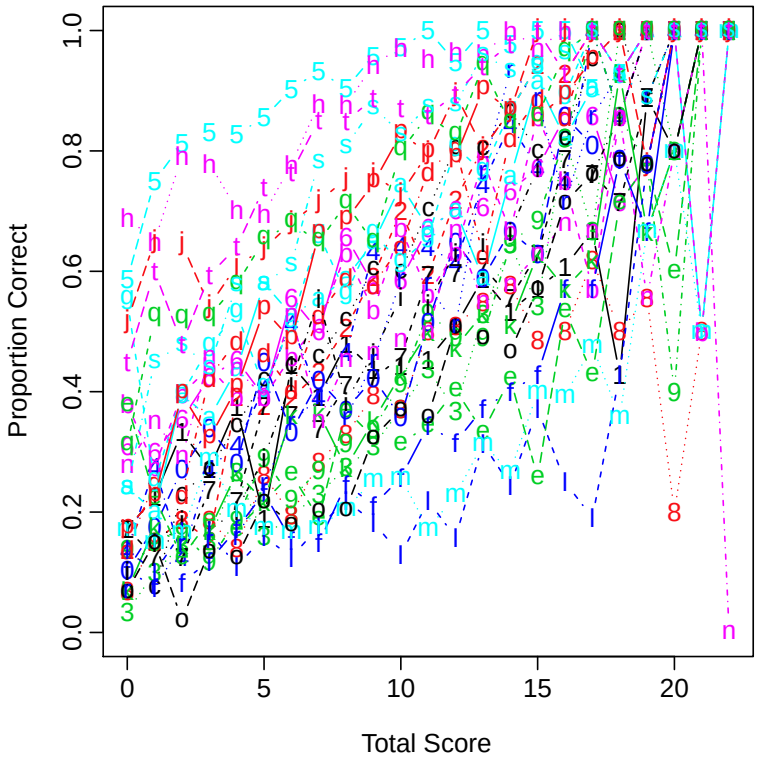
```
Pairwise Associations:
   Item i Item j p.value
1       6     15   0.976
2       1     11   0.972
3       5     16   0.969
4      11     23   0.968
5       1      6   0.963
6      18     23   0.959
7       9     15   0.958
8      18     24   0.957
```

```
9         1       21    0.953
10       20       23    0.949
```

Ainda, é possível fazer um gráfico com os resultados da função `descript()`.

Figura 1: Gráfico do item.

```
> plot(manha.desc,ty='b',includeFirstLast=TRUE)
```



A opção `includeFirstLast` indica que todos os scores devem ser inseridos no gráfico. Lembrando que é a frequência de ocorrência dos scores é que são inseridas nesse gráfico. Observe que há 8 valores que não ocorreram nesse conjunto de respostas. Os scores 0 até 6 e o score 30 (ninguém acertou todos os itens).

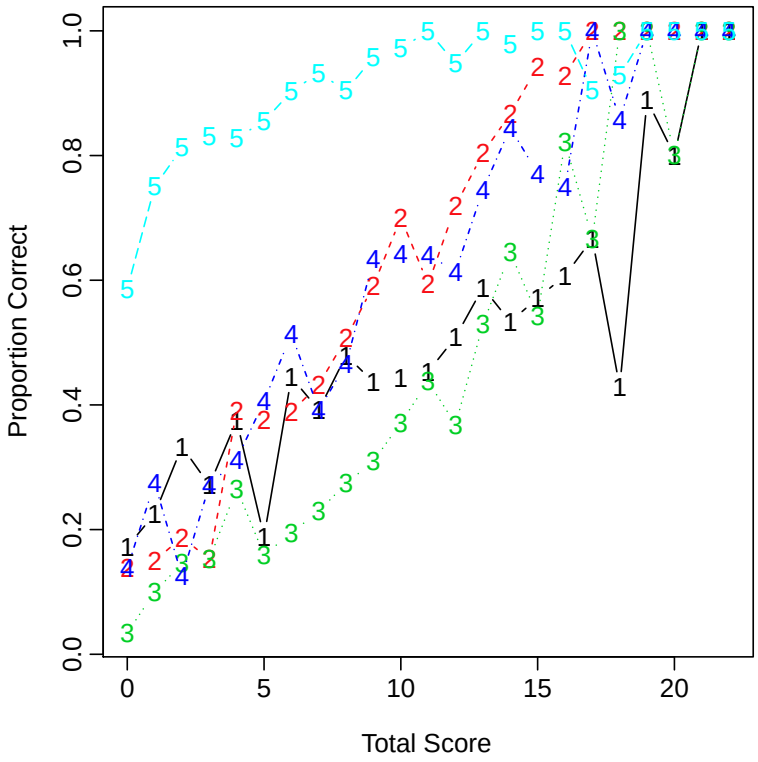
Quando há muitos itens, pode ser necessário realizar a análise por partes:

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

Primeiro, carregue o pacote:

Figura 2: Gráfico dos itens 1 a 5.

```
> plot(manha.desc,items=c(1:5),ty="b",includeFirstLast=TRUE,pch=c('1','2','3','4',
```




```
> require(CTT)
```

```
> manha.reliab<-reliability(manha)
> names(manha.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:

```
> manha.reliab$pbis
```

```
 [1] 0.13736404 0.41387410 0.32309618 0.32516840 0.19149991
 [6] 0.19744252 0.26985154 0.20017153 0.31698998 0.23183743
[11] 0.23962714 0.17598453 0.38193540 0.31378028 0.10723096
[16] 0.23981169 0.24916980 0.23141395 0.31633538 0.18138136
[21] 0.23405126 0.10171049 0.07174077 0.09309382 0.31882317
[26] 0.35605384 0.24276186 0.41611232 0.32768190 0.26585679
```

3 Uso do pacote ltm

O pacote `ltm` é um dos mais completos dentro do **R** para uso com TRI.

Nesse capítulo serão utilizadas algumas funções para modelos de TRI.

Utilizaremos o conjunto de dados `LSAT` (Law School Admission Test, Bock e Lieberman (1970)) disponível dentro do pacote `ltm`:

```
> head(data(LSAT))
```

```
[1] "LSAT"
```

Inicialmente, pode-se explorar os dados com a função `descript()`:

```
> descript(LSAT)
```

```
Descriptive statistics for the 'LSAT' data-set

Sample:
 5 items and 1000 sample units; 0 missing values

Proportions for each level of response:
           0     1  logit
Item 1 0.076 0.924 2.4980
Item 2 0.291 0.709 0.8905
Item 3 0.447 0.553 0.2128
Item 4 0.237 0.763 1.1692
Item 5 0.130 0.870 1.9010
```

```
Frequencies of total scores:
     0  1  2   3   4   5
Freq 3 20 85 237 357 298


Point Biserial correlation with Total Score:
       Included Excluded
Item 1   0.3618   0.1128
Item 2   0.5665   0.1531
Item 3   0.6181   0.1727
Item 4   0.5342   0.1444
Item 5   0.4351   0.1215


Cronbach's alpha:
                  value
All Items        0.2950
Excluding Item 1 0.2754
Excluding Item 2 0.2376
Excluding Item 3 0.2168
Excluding Item 4 0.2459
Excluding Item 5 0.2663


Pairwise Associations:
   Item i Item j p.value
1       1      5   0.565
2       1      4   0.208
3       3      5   0.113
4       2      4   0.059
5       1      2   0.028
6       2      5   0.009
7       1      3   0.003
8       4      5   0.002
9       3      4   7e-04
10      2      3   4e-04
```

No resultado da função `descript` é fornecida a proporçã ode respostas para cada item, a frequência de escores, a correlação bisserial e uma estatística χ^2 para associações pareadas entre os cinco items. Segundo Dimitris (2006), associações não significativas podem indicar itens problemáticos.

3.1 Ajuste do modelo de Rasch

A função `rasch()` do pacote `ltm` estima o parâmetro de discriminação. Para definir o parâmetro de discriminação como sendo 1, deve-se utilizar o argumento `constraint`.

Esse argumento deve ser fornecido na forma de uma matriz. Assim, para p itens o número $p+1$ indica o parâmetro de discriminação. Assim, `constraint=cbind(length(LSAT)+1,1)` indica que o parâmetro $p+1$, que corresponde ao parâmetro de discriminação, dve ser 1.

```
> fit1<-rasch(LSAT,constraint=cbind(length(LSAT)+1,1))
```

Para ver o resultado:

```
> summary(fit1)

Call:
rasch(data = LSAT, constraint = cbind(length(LSAT) + 1, 1))

Model Summary:
   log.Lik      AIC      BIC
 -2473.054 4956.108 4980.646

Coefficients:
                value std.err   z.vals
Dffclt.Item 1 -2.8720  0.1287 -22.3066
Dffclt.Item 2 -1.0630  0.0821 -12.9458
Dffclt.Item 3 -0.2576  0.0766  -3.3635
Dffclt.Item 4 -1.3881  0.0865 -16.0478
Dffclt.Item 5 -2.2188  0.1048 -21.1660
Dscrmn         1.0000      NA       NA

Integration:
```

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

Optimization:
Convergence: 0
max(|grad|): 6.3e-05
quasi-Newton: BFGS
```

Uma outra maneira de se obter as estimativas dos parâmetros é:

```
> coef(fit1,prob=TRUE, order=TRUE)

           Dffclt Dscrmn P(x=1|z=0)
Item 1 -2.8719712      1  0.9464434
Item 5 -2.2187785      1  0.9019232
Item 4 -1.3880588      1  0.8002822
Item 2 -1.0630294      1  0.7432690
Item 3 -0.2576109      1  0.5640489
```

Sem o argumento `constraint`, a função estima o parâmetro de discriminação.

```
> fit2<-rasch(LSAT)
> summary(fit2)

Call:
rasch(data = LSAT)

Model Summary:
   log.Lik      AIC      BIC
 -2466.938 4945.875 4975.322

Coefficients:
                value std.err   z.vals
Dffclt.Item 1 -3.6153  0.3266 -11.0680
Dffclt.Item 2 -1.3224  0.1422  -9.3009
Dffclt.Item 3 -0.3176  0.0977  -3.2518
Dffclt.Item 4 -1.7301  0.1691 -10.2290
```

```
Dffclt.Item 5 -2.7802  0.2510 -11.0743
Dscrmn         0.7551  0.0694  10.8757

Integration:
method: Gauss-Hermite
quadrature points: 21

Optimization:
Convergence: 0
max(|grad|): 2.5e-05
quasi-Newton: BFGS
```

OBS: utilizar ANOVA para comparar modelos.

Os valores de $\hat{\theta}$ podem ser obtidos com a função `factor.scores()`:

```
> factor.scores(fit2,met="EAP")

Call:
rasch(data = LSAT)

Scoring Method: Expected A Posteriori

Factor-Scores for observed response patterns:
   Item 1 Item 2 Item 3 Item 4 Item 5 Obs     Exp     z1
1       0      0      0      0      0   3   2.364 -1.910
2       0      0      0      0      1   6   5.468 -1.429
3       0      0      0      1      0   2   2.474 -1.429
4       0      0      0      1      1  11   8.249 -0.941
5       0      0      1      0      0   1   0.852 -1.429
6       0      0      1      0      1   1   2.839 -0.941
7       0      0      1      1      0   3   1.285 -0.941
8       0      0      1      1      1   4   6.222 -0.439
9       0      1      0      0      0   1   1.819 -1.429
10      0      1      0      0      1   8   6.063 -0.941
11      0      1      0      1      1  16  13.288 -0.439
12      0      1      1      0      1   3   4.574 -0.439
13      0      1      1      1      0   2   2.070 -0.439
```

```
14      0      1      1      1      1  15  14.749  0.084
15      1      0      0      0      0  10  10.273 -1.429
16      1      0      0      0      1  29  34.249 -0.941
17      1      0      0      1      0  14  15.498 -0.941
18      1      0      0      1      1  81  75.060 -0.439
19      1      0      1      0      0   3   5.334 -0.941
20      1      0      1      0      1  28  25.834 -0.439
21      1      0      1      1      0  15  11.690 -0.439
22      1      0      1      1      1  80  83.310  0.084
23      1      1      0      0      0  16  11.391 -0.941
24      1      1      0      0      1  56  55.171 -0.439
25      1      1      0      1      0  21  24.965 -0.439
26      1      1      0      1      1 173 177.918  0.084
27      1      1      1      0      0  11   8.592 -0.439
28      1      1      1      0      1  61  61.235  0.084
29      1      1      1      1      0  28  27.709  0.084
30      1      1      1      1      1 298 295.767  0.632
   se.z1
1  0.797
2  0.800
3  0.800
4  0.809
5  0.800
6  0.809
7  0.809
8  0.823
9  0.800
10 0.809
11 0.823
12 0.823
13 0.823
14 0.841
15 0.800
16 0.809
17 0.809
18 0.823
19 0.809
20 0.823
```

```
21 0.823
22 0.841
23 0.809
24 0.823
25 0.823
26 0.841
27 0.823
28 0.841
29 0.841
30 0.864
```

Após a calibração, pode-se utilizar um padrão de respostas qualquer para estimar a habilidade. Basta fornecer o padrão de repostas da seguinte forma:

```
> factor.scores(fit2, resp.patterns = rbind(c(1,0,1,0,1), c(NA,1,0,NA,1)))

Call:
rasch(data = LSAT)

Scoring Method: Empirical Bayes

Factor-Scores for specified response patterns:
  Item 1 Item 2 Item 3 Item 4 Item 5 Obs     Exp     z1
1      1      0      1      0      1  28  25.834 -0.466
2     NA      1      0     NA      1   0 252.440 -0.104
  se.z1
1 0.816
2 0.871
```

3.2 Ajuste do modelo com dois parâmetros

O ajuste do modelo com dois parâmetro pode ser obtido com a função `ltm()`.

Essa função pode ser especificada como uma fórmula onde o lado esquerdo deve conter os dados e do lado direito a variável latente z1:

```
> fit3<-ltm(LSAT~z1)
```

OBS: utilizar ANOVA para comparar modelos.

3.3 Ajuste do modelo com três parâmetros

Para ajustar um modelo com 3 parâmetros utiliza-se a função `tpm`)).

Será utilizado o conjunto de dados 'saresp manha' para estimar esse modelo

```
> manha.tpm<-tpm(manha,control=list(optimizer="nlminb"))
```

Os valores dos parâmetros estimados foram:

```
> manha.tpm

Call:
tpm(data = manha, control = list(optimizer = "nlminb"))

Coefficients:
     Gussng  Dffclt  Dscrmn
i1    0.243   1.979   0.632
i2    0.191   0.254   1.896
i3    0.156   1.114   1.717
i4    0.140   0.246   1.078
i5    0.024  -2.761   0.935
i6    0.025  -0.463   0.562
i7    0.098   0.786   0.812
i8    0.000   1.344   0.519
i9    0.175   1.043   1.520
i10   0.308   1.292   1.463
i11   0.005  -0.659   0.600
i12   0.323   0.972   0.771
i13   0.040  -0.104   1.169
i14   0.244   0.236   1.230
i15   0.265   2.298   1.741
i16   0.158   1.709   1.605
i17   0.493   0.798   2.322
i18   0.732   0.003   3.155
i19   0.076   0.434   0.956
i20   0.591   0.532   1.322
i21   0.000   0.975   0.581
```

```
i22   0.172   2.199   2.803
i23   0.215   2.352   2.245
i24   0.400   2.588   0.787
i25   0.002   0.996   0.959
i26   0.193  -0.244   1.362
i27   0.578   0.604   1.985
i28   0.067   0.361   1.518
i29   0.000  -1.089   1.219
i30   0.427  -0.616   1.365

Log.Lik: -17784.91
```

Os valores de θ podem ser obtidos com a função `factor.scores()` aplicada sobre o objeto da classe `tpm` que contém as estimativas dos parâmetros do modelo.

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

Os valores de *theta* estimados são:

```
> head(manha.prof$score)
```

```
  i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14 i15 i16
1  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0
2  0  0  0  0  0  0  0  0  0   0   0   1   0   1   1   0
3  0  0  0  0  0  0  0  0  0   1   0   0   0   0   1   0
4  0  0  0  0  0  0  0  0  1   1   0   0   0   0   1   0
5  0  0  0  0  0  0  0  1  0   0   0   1   0   1   0   0
6  0  0  0  0  0  0  0  1  0   1   0   0   0   0   1   0
  i17 i18 i19 i20 i21 i22 i23 i24 i25 i26 i27 i28 i29 i30
1   1   1   0   1   0   0   0   1   0   0   1   1   1   1
2   0   1   0   1   0   0   0   1   0   0   1   0   0   1
3   0   0   0   0   0   1   1   0   0   0   1   1   0   1
4   0   1   0   0   1   0   1   0   0   1   1   0   0   1
5   0   1   0   0   0   0   1   0   1   0   0   0   1   1
6   0   1   0   1   0   0   0   0   0   0   0   1   1   0
  Obs Exp        z1     se.z1
1   1   0 -1.254365 0.6110829
```

```
2    1    0 -1.800145 0.6841814
3    1    0 -1.840739 0.6910352
4    1    0 -1.526407 0.6573003
5    1    0 -0.976459 0.5065874
6    1    0 -1.330498 0.5644371
```

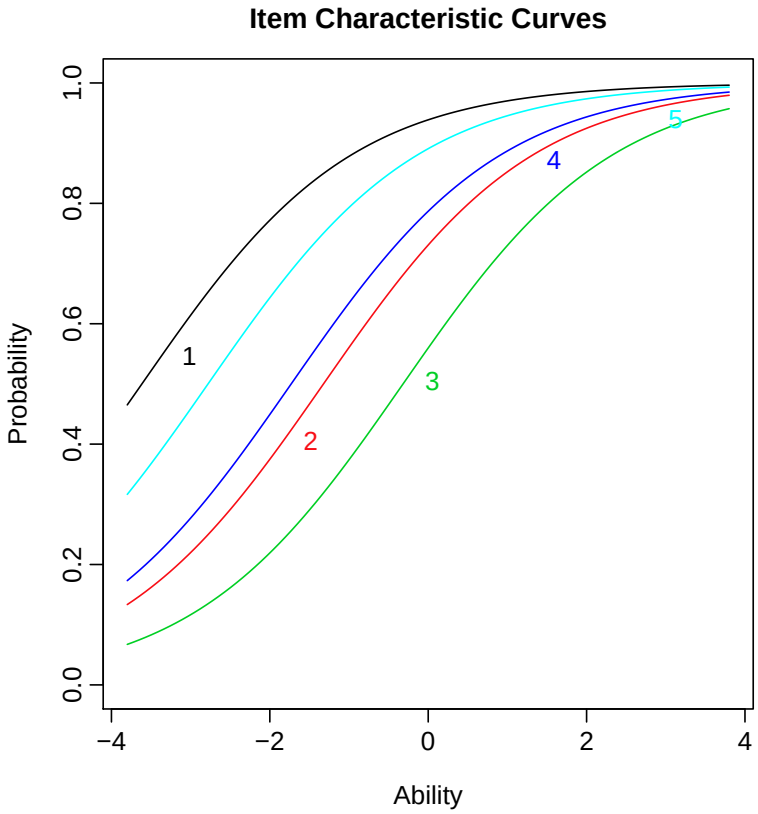
3.4 Gráficos

No **R** a função `plot()` pode assumir um comportamento específico para cada tipo de objeto.

Por exemplo, a função `plot` aplicada ao objeto `fit2` produz a CCI.

Figura 3: CCI para o modelo fit2.

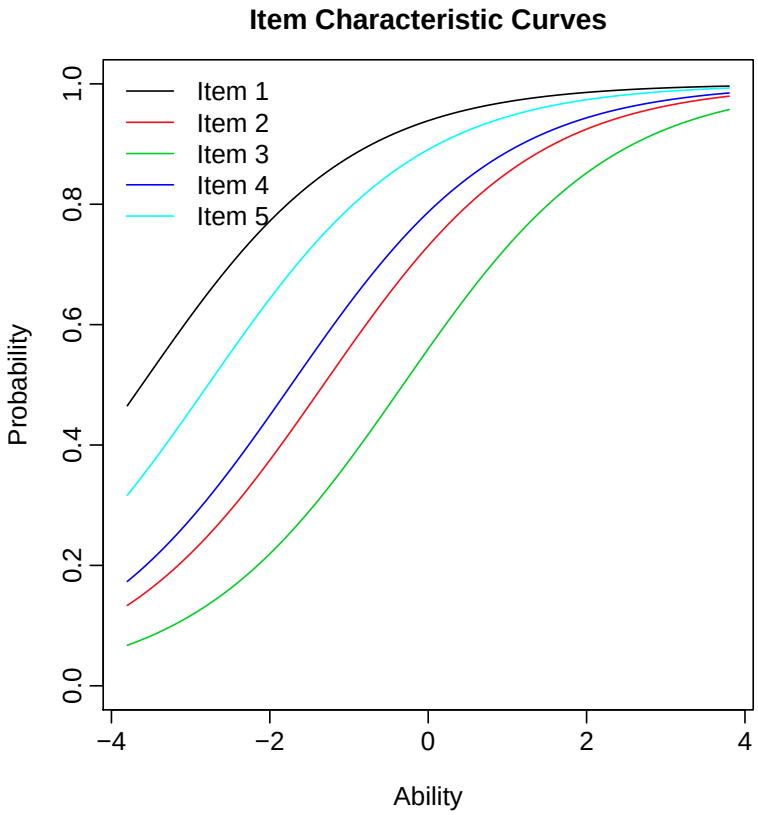
```
> plot(fit2)
```



ou

Figura 4: CCI para o modelo fit2.

```
> plot(fit2,legend=T)
```



As curvas de informação do item podem ser obtidas da seguinte maneira:

A função de informação do teste pode ser obtida da seguinte maneira:

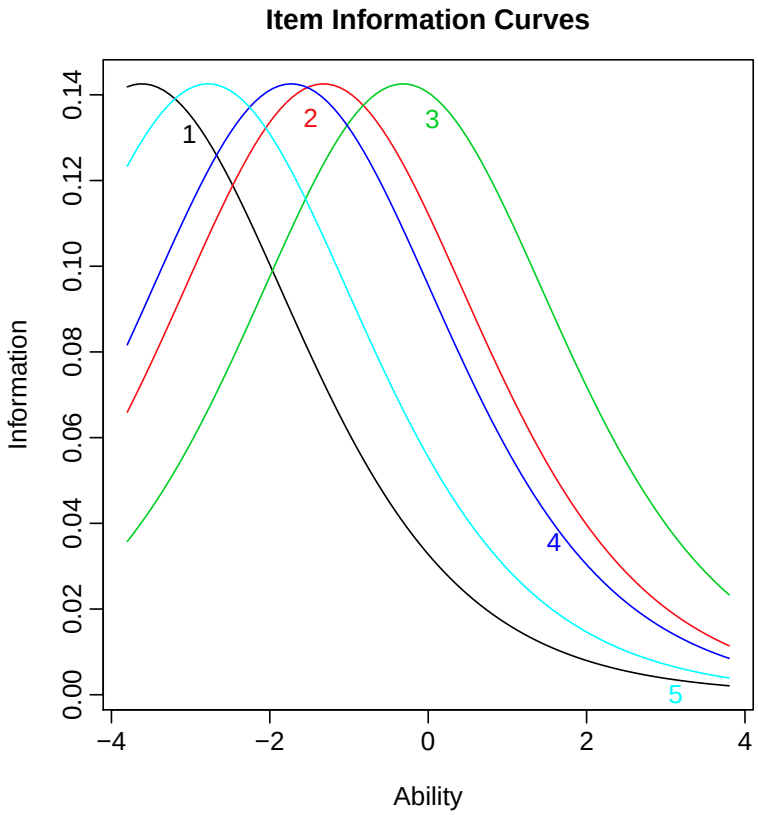
Um `plot` do objeto que contém a habilidade fornece um gráfico de densidade dessa variável.

3.5 Exercício

Utilize o conjunto de dados 'saresp noturno' e ajuste um modelo com 3 parâmetros.

Figura 5: CCI para o modelo fit2.

```
> plot(fit2,type='IIC')
```



3.6 Pacote irtoys

O pacote `irtoys` possui uma função chamada `est()` que possui características similares ao software BILOG quanto as opções de estimação. Inclusive, é possível utilizar as ferramentas de estimação do BILOG caso esse esteja instalado em seu computador.

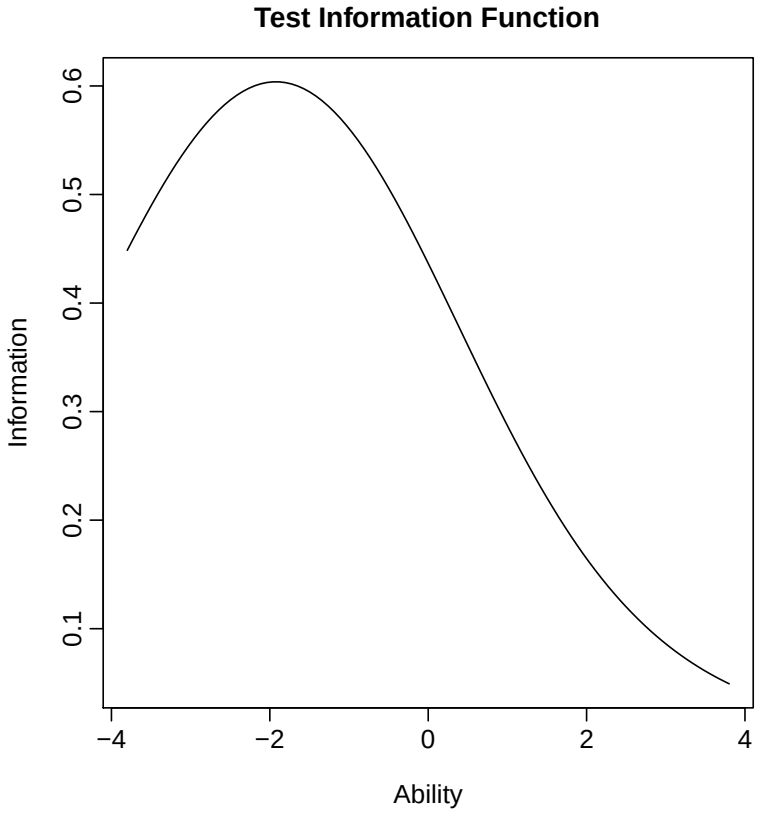
```
> require(irtoys)
```

```
Package `sm', version 2.2-3;  Copyright (C) 1997, 2000, 2005, 2007 A.W.Bowman & A.A
type help(sm) for summary information
```

Nesse material, não será abordado o BILOG. Será utilizado o pacote `ltm` como ferramenta de estimação dos parâmetros dos modelos de interesse.

Figura 6: TIF para o modelo fit2.

```
> plot(fit2,type='IIC',items=0)
```



Para mais informações sobre a função `est()` veja o arquivo de ajuda do pacote (help(est)).

O ajuste de modelos com três parâmetros pode ser obtido da seguinte maneira:

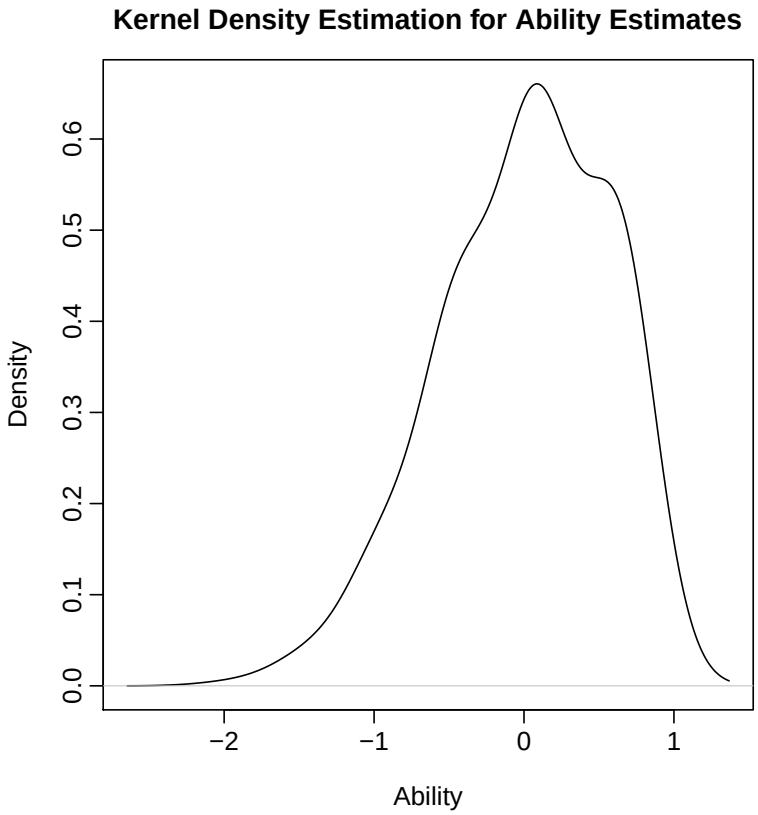
```
> manha.par<-est(manha, model = "3PL", engine = "ltm", nqp = 20, est.distr = FALSE,
+   logistic = TRUE, nch = 5, a.prior = TRUE, b.prior = FALSE, c.prior = TRUE,
+   bilog.defaults = TRUE, rasch = FALSE, run.name = "meumodelo")
```

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

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

Figura 7: Information para o modelo fit2.

```
> plot(factor.scores(fit2,met="EAP"))
```




```
        [,1]       [,2]       [,3]
i1 0.6644912  1.9939205 0.25346762
i2 1.9176240  0.2600441 0.19425154
i3 1.7516722  1.1147812 0.15854534
i4 1.0914463  0.2583877 0.14447929
i5 0.9412978 -2.7227852 0.04245911
i6 0.5695324 -0.4206178 0.03557638
```

Para obter o θ ou a proficiência ou habilidade de cada indivíduo, após a estimação dos parâmetros do modelo, pode-se utilizar a função `eap()`. É necessário fornecer o arquivo de respostas e o objeto com os parêmetros estimados do modelo:

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

O objeto `manha.sco` contém o vetor θ de cada indivíduo.

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

```
             est       sem  n
[1,]  0.06110404 0.4771474 30
[2,]  0.58837056 0.4089677 30
[3,]  1.92623806 0.4680314 30
[4,] -1.71143568 0.6046441 30
[5,] -0.47840008 0.4540995 30
[6,] -0.29040561 0.4498074 30
```

Um objeto final, com os scores, identificação e **posição** dos respondentes pode ser criado com os seguintes comandos:

```
> final.rank<-data.frame('id'=saresp$id,'turno'=saresp$turno,'score'=manha.sco[,1]
> head(final.rank)
```

```
           id turno       score posição acertos
1 11001138433   m07  0.06110404     538      16
2 11002964093   m07  0.58837056     748      17
3 11004154243   m07  1.92623806     987      26
4 11005367283   m07 -1.71143568      15       8
5 11007519633   m07 -0.47840008     326      12
6 11008054863   m07 -0.29040561     398      13
```

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

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

```
              id turno      score posição acertos
25   21002457753   m07 -0.9453425     156       7
88   41044200343   m07 -1.9417899       2       7
128  71025145903   m07 -1.6560666      18       7
149  81024867083   m07 -1.5962607      24       7
183  91057432493   m07 -1.8380660       7       7
192 101006567063   m07 -1.6005866      23       7
```

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

```
              id turno    score posição acertos
422 281032975343   m07 2.450968     999      27
803 711010603953   m07 2.348995     997      27
859 761011075233   m07 2.177786     993      27
229 121035651183   m07 2.604056    1000      28
628 511003326933   m07 2.360057     998      28
558 441003142213   m07 2.766691    1001      29
```

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

```
              id turno     score posição acertos
257 131061868283   m07 -1.942270       1       7
88   41044200343   m07 -1.941790       2       7
971 861004925633   m07 -1.905838       3       7
315 191001523743   m07 -1.888097       4       8
325 201008250873   m07 -1.857824       5      10
395 261017778043   m07 -1.847286       6       7
```

```
> tail(final.score)
```

```
              id turno    score posição acertos
409 271024133193   m07 2.335383     996      27
803 711010603953   m07 2.348995     997      27
628 511003326933   m07 2.360057     998      28
422 281032975343   m07 2.450968     999      27
229 121035651183   m07 2.604056    1000      28
558 441003142213   m07 2.766691    1001      29
```

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

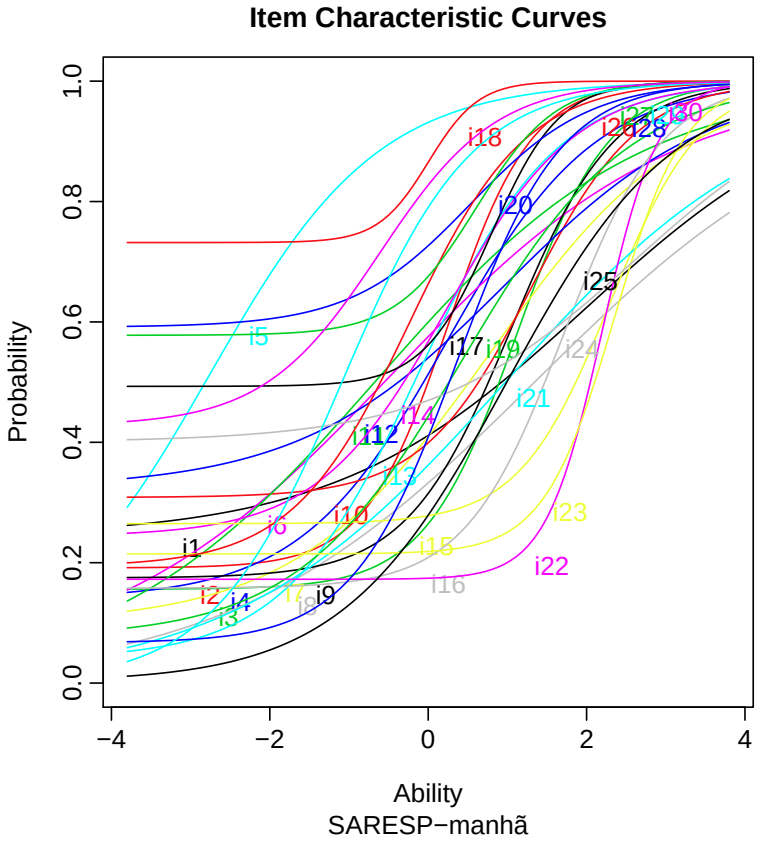
```
> novo.score<-score.transform(manha.sco[,1], mu.new = 500, sd.new = 10, normalize
> head(novo.score$new,n=30)
```

```
 [1] 500.6941 506.6993 521.9368 480.5060 494.5495 496.6906
 [7] 496.1962 496.0810 491.9442 507.9800 504.4469 499.2399
[13] 504.9959 495.8971 492.5432 490.3022 513.9534 511.7711
[19] 520.0580 510.6101 501.2948 505.0943 506.0797 505.7781
[25] 489.2313 513.4319 503.0251 514.0883 487.8618 504.2727
```

Para obtenção da curva característica pode-se obter um plot do objeto que contém o resultado da estimação dos parâmetros.

Figura 8: Curva característica dos itens.

```
> plot(manha.tpm,item=1:30,sub='SARESP-manhã',legend=F)
```

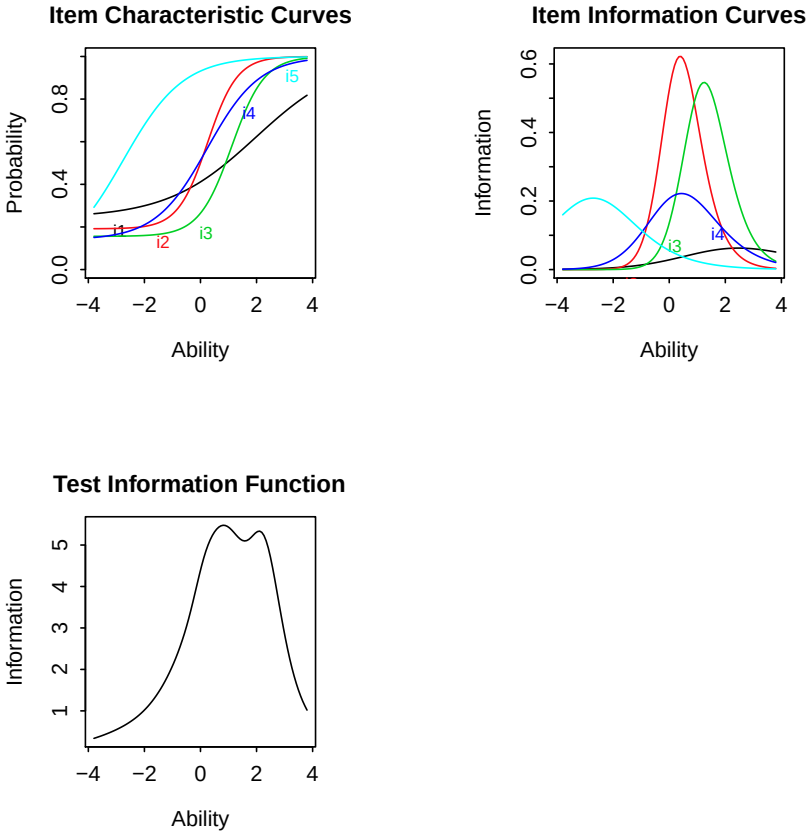



```
> summary(manha.tpm)

Call:
tpm(data = manha, control = list(optimizer = "nlminb"))
```

Figura 9: Curva característica dos itens.

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




```
Model Summary:
   log.Lik      AIC      BIC
 -17784.91 35749.82 36191.61

Coefficients:
             value std.err   z.vals
Gussng.i1   0.2429  0.1601   1.5173
Gussng.i2   0.1914  0.0715   2.6755
Gussng.i3   0.1560  0.0396   3.9397
```

```
Gussng.i4   0.1398  0.1256   1.1126
Gussng.i5   0.0243  0.8273   0.0294
Gussng.i6   0.0251     NaN      NaN
Gussng.i7   0.0982  0.1198   0.8192
Gussng.i8   0.0001  0.0065   0.0183
Gussng.i9   0.1748  0.0413   4.2321
Gussng.i10  0.3085  0.0486   6.3469
Gussng.i11  0.0049  0.1762   0.0278
Gussng.i12  0.3234  0.1220   2.6495
Gussng.i13  0.0400     NaN      NaN
Gussng.i14  0.2438  0.0981   2.4849
Gussng.i15  0.2649  0.0281   9.4235
Gussng.i16  0.1577  0.0326   4.8373
Gussng.i17  0.4930  0.0404  12.2168
Gussng.i18  0.7319  0.0414  17.6636
Gussng.i19  0.0756  0.1514   0.4992
Gussng.i20  0.5913  0.0759   7.7905
Gussng.i21  0.0001  0.0073   0.0160
Gussng.i22  0.1724  0.0166  10.3866
Gussng.i23  0.2147  0.0178  12.0598
Gussng.i24  0.4004  0.0700   5.7213
Gussng.i25  0.0015  0.0620   0.0242
Gussng.i26  0.1934  0.1306   1.4811
Gussng.i27  0.5778  0.0505  11.4507
Gussng.i28  0.0670  0.0554   1.2108
Gussng.i29  0.0002  0.0133   0.0180
Gussng.i30  0.4272  0.1537   2.7800
Dffclt.i1   1.9792  0.5517   3.5877
Dffclt.i2   0.2536  0.1613   1.5723
Dffclt.i3   1.1143  0.1128   9.8832
Dffclt.i4   0.2461  0.3465   0.7103
Dffclt.i5  -2.7609  1.4287  -1.9324
Dffclt.i6  -0.4627     NaN      NaN
Dffclt.i7   0.7861  0.3722   2.1121
Dffclt.i8   1.3443  0.2382   5.6444
Dffclt.i9   1.0433  0.1260   8.2807
Dffclt.i10  1.2917  0.1692   7.6354
Dffclt.i11 -0.6594  0.6338  -1.0405
```

```
Dffclt.i12  0.9722  0.5143   1.8905
Dffclt.i13 -0.1045     NaN      NaN
Dffclt.i14  0.2355  0.2848   0.8269
Dffclt.i15  2.2984  0.3702   6.2085
Dffclt.i16  1.7088  0.1657  10.3109
Dffclt.i17  0.7982  0.1462   5.4585
Dffclt.i18  0.0034  0.2019   0.0170
Dffclt.i19  0.4344  0.4160   1.0443
Dffclt.i20  0.5322  0.3892   1.3672
Dffclt.i21  0.9748  0.1749   5.5718
Dffclt.i22  2.1989  0.2248   9.7807
Dffclt.i23  2.3523  0.3034   7.7525
Dffclt.i24  2.5881  0.7172   3.6086
Dffclt.i25  0.9957  0.1644   6.0548
Dffclt.i26 -0.2439  0.3168  -0.7699
Dffclt.i27  0.6040  0.2160   2.7956
Dffclt.i28  0.3611  0.1299   2.7800
Dffclt.i29 -1.0891  0.1077 -10.1158
Dffclt.i30 -0.6164  0.4812  -1.2809
Dscrmn.i1   0.6319  0.4368   1.4468
Dscrmn.i2   1.8956  0.4190   4.5241
Dscrmn.i3   1.7169  0.4210   4.0781
Dscrmn.i4   1.0778  0.2710   3.9764
Dscrmn.i5   0.9354  0.2044   4.5769
Dscrmn.i6   0.5622     NaN      NaN
Dscrmn.i7   0.8122  0.2363   3.4365
Dscrmn.i8   0.5186  0.0843   6.1488
Dscrmn.i9   1.5202  0.3262   4.6604
Dscrmn.i10  1.4633  0.4762   3.0728
Dscrmn.i11  0.5996  0.1182   5.0711
Dscrmn.i12  0.7714  0.3154   2.4456
Dscrmn.i13  1.1690     NaN      NaN
Dscrmn.i14  1.2303  0.2896   4.2484
Dscrmn.i15  1.7405  0.9728   1.7892
Dscrmn.i16  1.6050  0.5065   3.1688
Dscrmn.i17  2.3217  0.7876   2.9476
Dscrmn.i18  3.1551  1.1588   2.7228
Dscrmn.i19  0.9562  0.2872   3.3297
```

```
Dscrmn.i20  1.3218  0.5227   2.5289
Dscrmn.i21  0.5815  0.0856   6.7911
Dscrmn.i22  2.8027  1.3812   2.0292
Dscrmn.i23  2.2450  0.9678   2.3197
Dscrmn.i24  0.7867  0.5440   1.4460
Dscrmn.i25  0.9592  0.2003   4.7884
Dscrmn.i26  1.3621  0.2909   4.6830
Dscrmn.i27  1.9852  0.7067   2.8091
Dscrmn.i28  1.5183  0.2473   6.1397
Dscrmn.i29  1.2185  0.1310   9.2982
Dscrmn.i30  1.3646  0.3426   3.9827

Integration:
method: Gauss-Hermite
quadrature points: 21

Optimization:
Optimizer: nlminb
Convergence: 0
max(|grad|): 0.012
```

A estimativa da proficiência de cada indivíduo pode ser obtida com a função `factor.scores()`:

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

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

```
4   0   1   0   0   1   0   1   0   0   1   1   0   0   1
5   0   1   0   0   0   0   1   0   1   0   0   0   1   1
6   0   1   0   1   0   0   0   0   0   0   0   1   1   0
  Obs Exp        z1     se.z1
1   1   0 -1.254365 0.6110829
2   1   0 -1.800145 0.6841814
3   1   0 -1.840739 0.6910352
4   1   0 -1.526407 0.6573003
5   1   0 -0.976459 0.5065874
6   1   0 -1.330498 0.5644371
```

```
> names(manha.prof)
```

```
[1] "score.dat" "method"    "B"         "call"
[5] "resp.pats" "coef"
```

4 Teste de Unidimensionalidade

Dentro do pacote `ltm` há uma função chamada `unidimTest()` que ode ser utilizada para testar a unidimensionalidade.

Utilize o arquivo `soldi`.

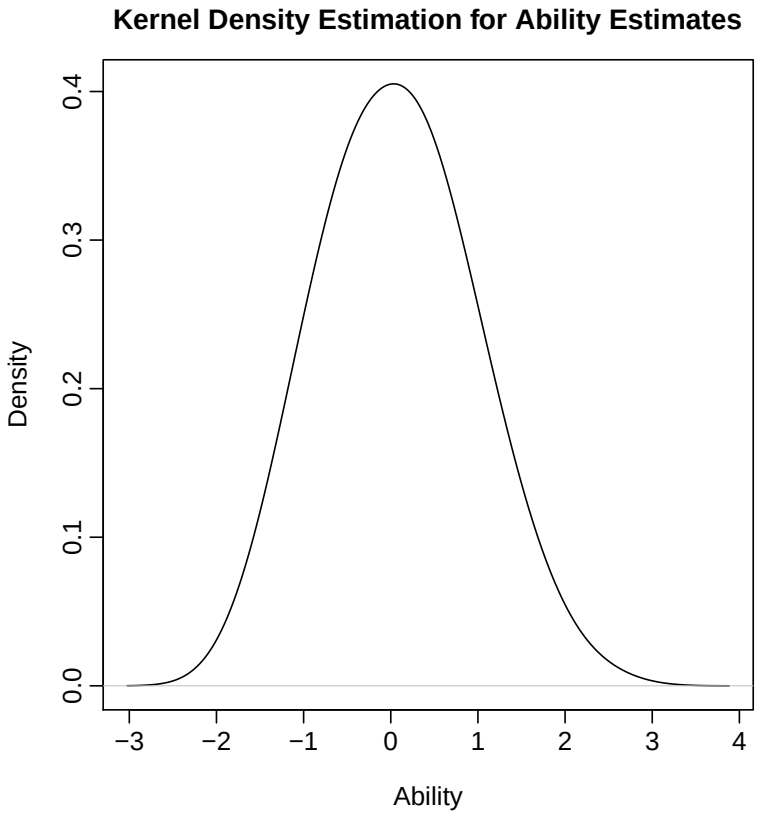
O arquivo pode ser obtido da seguinte maneira:

```
> soldi<-read.spss('/home/adilson/Área de Trabalho/RTRI/dados/Soldi.sav',to.data.f
> head(soldi)
```

```
  Id Afeti_1 Afeti_2 Afeti_3 Afeti_4 Afeti_5 Afeti_6 Inst_1
1  1       0       0       0       0       0       0      1
2  2       0       1       0       0       1       0      1
3  3       0       1       0       0       0       0      0
4  4       0       0       0       0       0       0      1
5  5       1       1       0       0       0       1      0
6  6       0       0       0       0       0       0      1
  Inst_2 Inst_3 Inst_4 Inst_5 Inst_6 Norm_1 Norm_2 Norm_3
1      0      0      1      1      1      0      0      0
```

Figura 10: Proficiência para o turno manhã.

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




```
2      1      0      1      1      1      1      1      1
3      0      0      1      0      1      0      1      1
4      0      1      1      0      0      0      1      1
5      1      0      0      1      0      0      1      1
6      1      1      1      1      1      0      1      1
  Norm_4 Norm_5 Norm_6
1      0      0      0
2      0      1      0
3      0      0      0
4      1      1      0
5      1      1      1
6      0      1      0
```

Considere inicialmente o modelo de Rasch.

```
> soldi.ras<-rasch(soldi[,-1])
```

Realizando o teste:

```
> unidimTest(soldi.rasch)
```

```
Alternative hypothesis: the second eigenvalue of the observed data is
substantially larger than the second eigenvalue
of data under the assumed IRT model

Second eigenvalue in the observed data: 2.5036
Average of second eigenvalues in Monte Carlo samples: 0.9551
Monte Carlo samples: 100
p-value: 0.0099
```

Experimente fazer o mesmo teste com os dados de LSAT do pacote `ltm`:

```
> unidimTest(rasch(LSAT))
```

5 Simulação de respostas no R

5.1 Simulação utilizando o pacote irtoys

Primeiro Carregue o pacote `irtoys`:

```
> require(irtoys)
```

Em seguida, gere os valores dos parâmetros de interesse. Considere, por exemplo 18 itens:

```
> set.seed(2345) #
> a<-runif(18,.2,3)
> b<-seq(-2,2,length=18)
```

```
> set.seed(321)
> #c<-runif(18,.10,.25) ou
> c<-rep(.33,18)
> pa<-cbind(a,b,c);pa
```

```
               a          b    c
 [1,] 0.5268817 -2.0000000 0.33
 [2,] 0.7460703 -1.7647059 0.33
 [3,] 2.1840811 -1.5294118 0.33
 [4,] 0.2964729 -1.2941176 0.33
 [5,] 1.5303363 -1.0588235 0.33
 [6,] 1.0236516 -0.8235294 0.33
 [7,] 1.8930629 -0.5882353 0.33
 [8,] 2.4077543 -0.3529412 0.33
 [9,] 1.3403296 -0.1176471 0.33
[10,] 2.1934367  0.1176471 0.33
[11,] 0.6460858  0.3529412 0.33
[12,] 1.1592440  0.5882353 0.33
[13,] 0.4269729  0.8235294 0.33
[14,] 0.6198062  1.0588235 0.33
[15,] 1.3493747  1.2941176 0.33
[16,] 1.1250785  1.5294118 0.33
[17,] 1.9192593  1.7647059 0.33
[18,] 1.4157916  2.0000000 0.33
```

Simule $\hat{\theta}$:

```
> set.seed(1236)
> pf<-rnorm(100)
```

Simule o padrão de resposta em função dos parâmetros e da habilidade:

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

Obtenha as estimativas de um modelo com 3 parâmetros,:

```
> require(ltm)
> dados.tpm<-tpm(dados.sim,constraint = cbind(1:18, 1, 0.33))
> coef(dados.tpm)
```

```
        Gussng      Dffclt    Dscrmn
Item 1    0.33 -1.61927398 0.6234960
Item 2    0.33 -2.09516792 0.6396838
Item 3    0.33 -2.30928419 1.6131551
Item 4    0.33 -0.29932386 0.8661321
Item 5    0.33 -1.35493738 1.5896167
Item 6    0.33 -3.15976690 0.4083628
Item 7    0.33 -0.94611590 1.1435501
Item 8    0.33 -0.80065098 2.1353497
Item 9    0.33  0.24145882 1.9554742
Item 10   0.33 -0.22772459 3.1784585
Item 11   0.33  0.06480098 0.4083328
Item 12   0.33  0.48424558 4.2162147
Item 13   0.33  0.80956878 0.8410591
Item 14   0.33  1.80068781 0.5857634
Item 15   0.33  1.33313218 0.9144555
Item 16   0.33  0.89631520 2.5332931
Item 17   0.33  1.56788678 1.8118367
Item 18   0.33  2.28620915 1.0794860
```

5.2 Simulação utilizando o pacote ltm

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

Para simular dados considerando o modelo de **Rasch** utilize a seguinte sintaxe, considerando 5 itens e 10 respondentes:

```
> rmvlogis(10,cbind(seq(-2,2,1),1))
```

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

```
 [8,]    1    0    1    0    0
 [9,]    1    1    0    0    0
[10,]    1    0    1    0    0
```

Veja a estrutura de cbind:

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

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

Para simular um padrão considerando um modelo de resposta gradual:

```
> thetas <- lapply(1:5, function(u) c(seq(-1, 1, len = 2), 1.2))
> rmvordlogis(10, thetas)
```

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

6 Questionário sobre Altura - Antigo

Estes dados referem-se ao questionário sobre Altura com 14 itens.

6.1 Leitura do arquivo

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

```
> altura<-read.fwf('http://www.ufpr.br/~aanjos/TRI/dados/altura211.dat',widths=c(3
> colnames(altura)<-c('id','altura',paste('i',1:14,sep="")) # insere nomes
```

Os dados necessitam ser formatados para que possam ser analisados no **R**.

```
> head(altura)
```

```
  id altura i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14
1  1   1,81  0  1  0  1  1  1  1  0  1   1   1   0   1   0
2  2   1,64  0  0  0  0  0  0  0  0  1   0   0   0   0   0
3  3   1,80  0  0  1  1  0  0  1  0  1   1   0   1   0   1
4  4   1,78  0  1  1  1  1  0  1  0  1   1   0   1   0   1
5  5   1,66  0  0  0  0  0  0  1  0  1   1   0   0   0   0
6  6   1,67  0  0  0  0  0  0  1  0  1   1   0   0   0   0
```

```
> class(altura)
```

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

6.2 Ajuste do modelo com dois parâmetros

No **R** o ajuste de modelos com dois parâmetros pode ser realizado com a função `ltm()`. Deve-se redefinir o argumento `constraint` para que o parâmetro c seja zero. Nessa opção, no 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[,3:16],const=cbind(1:14,1,0))
> altura.tpm
```

```
Call:
tpm(data = altura[, 3:16], constraint = cbind(1:14, 1, 0))
```

```
Coefficients:
     Gussng  Dffclt  Dscrmn
i1        0   2.537   0.298
i2        0   1.221   1.156
i3        0   1.223   1.477
i4        0   1.345   0.863
i5        0   1.076   2.196
i6        0   1.124   1.805
i7        0  -0.114   4.150
i8        0   2.636   1.101
i9        0  -0.813   1.131
i10       0   0.035   3.118
i11       0   0.405   1.172
i12       0   0.234   2.395
i13       0   0.808   1.748
i14       0   0.511   2.829

Log.Lik: -1425.466
```

Para obter os valores de θ basta aplicar a função `factor.scores()` sobre o objeto `altura.tpm`:

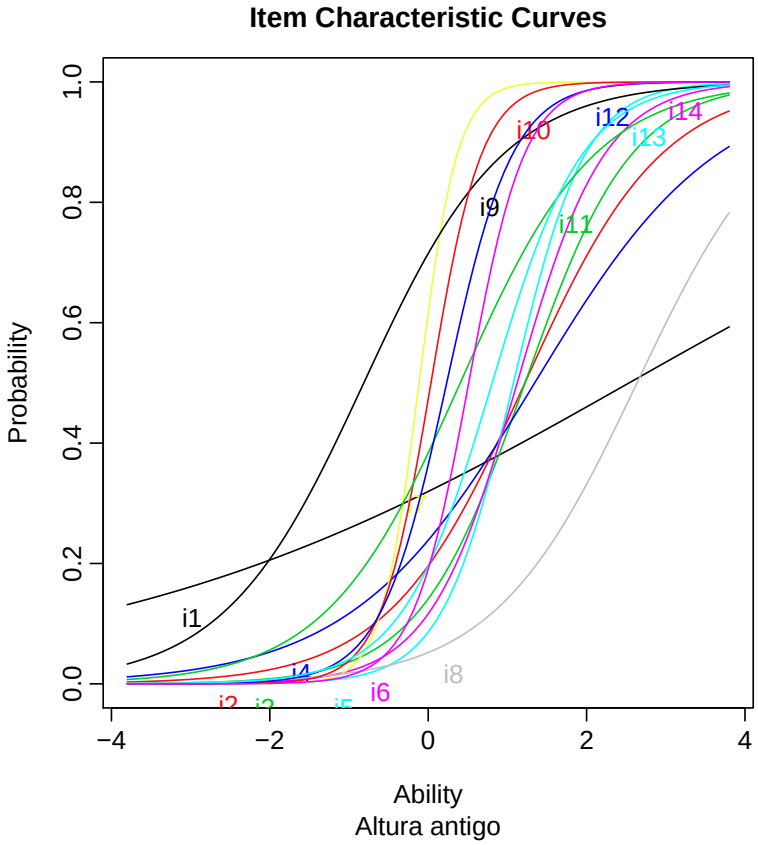
```
> altura.prof<-factor.scores(altura.tpm)
> head(altura.prof$score)

  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 -1.1923057 0.6324131
2 -0.3171599 0.3178183
3 -0.5874984 0.3974938
4 -0.3623446 0.3281457
5 -0.4275997 0.3453524
6 -0.8352894 0.4953336
```

As curvas características dos itens podem ser obtidas da seguite maneira:

Figura 11: Curva característica dos itens sobre altura.

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



Outras curvas também podem ser obtidas com os seguintes comandos:

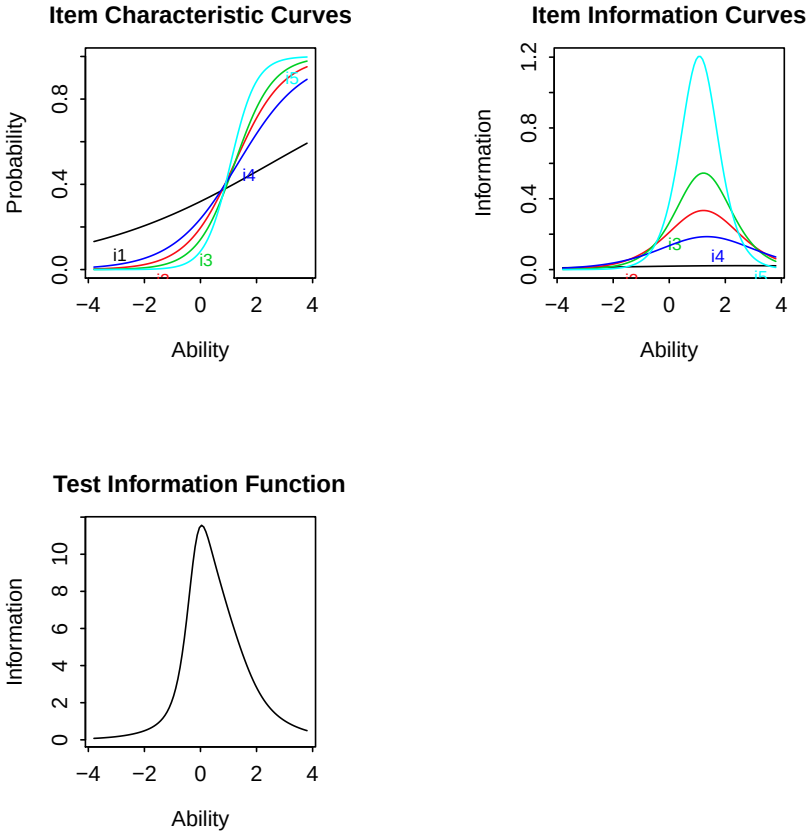
6.2.1 Análise pelo pacote irtoys

O modelo com dois parâmetros também pode ser obtido com a função `est()` do pacote `irtoys`:

```
> altura.par<-est(altura[,3:16], model = "2PL", engine = "ltm", nqp = 20, est.dist
+   logistic = TRUE, nch = 5, a.prior = TRUE, b.prior = FALSE, c.prior = FALSE,
+   bilog.defaults = TRUE, rasch = FALSE, run.name = "alturaR")
```

Figura 12: Curva característica dos itens sobre altura.

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



o objeto `altura.par` contém os resultados da função aplicda sobre os dados de altura.

```
> altura.par
```

```
          [,1]         [,2] [,3]
i1  0.3023088  2.51864451    0
i2  1.1786203  1.22417286    0
i3  1.4714661  1.24389140    0
i4  0.8919993  1.32955756    0
```

```
i5  2.2126637  1.09101933    0
i6  1.8039464  1.14264208    0
i7  4.0909896 -0.10296482    0
i8  1.1080365  2.64225344    0
i9  1.1431516 -0.78759448    0
i10 3.1866217  0.05536344    0
i11 1.1649784  0.42652777    0
i12 2.3735103  0.25586609    0
i13 1.7427149  0.82894898    0
i14 2.8101746  0.53301327    0
```

Os valores de θ para cada um dos indivíduos pode ser obtido com a função `eap()`.

```
> altura.sco<-eap(altura[,3:16],altura.par,qu=normal.qu())
```

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

```
             est       sem  n
[1,]  0.88403653 0.3476463 14
[2,] -0.99531379 0.5210093 14
[3,]  0.72098354 0.3221649 14
[4,]  1.13417926 0.3692381 14
[5,]  0.01571403 0.2818606 14
[6,]  0.01571403 0.2818606 14
```

6.2.2 Comparação com resultados do BILOG

Se você utilizou o software BILOG obter os parâmetros dos itens, observe que não são estimados exatamente os mesmos valores.

Para os dados de altura, comparamos os valores estimados do parâmetro it a pelo software **R** utilizando o pacote `ltm` e o software BILOG.

```
> a.bilog<-c(0.48201,1.13942,1.41464,0.89937,1.99707,1.70118,3.23086,1.08399,1.115
> a.r<-altura.tpm$coef[,3];a.r
```

```
       i1        i2        i3        i4        i5        i6
0.2978269 1.1556397 1.4769008 0.8628302 2.1961594 1.8050581
```

```
       i7        i8        i9       i10       i11       i12
4.1497827 1.1013725 1.1311536 3.1179914 1.1715320 2.3952802
      i13       i14
1.7481013 2.8288379
```

Figura 13: Associação entre valores estimados do parâmetro a no **R** e no BILOG.

```
> plot(a.r,a.bilog)
```