Apêndice – Manual de utilização dos programas

Apresentaremos, neste Apêndice, um manual de referência das implementações desenvolvidas neste trabalho. Neste manual são explicados em detalhes como configurar as implementações de modo a funcionarem corretamente no software **R** e descrevemos, também, as saídas geradas por cada programa, além de explicar como realizar a instalação e utilização do programa *EstatR.exe*.

A.1. *EstatR.exe*

Para a utilização do software *EstatR.exe* é necessário que o **R** já esteja instalado no computador e também fazer a instalação prévia do software RDCOM que pode ser baixado a partir do endereço:
http://cran.r-project.org/contrib/extra/dcom/RSrv135.exe. Este programa permite que o programa *EstatR.exe* e o **R** passem a se comunicar. A instalação deste software deve ser feita no mesmo diretório em que o **R** foi instalado (geralmente é **C:/Arquivos de programas/R**). Feito isso, basta abrir o programa *EstatR.exe*. Na janela que aparece, no canto superior esquerdo há um botão semelhante a um plug, conforme figura A.1. Deve-se clicar nesse botão de modo que o programa rode as implementações do **R**.

Figura A.1: *Plug* para o EstatR.exe se comunicar com o R.

Após isso, o *EstatR.exe* já está apto a executar as implementações. No menu lateral esquerdo, deve-se escolher a implementação que se deseja carregar. Clicando sobre a implementação desejada com o botão esquerdo do mouse, aparecerá no campo direito uma tela contendo duas guias na parte superior. A primeira contém o nome do programa e algumas instruções de uso. A outra guia é

chamada de **Detalhes** e contém o nome da pasta que armazena o programa a ser executado.

Em todas as implementações, o usuário deve fazer algumas configurações iniciais de parâmetros a serem utilizados e do local (diretório) em que se encontram os arquivos de entrada e onde serão salvos os arquivos de saída. Para se fazer tais configurações, o *EstatR.exe* possui um recurso de edição dos arquivos onde estão os programas desenvolvidos. Basta clicar com o botão direito do mouse sobre a implementação que se deseja rodar e selecionar, no menu que se abre, a opção **Configurar**. Uma janela será aberta contendo, no campo da direita o programa que deverá ser editado. Após fazer as alterações necessárias, deve-se clicar no botão **Salvar** que se encontra na parte inferior direita da tela. Feito isso, a janela irá se fechar e a tela principal do programa voltará a ser exibida.

Para rodar a implementação há duas possibilidades: na guia que contém o nome do programa, clique no botão **Executar** no canto inferior direito da tela. Alternativamente, pode-se clicar sobre o nome da simulação com o botão direito do mouse (no menu lateral esquerdo) e selecionar a opção **Executar**. No *EstatR.exe* o console do **R** não é exibido em nenhum momento, fazendo com que o usuário só saiba o que está ocorrendo ao longo do processamento através de janelas de comunicação emitidas pelos programas que estão sendo executados.

A.2. Configuração dos arquivos de entrada

Cada um dos programas desenvolvidos tem como requisito para ser executado de um a seis arquivos de entrada que devem ser criados pelo usuário e atribuídos exatamente os mesmos nomes como citaremos a seguir. Alguns arquivos podem tanto ser criados pelo usuário como podem ser gerados automaticamente a partir de algumas das implementações, conforme citaremos mais adiante. Como já foi dito, pelo fato de se tratar do uso de 2 testes ($\mathbf{X}$ e $\mathbf{Y}$), alguns arquivos, embora

tenham nomes diferentes, possuem mesma configuração. A diferença entre eles está apenas nos conteúdos, mas não no modo de configurá-los. Assim, os exemplos apresentados se referem a um teste **X** (note que os arquivos correspondentes a este teste têm sempre o nome com a terminação x – *aaax.txt,* enquanto que para o teste **Y** os arquivos tem sempre a terminação y – *aaay.txt*). Para o teste **Y**, a configuração é idêntica.

A.2.1. Arquivos *entradax.txt* e *entraday.txt*

Estes arquivos são básicos e necessários (ao menos um deles) em todas as implementações. Eles contêm dados sobre o teste, tais como informações dos itens e respostas dos alunos. As respostas destes arquivos podem ser dados numéricos (0 indicando item errado ou 1 para item correto) ou literais (A,B,C,D,E...). A limitação para o uso de respostas literais é de até 20 letras, ou seja, de A até T. Na figura A.2 apresentamos um exemplo contendo parte de um arquivo literal, baseados em dados do SARESP, chamado *entradax.txt* .

```
ANSWER,C,C,C,C,C,C,C,C,B,B,C,D,C,C,C,D,A,C,C,C,C,A,D,A,B,B,D,D,C,D,B,D
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4
S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S
Aluno,29,23,31,32,33,34,35,36,24,11,13,14,37,38,25,26,27,39,40,41,17,18,19,20
,21,1,3,4,7,28,15,16
13118031,C,C,C,C,C,C,C,C,B,B,A,B,B,B,C,D,A,C,C,C,B,B,D,A,B,B,D,D,C,A,B,B
13118021,D,C,C,B,B,C,C,C,C,B,D,A,B,B,A,D,C,B,C,B,B,A,A,A,B,B,A,B,C,A,A,C
13118011,A,A,C,C,C,C,C,C,B,D,C,B,B,C,A,B,B,C,C,C,B,A,D,A,A,D,*,B,D,B,C,A
13115251,C,C,C,C,C,C,C,C,A,B,C,D,C,C,C,A,B,C,C,C,B,A,D,C,C,B,D,B,C,C,A,A
13115241,C,B,C,C,C,C,C,C,A,B,A,D,B,C,D,C,B,C,B,C, , , , , ,D,D,C,C,C,A,A
13111251,C,C,C,C,C,C,C,C,B,B,C,D,C,B,B,D,A,B,C,C,C,A,D,A,B,B,D,D,A,C,C,D
13111241,D,C,C,C,B,C,C,C,B,B,C,A,C,C,C,D,C,C,C,C,B,A,D,A,A,B,D,D,C,B,D,D
13111231,A,C,C,C,C,C,C,C,B,D,C,C,C,C,C,A,A,B,C,C,B,D,C,A,D,A,A,C,C,A,D,
13111221,D,C,C,C,C,C,C,C,B,B,D,D,C,B,*,C,A,C,C,*,B,A,D,B,D,B,D,D,A,B,A,A
13111211,D,C,C,C,B,C,C,C,A,B,C,A,B,C,C,D,A,B,C,C,A,A,D,C,D,B,D,A,D,C,A,D
13111201,D,C,C,C,C,C,C,C,B,B,C,A,C,C,C,B,A,C,C,C,B,A,D,A,B,B,D,D,A,C,D,D
```

Figura A.2: Exemplo de um arquivo literal *entradax.txt.*

Nesse exemplo podemos observar como devem ser configurados esses arquivos:

- Na primeira linha, o nome do gabarito, no caso, “ANSWER” e as alternativas corretas de cada item separadas por vírgula (C,C,C,...).
- Na segunda linha deve ser informado o número de alternativas de cada um dos itens, ou seja, se um item contém as alternativas A,B,C e D, o número a ser informado é 4, que é o caso do nosso exemplo. É possível termos um teste que contenha itens com número de alternativas diferentes, por exemplo, se o primeiro item tiver 5 alternativas, o segundo 6, o terceiro 4 e o quarto item contiver 5 alternativas, essa linha deverá ser 5,6,4,5. Os programas reconhecem quantas são as alternativas e emitem análises fazendo essas considerações. Como já dissemos, o número máximo de alternativas é igual a 20. Caso ultrapasse esse valor, os programas não conseguirão realizar as análises e uma janela de erro será emitida ao usuário.
- Na terceira linha, devemos colocar, para cada item, separado por vírgulas e em letras maiúsculas as letras S ou N que representam uma resposta Sim ou Não à questão: “O item deve ser considerado na análise?”. Este recurso é bastante interessante visto que permite facilmente incluir ou excluir itens das análises. Assim, um item ruim, pode ser removido sem a necessidade de ser criar um novo arquivo de entrada.
- Na quarta linha colocamos um nome para a identificação da coluna dos respondentes (no caso, “Aluno”) e, em seguida, separado por vírgulas, os nomes de cada um dos itens a serem analisados (29, 23, 31,...). Deve-se tomar o cuidado de atribuir, no arquivo *entradax.txt* e *entraday.txt*, os mesmos nomes para os itens comuns (ou itens âncoras). Em contrapartida, deve-se ter cuidado de atribuir nomes diferentes a itens distintos em cada um dos dois arquivos, visto que os programas entendem que itens com mesmo nome deve ser tratados como comuns. Se em um dos arquivos tivermos um item chamado 5 que será considerado na análise (“S” na terceira linha), mas no arquivo correspondente ao outro teste tivermos um item também chamado 5 mas que deve ser desconsiderado da análise (“N” na terceira linha), os programas entenderão o item chamado 5 como sendo um item <u>não comum</u>.

- Da quinta linha em diante, deve ser colocado o nome ou identificação do respondente (no caso da quinta linha, 13118031) seguido das alternativas marcadas por esse indivíduo (C,C,C,C,...). Dois aspectos são muito importantes: caso o indivíduo tenha deixado a questão em branco ou tenha assinalado mais de uma alternativa, o que invalidaria esse item para esse aluno, deve-se deixar um espaço em branco separado por vírgulas ou colocar um caractere alternativo, como um asterisco (*). Por exemplo, podemos observar que o quinto indivíduo (identificado como 13115241) deixou em branco alguns itens e que o terceiro indivíduo (identificado como 13118011) anulou (veja o asterisco) o item identificado como 3. Outro ponto importante é que os programas criados diferenciam maiúsculas de minúsculas, ou seja, “A” é diferente de “a”. Por isso, é necessário que os gabaritos bem como as respostas dos indivíduos estejam todos em maiúsculas ou todos em minúsculas a fim de serem reconhecidos pelas implementações.

Uma outra possibilidade de configuração dos arquivos de entrada é a utilização de “zeros e uns”, muito útil quando já se tem o teste corrigido (0 representa resposta incorreta e 1, resposta correta). Neste caso, a configuração do arquivo é a mesma, conforme ilustra a Figura A.3. Para este exemplo, utilizamos um outro conjunto de dados que não corresponde àquele apresentado na Figura A.2. Porém, algumas considerações devem ser feitas:

- na primeira linha, deve contar o nome do gabarito seguido do mesmo, que no caso são os números “1” (que indicam que o item foi acertado);
- na segunda linha, o número de alternativas deve ser igual a 2, visto que estamos considerando apenas certo ou errado;
- da quinta linha em diante, nas respostas dos indivíduos, itens que foram deixados em brancos ou foram anulados pelo respondente devem ser corrigidos como “0” (item incorreto), não devendo aparecer outros caracteres nem espaços em branco.

```
Gaba,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
S,S,N,S,S,N,S,S,S,S,S,S,N,S,S,S,N,S
Aluno,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,I17,I18
Aluno1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
Aluno2,1,1,1,0,0,1,0,0,0,0,0,1,0,0,1,0,1,0
Aluno3,1,1,1,0,1,1,0,1,0,0,1,0,0,0,0,0,0,0
Aluno4,1,0,0,1,1,1,1,0,0,0,0,0,1,1,0,0,0,1
Aluno5,1,1,1,0,1,1,0,1,1,1,1,0,1,1,0,0,0,0
Aluno6,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,1,0,0,1
Aluno7,1,0,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1
Aluno8,1,1,1,0,1,1,1,1,1,1,1,0,0,0,1,0,1,1,0
Aluno9,1,1,1,0,1,1,1,1,1,1,0,0,0,0,1,0,0,1,0
Aluno10,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0
```

Figura A.3: exemplo de arquivo numérico *entradax.txt.*

A.2.2. Arquivos *parametrosx.txt* e *parametrosy.txt*

Esses arquivos são necessários apenas para se realizar equalizações na TRI e contém os parâmetros de cada um dos itens considerados em uma análise. Esse arquivo pode ser criado pelo usuário ou pode ser gerado através da implementação que realiza a calibração dos parâmetros dos itens. A configuração deste arquivo consiste em colocar em cada linha os parâmetros dos itens, previamente calibrados em uma escala qualquer, de um modelo logístico com 3 parâmetros, separados por vírgulas e os nomes dos itens. Ou seja, nas três primeiras colunas temos os parâmetros: índice de discriminação (a), índice de dificuldade (b) e acerto casual (c). Na quarta coluna, os nomes dos itens, conforme ilustra a Figura A.4. Este arquivo, quando gerado por outro programa (e não pelo usuário) pode apresentar uma quantidade de casas decimais elevadas. Tais resultados, entretanto, não serão mostrados ao usuário e são usados apenas internamente, em outras implementações.

```
1.19065000300047    ,1.05894362196974    ,0.0577525312294606  ,29
1.63170324934713    ,-1.15064544562281   ,0.0249504056896564  ,23
1.27101386879851    ,-0.350566819487568  ,0.0916349908420657  ,33
1.51335134064236    ,-1.57325792844795   ,0.499569186021201   ,34
1.49029859844648    ,-1.8658995149446    ,0.445946237744564   ,35
1.42016428579667    ,-2.15817539509318   ,0.3630521156034     ,36
0.967888483683066   ,0.110143379298512   ,0.0777770051239651  ,24
1.87384011095949    ,-0.820206927196946  ,0.0277360174494469  ,11
```

Figura A.4: Arquivo de exemplo *parametrosx.txt.*

Uma configuração importante deste arquivo é que, nele, devem constar apenas os parâmetros dos itens considerados na análise. Assim, por exemplo, suponhamos que a configuração de *entradax.txt* fosse feita conforme a Figura A.5:

```
ANSWER,C,C,C,C,C,C,C,C,B,B,C,D,C,C,C,D,A,C,C,C,C,A,D,A,B,B,D,D,C,D,B,D
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4
S,S,N,N,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S,S
Aluno,29,23,31,32,33,34,35,36,24,11,13,14,37,38,25,26,27,39,40,41,17,18,19,20
,21,1,3,6,7,28,15,16
```

Figura A.5: Suposição de configuração para *entrada.txt*.

No arquivo de entrada apresentado na Figura A.5 temos que o item 31 e 32 não devem ser considerados na análise. Então, nesse caso, na Figura A.4 teríamos que a linha 1 corresponde aos parâmetros do item 29, a linha 2 ao item 23, a linha 3 ao item 32, a linha 4 ao item 34, a linha 5 ao item 35 e assim por diante, ou seja, não devem constar os itens desconsiderados na análise.

A.2.3. Arquivos *habilidadesx.txt* e *habilidadesy.txt*

Estes arquivos possuem os valores das habilidades estimadas para cada um dos indivíduos na mesma métrica dos testes $\mathbf{X}$ e $\mathbf{Y}$, respectivamente. Em cada linha, conforme podemos ver na Figura A.6, devemos colocar os valores das habilidades para cada indivíduo. Tais valores podem ser incluídos pelo usuário ou obtidos através da implementação que realiza a estimação das habilidades pela TRI.

```
 0.732962070
-1.003800439
-0.717748062
 0.313986020
-0.941743301
 0.872487050
 0.583760022
-0.624258819
```

Figura A.6: Exemplo de um arquivo *habilidadesx.txt.*

Uma observação importante é que os valores das habilidades podem ser colocados com números decimais “normais”, conforme mostra a Figura A.6 ou colocados em forma de notação científica (7.32962070e-01). Os programas que fazem as leituras dos arquivos de habilidades conseguem reconhecer as duas formas de se entrar com os números.

A.2.4. Arquivos *itenscomunsx.txt* e *itenscomunsy.txt*

Estes arquivos são necessários apenas quando se deseja realizar uma equalização de um dos testes para uma métrica que já está pré-definida. Neste caso, faz-se necessário saber quais itens devem ser tratados como comuns. Assim, estes arquivos contém apenas uma linha com os nomes dos itens comuns separados por vírgulas, conforme ilustra a Figura A.7.

```
29,23,24,11,13,14,25,26,27,17,18,19,20,21,1,3,6,7,28,15,16
```

Figura A.7: Exemplo de um arquivo *itencomunsx.txt* .

A.3. Programa – Análise Clássica de Itens

Nome do programa em *EstatR.exe*:	Fase 1 – Análise de itens – Teste X
Nome do arquivo:	fase1-x.txt
Versão:	Teste X
Arquivos de entrada necessários:	entradax.txt
Arquivos de saída gerados:	saida-fase1-x.txt saida-fase1-hist-x.jpg

Nome do programa em *EstatR.exe*:	Fase 1 – Análise de itens – Teste Y
Nome do arquivo:	fase1-y.txt
Versão:	Teste Y
Arquivos de entrada necessários:	entraday.txt
Arquivos de saída gerados:	saida-fase1-y.txt saida-fase1-hist-y.jpg

Estas implementações realizam uma análise clássica dos itens de um teste X ou de um teste Y.Os programas *fase1-x.txt* e *fase1-y.txt* devem ser configurados antes de se realizar as análises. Um exemplo de configuração é mostrado na Figura A.8:

```
# Configure o local onde estão os arquivos e onde será gravado o
arquivo de saída.
setwd("C:/TRI/exemplo")

# respostas são literais (A,B,C...) ou são numéricas (corrigidas:
1=acerto, 0=erro)?
# Escolha "L" para literais ou "N" para numericas
tiporesposta<-"L"
```

Figura A.8: Exemplo de configuração do arquivo *fase1-x.txt.*

Devemos configurar apenas o que está destacado em negrito na Figura A.8. Ou seja, devemos informar em que diretório se encontram os arquivos de entrada e onde serão salvos os arquivos gerados na saída. Em nosso exemplo, tais arquivos encontram-se no diretório **C:/TRI/exemplo**. Cuidado na digitação do endereço: o programa faz distinção entre maiúsculas e minúsculas. Mais abaixo, devemos configurar se desejamos trabalhar com dados literais (**L**), conforme exemplo da Figura A.2 ou numéricos (**N**), conforme exemplo da Figura A.3.Após executar o programa, são gerados dois arquivos de saída: um arquivo de texto chamado *saída-fase1-x.txt* (ou *saída-fase1-y.txt*) e um arquivo de imagem chamado *saida-fase1-hist-x.jpg* (ou *saida-fase1-hist-y.jpg*).

No arquivo *saída-fase1-x.txt* podemos observar no seu início uma identificação do teste (X ou Y) e, logo a seguir, temos todos os itens presentes no teste e que foram considerados na análise (“S” na terceira linha de *entradax.txt*). Para cada item, podemos observar os seguintes elementos, que já foram discutidos no Capítulo 3:

- ITEM: nome do item analisado
- Prop. corr.: é a proporção de acertos do item, também conhecido como índice de dificuldade do item.
- Ind. discrim.: é o índice de discriminação do item.
- Pto. biss.: é o coeficiente de correlação ponto bisserial.

Dentro de cada item, temos as seguintes colunas:

- Alter.: alternativas de cada item acrescido da opção “Outros” que contém os itens deixados em branco ou com resposta diferente da informada. Por exemplo, se foi informado que um item contém 5 alternativas mas nas respostas de um dos indivíduos apareceu a letra “F”, ou estava em branco, ou apareceu um outro caracter (como *), essa resposta é considerada na opção “Outros”. No caso de respostas numéricas (do tipo 0 ou 1), na saída, a alternativa “A” corresponde ao “1”, ou seja, representa a alternativa correta, enquanto que a alternativa B representa o “0”, ou seja, a alternativa errada. Visto que as respostas já estão corrigidas, não espera-se encontrar dados omissos. Por isso, espera-se que, em todos itens analisados, a opção “Outros” apareça com valores iguais a zero em todos os campos.
- Prop.: proporção de acerto da referida alternativa. A soma dos valores desta coluna, dentro do processamento do programa, é igual a 1. Porém, por motivos de arredondamento dos valores, em alguns itens o somatório pode ser diferente de 1.
- GI: proporção de acerto de cada uma das alternativas no grupo inferior, ou seja, no grupo dos 27% de indivíduos que obtiveram os escores mais baixos.

- GS: proporção de acerto de cada uma das alternativas no grupo superior, ou seja, no grupo dos 27% de indivíduos que obtiveram os escores mais altos.
- Pto.biss.: correlação ponto bisserial para cada uma das alternativas.
- Gabarito: o asterisco indica qual alternativa foi indicada como sendo o gabarito do item.

Ao final desse arquivo são apresentadas algumas estatísticas gerais do teste:

- N. itens: número total de itens considerados na análise.
- N. descartados: número de itens descartados da análise.
- N. indivíduos: número de indivíduos considerados na análise.
- escore médio: escore médio dos indivíduos para os itens considerados.
- desvio padrão: desvio padrão amostral dos escores.
- maximo: escore máximo obtido.
- minimo: escore mínimo obtido.
- mediana: mediana dos escores.
- ALFA: coeficiente alfa de fidedignidade do teste
- EPM: erro padrão de medida do teste.
- num. Indiv. GI: número de indivíduos pertencentes ao grupo dos respondentes com os menores escores.
- escore max. GI: maior escore obtido no grupo GI.
- num. Indiv. GS: número de indivíduos pertencentes ao grupo dos respondentes com os maiores escores.
- escore min. GS: menor escore obtido no grupo GS.

Os arquivos de imagem (*saida-fase1-hist-x.jpg* e *saida-fase1-hist-y.jpg*) produzem um histograma dos escores obtidos pelos respondentes. O número de classes desse histograma é automaticamente definido pelo **R**.

A.4. Programa de equalização na TC – método de Tucker

Nome do programa em *EstatR.exe*:	Fase 2 – Equalização Clássica - Tucker
Nome do arquivo:	fase2-Tucker.txt
Versão:	única
Arquivos de entrada necessários:	entradax.txt entraday.txt
Arquivos de saída gerados:	saida-fase2-Tucker.txt saida-fase2-graf-Tucker.jpg

Para rodar este programa, é necessário que o usuário configure, no início do programa (arquivo *fase2-Tucker.txt*), o local (diretório) onde estão os arquivos de entrada e onde serão criados os arquivos de saída que, no caso do exemplo, é **C:/TRI/exemplo**, e configure os valores dos pesos da população sintética (w_1 e w_2). Na realidade, basta alterar o valor de **w1**, pois **w2** é calculado como sendo $1-w1$. O usuário também deve definir se deseja trabalhar com dados literais (**L**), conforme exemplo da Figura A.2 ou numéricos (**N**), conforme exemplo da Figura A.3. As alterações possíveis estão destacadas em negrito na Figura A.9.

```
# Configure o local onde estão os arquivos e onde será gravado o
arquivo de saída.
setwd("C:/TRI/exemplo")

# atribui os pesos para as populações sintéticas
# w1 = peso para a população X
# w2 = peso para a população Y

w1<-1
w2<-1-w1

# respostas são literais (A,B,C...) ou são numéricas (corrigidas:
1=acerto, 0=erro)?
# Escolha "L" para literais ou "N" para numericas
tiporesposta<-"L"
```

Figura A.9: Exemplo de configuração do arquivo *fase2-Tucker.txt*.

Este programa tem algumas sub-rotinas de erros implementadas, como por exemplo a que detecta a existência de algum item que ultrapasse 20 alternativas e outra que informa ao usuário caso não sejam encontrados itens comuns nos testes **X** e **Y**. Como já foi explicado anteriormente, o programa consegue detectar os itens comuns a partir de seus nomes (quarta linha dos arquivos *entradax.txt* e *entraday.txt*). Portanto, itens comuns devem necessariamente ter o mesmo nome, para que possam ser identificados pelo programa e, assim, poder realizar a equalização.

Na saída, são gerados dois arquivos: um de texto chamado *saída-fase2-Tucker.txt* e outro de imagem chamado *saída-fase2-graf-Tucker.jpg*. O primeiro arquivo contém os resultados e algumas estatísticas, enquanto que, no segundo, encontramos gráficos de regressão através do método dos mínimos quadrados.

No arquivo de saída *saída-fase2-Tucker.txt*, temos as informações do método utilizado (Tucker), e de que a amostra da população 1 realizou a prova tipo **X** e, da população 2, tipo **Y**. Em seguida, os pesos w_1 e w_2 correspondentes à população sintética e que foram definidos nas configurações iniciais do programa pelo usuário. Após isso, são informados o número de itens não comuns da prova **X**, o número de itens não comuns presentes na prova **Y** e, finalmente, o número de itens comuns aos dois testes, chamado de teste **V**.

A seguir, são mostrados os números de respondentes de cada um dos dois testes. Depois, aparecem algumas estatísticas como média e desvio-padrão correspondentes aos:

– G1 – teste **X**: indivíduos provenientes da população 1 e que realizaram os itens não comuns da prova **X**.

– G1 – teste **V**: indivíduos provenientes da população 1 e que realizaram os itens comuns da prova **X**.

– G2 – teste **Y**: indivíduos provenientes da população 2 e que realizaram os itens não comuns da prova **Y**.

– G2 – teste **V**: indivíduos provenientes da população 2 e que realizaram os itens comuns da prova **Y**.

Mais abaixo, no mesmo arquivo, podemos ver algumas estatísticas do teste, que são obtidos a partir das fórmulas apresentadas no Quadro-resumo 1 do Capitulo 4:

- gama1: valor de γ_1.
- gama2: valor de γ_2.
- media pop. sint. X: média dos escores da população sintética para o teste **X**.
- media pop. sint. Y: média dos escores da população sintética para o teste **Y**.
- desvio-padrao X: desvio padrão dos escores da população sintética para o teste **X**.
- desvio-padrao Y: desvio padrão dos escores da população sintética para o teste **Y**.

Em seguida, são apresentados os resultados da equalização que transforma os escores do teste **X** na escala de escores do teste **Y** dada por $l_Y(x) = Ax + B$:

- A (inclinação): corresponde ao coeficiente angular da reta que ajusta a equalização;
- B (intercepto): corresponde ao coeficiente linear da reta que ajusta a equalização.

Em seguida, é apresentada uma tabela onde na coluna da esquerda aparecem os valores dos possíveis escores do teste **X**, admitindo que o teste âncora seja interno, e, na coluna da direita, aparecem os correspondentes escores ajustados para a escala do teste **Y**. Em alguns casos, ocorrem problemas nos extremos dos escores equalizados, ou seja, podem ser calculados escores que não existam na escala do teste **Y**, como, por exemplo, escores negativos ou escores superiores ao total de itens do teste **Y**. Em razão disso, utilizou-se uma sub-rotina que

transforma escores equalizados negativos em 0 e escores superiores ao máximo possível em valores iguais ao total de itens do teste **Y**. É gerado também um arquivo chamado *saída-fase2-graf-Tucker.jpg* contendo os gráficos de regressão dos escores do teste **X** (não comuns) sobre **V** e do teste **Y** (não comuns) sobre **V**. Abaixo dos mesmos, apresenta-se um gráfico de resíduos para ajudar na verificação da suposição de regressão linear do método de Tucker.

A.5. Programa de equalização na TC – método de Levine

Nome do programa em *EstatR.exe*:	Fase 2 – Equalização Clássica - Levine
Nome do arquivo:	fase2-Levine.txt
Versão:	única
Arquivos de entrada necessários:	entradax.txt entraday.txt
Arquivos de saída gerados:	saida-fase2-Levine.txt saida-fase2-graf-Levine.jpg

Para rodar este programa, é necessário que o usuário configure, no início do programa (arquivo *fase2-Levine.txt*), o local (diretório) onde estão os arquivos de entrada e onde serão criados os arquivos de saída que, no caso do exemplo, é C:/TRI/exemplo, e configure os valores dos pesos da população sintética (w_1 e w_2). Na realidade, basta alterar o valor de **w1**, pois **w2** é calculado como sendo 1-w1. O usuário também deve definir se deseja trabalhar com dados literais (**L**), conforme exemplo da Figura A.2 ou numéricos (**N**), conforme exemplo da Figura A.3. Além disso, é necessário definir se o teste âncora utilizado é interno (**I**) ou externo (**E**). As alterações possíveis estão destacadas em negrito na Figura A.10.

```
# Configure o local onde estão os arquivos e onde será gravado o
arquivo de saída.
setwd("C:/TRI/exemplo")

# atribui os pesos para as populações sintéticas
# w1 = peso para a população X
# w2 = peso para a população Y

w1<-1
w2<-1-w1

# Define se o teste âncora é interno (I) ou externo (E)
testeancora<-"I"

# respostas são literais (A,B,C...) ou são numéricas (corrigidas:
1=acerto, 0=erro)?
# Escolha "L" para literais ou "N" para numericas
tiporesposta<-"L"
```

Figura A.10: Exemplo de configuração do arquivo *fase2-Tucker.txt*.

Algumas sub-rotinas de erro foram implementadas da mesma forma que as descritas em A.2. Na saída, são gerados dois arquivos: um de texto chamado *saída-fase2-Levine.txt* e outro de imagem chamado *saída-fase2-graf-Levine.jpg*. O primeiro arquivo contém os resultados e algumas estatísticas, enquanto que, no segundo, encontramos gráficos de regressão através do método dos mínimos quadrados. No arquivo *saída-fase2-Levine.txt* encontramos os seguintes elementos:

- w1 e w2: pesos das populações sintéticas definidos pelo usuário.
- Teste âncora: pode ser interno ou externo e é definido pelo usuário.
- Itens consider. X: número de itens considerados do teste $\mathbf{X}$.
- Itens consider. Y: número de itens considerados do teste $\mathbf{Y}$.
- Num. Itens comuns: número de itens comuns aos dois testes.

Em seguida, são mostrados os números de indivíduos que responderam os testes $\mathbf{X}$ e $\mathbf{Y}$. Mais abaixo, no mesmo arquivo, podemos ver algumas estatísticas do teste, que são obtidos a partir das fórmulas apresentadas no Quadro-resumo 2 do Capitulo 4:

- gama1: valor de γ_1.

- gama2: valor de γ_2.

- media pop. sint. X: média dos escores da população sintética para o teste **X**.
- media pop. sint. Y: média dos escores da população sintética para o teste **Y**.
- desvio-padrao X: desvio padrão dos escores da população sintética para o teste **X**.
- desvio-padrao Y: desvio padrão dos escores da população sintética para o teste **Y**.

Em seguida, são apresentados os resultados da equalização que transforma os escores do teste **X** na escala de escores do teste **Y** dada por $l_Y(x) = Ax + B$:

- A (inclinação): corresponde ao coeficiente angular da reta que ajusta a equalização;
- B (intercepto): corresponde ao coeficiente linear da reta que ajusta a equalização.

Em seguida, é apresentada uma tabela onde na coluna da esquerda aparecem os valores dos possíveis escores do teste **X** e, na coluna da direita, aparecem os correspondentes escores ajustados para a escala do teste **Y**. Em alguns casos ocorrem problemas nos extremos dos escores equalizados e a solução dada foi a mesma descrita em A.4. É gerado também um arquivo chamado *saída-fase2-graf-Levine.jpg* contendo os mesmos gráficos de *saída-fase2-graf-Tucker.jpg* descritos em A.4.

A.6. Programa de equalização na TC – método eqüipercentil

Nome do programa em *EstatR.exe*:	Fase 2 – Equalização clássica - Equipercentil
Nome do arquivo:	fase2-Equipercentil.txt
Versão:	única
Arquivos de entrada necessários:	entradax.txt entraday.txt
Arquivos de saída gerados:	saida-fase2-Equipercentil.txt

Para rodar este programa, é necessário que o usuário configure, no início do programa (arquivo *fase2-Equipercentil.txt*), o local (diretório) onde estão os arquivos de entrada e onde serão criados os arquivos de saída que, no caso do exemplo, é **C:/TRI/exemplo**. O usuário também deve definir se deseja trabalhar com dados literais (**L**), conforme exemplo da Figura A.2 ou numéricos (**N**), conforme exemplo da Figura A.3. Além disso, é necessário definir se o teste âncora utilizado é interno (**I**) ou externo (**E**). As alterações possíveis estão destacadas em negrito na Figura A.11.

```
# Configure o local onde estão os arquivos e onde será gravado o
arquivo de saída.
setwd("C:/TRI/exemplo")

# Define se o teste âncora é interno (I) ou externo (E)
testeancora<-"I"

# respostas são literais (A,B,C...) ou são numéricas (corrigidas:
1=acerto, 0=erro)?
# Escolha "L" para literais ou "N" para numericas
tiporesposta<-"L"
```

Figura A.11: Exemplo de configuração do arquivo *fase2-Equipercentil.txt*.

Algumas sub-rotinas de erro foram implementadas da mesma forma que as descritas em A.2. Na saída, é gerado um arquivo de texto chamado *saída-fase2-Equipercentil.txt*, que contém os resultados da equalização. No início do arquivo

encontramos o nome do método utilizado (equipercentil) e, a seguir, duas tabelas contendo as informações sobre os testes $\mathbf{X}$ e $\mathbf{Y}$:

- num. indivíduos: número de alunos que realizaram o teste mencionado.
- num. itens comuns: número de itens comuns a ambos testes.
- num. itens não comuns: número de itens não comuns no teste mencionado.

Nas colunas das tabelas encontramos os seguintes elementos:

- escore (x) ou escore (y): valores dos possíveis escores em cada uma das provas consideradas;
- f(x) ou g(y): proporção de indivíduos que obtiveram o escore da primeira coluna.
- F(x) ou G(y): distribuição acumulada dos acertos de cada escore.
- P(x) (%) ou Q(y) (%): *rank* percentil de cada escore.

Mais adiante, encontramos uma tabela que nos fornece os valores da equalização através do método eqüipercentil. Antes dela, há indicação se o teste âncora é interno ou externo e, a seguir, são apresentados os possíveis escores do teste $\mathbf{X}$ e, ao lado, os valores obtidos pela equalização. Em alguns casos ocorrem problemas nos extremos dos escores equalizados e a solução dada foi a mesma descrita em A.4.

No final do arquivo, temos uma tabela que apresenta os 4 primeiros momentos (média, desvio-padrão, assimetria e curtose) para o teste $\mathbf{Y}$, teste $\mathbf{X}$ e para os escores do teste equalizado $e_Y(x)$.

A.7. Programa de equalização na TC – método de Braun-Holland

Nome do programa em *EstatR.exe*:	Fase 2 – Equalização Clássica – Braun-Holland
Nome do arquivo:	fase2-Braun-Holland.txt
Versão:	única
Arquivos de entrada necessários:	entradax.txt entraday.txt
Arquivos de saída gerados:	saida-fase2-BraunHol.txt saída-fase2-graf-BraunHol.jpg

Para rodar este programa, é necessário que o usuário configure, no início do programa (arquivo *fase2-Braun-Holland.txt*), o local (diretório) onde estão os arquivos de entrada e onde serão criados os arquivos de saída que, no caso do exemplo, é **C:/TRI/exemplo**. É necessário definir se o teste âncora utilizado é interno (**I**) ou externo (**E**). Deve ser configurado o valor do peso w_1 da população sintética. O valor de w_2 é automaticamente calculado fazendo-se 1-w_1. O usuário também deve definir se deseja trabalhar com dados literais (**L**), conforme exemplo da Figura A.2 ou numéricos (**N**), conforme exemplo da Figura A.3. As alterações possíveis estão destacadas em negrito na Figura A.11.

```
# Configure o local onde estão os arquivos e onde será gravado o
arquivo de saída.
setwd("C:/TRI/exemplo")

# Define se o teste âncora é interno (I) ou externo (E)
testeancora<-"I"

# atribui os pesos para as populações sintéticas
# w1 = peso para a população X
# w2 = peso para a população Y
w1<-1
w2<-1-w1

# respostas são literais (A,B,C...) ou são numéricas (corrigidas:
1=acerto, 0=erro)?
# Escolha "L" para literais ou "N" para numericas
tiporesposta<-"L"
```

Figura A.11: Exemplo de configuração do arquivo *fase2-Braun-Holland.txt.*

Algumas sub-rotinas de erro foram implementadas da mesma forma que as descritas em A.2. Na saída, é gerado um arquivo de texto chamado *fase2-Braun-Holland.txt*, que contém os resultados da equalização e um arquivo gráfico denominado *saída-fase2-graf-BraunHol.jpg*. No início do arquivo *fase2-Braun-Holland.txt* encontramos o nome do método utilizado (Braun-Holland) e, a seguir, algumas informações sobre os testes:

- Teste âncora: informa qual tipo de teste utilizado na análise, que pode ser interno ou externo e é definido pelo usuário.
- w1 e w2: pesos das populações sintéticas definidos pelo usuário.
- Itens consider. X: número de itens considerados do teste **X**.
- Itens descart. X: número de itens descartados do teste **X**.
- Itens consider. Y: número de itens considerados do teste **Y**.
- Itens descart. Y: número de itens descartados do teste **Y**.
- Num. Itens comuns: número de itens comuns aos dois testes.

Em seguida, aparecem algumas estatísticas:

- media X: média dos escores do teste **X**.
- media Y: média dos escores do teste **Y**.
- desvio-padrao X: desvio-padrão dos escores do teste **X**.
- desvio-padrao Y: desvio-padrão dos escores do teste **Y**.

Em seguida, são apresentados os resultados da equalização que transforma os escores do teste **X** na escala de escores do teste **Y** dada por $l_Y(x) = Ax + B$:

- A (inclinação): corresponde ao coeficiente angular da reta que ajusta a equalização;
- B (intercepto): corresponde ao coeficiente linear da reta que ajusta a equalização.

E, finalmente, é mostrada uma tabela contendo os possíveis escores do teste X e os correspondentes escores equalizados para a escala do teste Y utilizando a

equação de equalização apresentada. Em alguns casos ocorrem problemas nos extremos dos escores equalizados e a solução dada foi a mesma descrita em A.4. É gerado também um arquivo chamado *saída-fase2-graf-BraunHol.jpg* contendo os mesmos gráficos de *saída-fase2-graf-Tucker.jpg* descritos em A.4.

A.8. Programa de estimação na TRI – estimação dos parâmetros dos itens

Nome do programa em *EstatR.exe*:	Fase 3 – Estimação dos parâmetros – Teste X
Nome do arquivo:	fase3-estima parametros-x.txt
Versão:	Teste X
Arquivos de entrada necessários:	entradax.txt
Arquivos de saída gerados:	saida-fase3-TRI-x.txt parametrosx.txt

Nome do programa em *EstatR.exe*:	Fase 3 – Estimação dos parâmetros – Teste Y
Nome do arquivo:	fase3-estima parametros-y.txt
Versão:	Teste Y
Arquivos de entrada necessários:	entraday.txt
Arquivos de saída gerados:	saida-fase3-TRI-y.txt parametrosy.txt

Esta implementação realiza a calibração dos parâmetros dos itens de um teste quando as habilidades dos respondentes são desconhecidas. Utiliza, para isso, o método de máxima verossimilhança marginal. Para rodar este programa, é necessário que o usuário realize diversas configurações, as quais estão divididas em duas partes: configurações básicas e configurações avançadas, que muitas vezes não precisam ser alteradas, podendo ser consideradas *default* do programa. Nas figuras A.12 e A.13, vemos quais são as configurações básicas que devem ser alteradas pelo usuário (em negrito). Lembramos que as configurações para os

arquivos *fase3-estima parametros-x.txt* e *fase3-estima parametros-y.txt* são idênticas.

```
# Configure o local onde estão os arquivos e onde será gravado o
arquivo de saída.
setwd("C:/TRI/exemplo")

# respostas são literais (A,B,C...) ou estão numéricas (corrigidas:
1=acerto, 0=erro)?
# Escolha "L" para literais ou "N" para numericas
tiporesposta<-"L"
```

Figura A.12: configurações básicas do arquivo *fase3-estima parametros-x.txt.*

Na Figura A.12 vemos que o usuário, assim como foi feito em alguns programas anteriores, deve configurar o local onde estão os arquivos de entrada e onde serão salvos os arquivos de saída. Deve, também, informar se os dados do teste contidos nos arquivos de entrada estão no formato de dados literais (**L**), conforme exemplo da Figura A.2 ou numéricos (**N**), conforme exemplo da Figura A.3.

Na Figura A.13 temos a continuação das configurações básicas. O usuário deve informar como deseja que sejam os “chutes iniciais” ou estimativas iniciais para os parâmetros dos itens. São dadas 5 opções ao usuário que deve colocar o número correspondente à opção no campo “chuteinicial”. A definição dessas estimativas iniciais é um passo muito importante na estimação pela TRI e deve ser escolhido com cautela. Percebemos que, no método de máxima verossimilhança marginal, os valores dos chutes iniciais podem gerar alterações bastante significativas nas estimativas. As opção disponíveis para escolha das estimativas iniciais dos parâmetros de discriminação (**a**), de dificuldade (**b**) e acerto casual (**c**) são:

```
### Define os chutes iniciais ('chuteinicial') de acordo com os
números a seguir (de 1 a 4):
# (Em todas as possibilidades, c é obtido a partir do número de
alternativas do item).
#
# 1 - chute padrão recomendado na literatura:
#     a = obtido a partir da correlação ponto bisserial
#     b = obtido a partir da curva normal + ponto bisserial
#
# 2 - a = todos os valores de a recebem uma constante (defina o valor
de 'chute.a') (sugestão: 1)
#     b = obtido a partir da curva normal + ponto bisserial
#
# 3 - a = todos os valores de a recebem uma constante (defina o valor
de 'chute.a') (sugestão: 1)
#     b = todos os valores de b recebem uma constante (defina o valor
de 'chute.b') (sugestão: 0)
#
# 4 - a = valores gerados de uma log-normal com média 'lognor.media'
e desvio-padrão
#         'lognor.dp' (sugestão: média 0 e variância 0.3)
#     b = obtido a partir da curva normal (0,1)
#
# 5 - a = valores gerados de uma log-normal com média 'lognor.media'
e desvio-padrão
#         'lognor.dp' (sugestão: média 0 e variância 0.3)
#     b = gerados a partir da curva normal N(0,1)
#     c = obtido a partir de uma beta (defina parametros c1 e c2)
(sugestão: c1=6 e c2=16)

chuteinicial<-2

chute.a<-1
chute.b<-0

lognor.media<-0
lognor.dp<-0.3

c1<-6
c2<-16
```

Figura A.13: Continuação das configurações básicas do arquivo *fase3-estima parametros-x.txt.*

1 – corresponde às estimativas iniciais recomendadas em Andrade et.al (2000). A escolha dos parâmetros é feita da seguinte forma:

- **a** é obtido a partir da correlação ponto bisserial;
- **b** é obtido a partir da normal associada à correlação ponto bisserial.
- **c** é obtido a partir do número de alternativas ($1/m_i$ onde m_i é o número de alternativas do item i).

2 – possibilita que o usuário atribua uma constante única para as estimativas iniciais de **a**, enquanto que **b** fica estimado da mesma maneira que na opção **1**. Ou seja:

- todos valores de **a** recebem uma constante definida pelo usuário no campo **chute.a**;
- **b** é obtido a partir da normal associada à correlação ponto bisserial.
- **c** é obtido a partir do número de alternativas ($1/m_i$ onde m_i é o número de alternativas do item i).

3 – nesta opção, pode-se atribuir constantes tanto para o parâmetro **a** quanto para o parâmetro **b**:

- todos valores de **a** recebem uma constante definida pelo usuário no campo **chute.a**;
- todos valores de **b** recebem uma constante definida pelo usuário no campo **chute.b**.
- **c** é obtido a partir do número de alternativas ($1/m_i$ onde m_i é o número de alternativas do item i).

4 – nesta opção, são gerados valores aleatórios para os parâmetros **a** e **b**:

- os parâmetros **a** são gerados aleatoriamente a partir de uma distribuição log-normal com média definida pelo usuário em **lognor.media** e desvio-padrão definido em **lognor.dp**.
- **b** é gerado a partir da curva normal (0,1).
- **c** é obtido a partir do número de alternativas ($1/m_i$ onde m_i é o número de alternativas do item i).

5 – nesta opção, são gerados valores aleatórios para os parâmetros **a**, **b** e **c**:

- os parâmetros **a** são gerados aleatoriamente a partir de uma distribuição log-normal com média definida pelo usuário em **lognor.media** e desvio-padrão definido em **lognor.dp**.

- **b** é gerado a partir da curva normal (0,1).
- **c** é gerado a partir de uma beta com parâmetros **c1** e **c2** definidos pelo usuário.

Depois disso, pode-se fazer algumas configurações que chamamos de avançadas. Como já dissemos, tais configurações não precisam ser necessariamente alteradas, podendo ser consideradas como padrão do programa. Tais configurações podem ser vistas na Figura A.14.

```
###
### Configurações avançadas
###

### Geração dos pontos de quadratura:

# número de pontos de quadratura
npq<-10

# amplitude do intervalo para pontos de quadratura
limite<-4

# média da curva normal para pontos de quadratura
m.norm<-0

# variância da curva normal para pontos de quadratura
var.norm<-1

### atualização condicional do algoritmo EM:

# limite inferior para parâmetro a
lia<-0

# limite superior para parâmetros a
lsa<-2.2

# limite para parâmetro d
ld<-6

### critérios de parada:

# 1 - número de ciclos do algoritmo EM
n.ciclo.EM<-50

# 2 - precisão das estimatimas dos itens
prec.item.EM<-0.001
```

Figura A.14: Configurações avançadas do arquivo *fase3-estima parametros-x.txt.*

Nas configurações avançadas, podemos fazer algumas definições com relação à geração dos pontos de quadratura:

- npq: número de pontos de quadratura a serem gerados;
- limite: amplitude do intervalo onde serão gerados os pontos de quadratura. Assim, por exemplo, um valor igual a 4 indica que os pontos de quadratura serão gerados no intervalo [-4,4];
- m.norm: média da curva normal utilizada para a geração dos pontos de quadratura;
- var.norm: variância da curva normal utilizada para a geração dos pontos de quadratura.

A seguir, aparecem algumas definições a respeito da atualização condicional do algoritmo EM. Por questões de convergência dos valores, em algumas simulações foi observado que poderiam ocorrer problemas com as estimativas dos parâmetros **a**. Por isso, pode-se restringir o intervalo de seus possíveis valores. Caso **a** ultrapasse os valores definidos pelo usuário, o que o programa faz é não atualizar, no algoritmo EM, o valor desse parâmetro, ficando o valor anterior. Da mesma forma, foi colocada, dentro do programa, uma atualização condicional que não permite que o valor de **c** ultrapasse o intervalo [0,1]. Foi utilizada uma reparametrização, por questões computacionais, que facilita o cálculo de algumas derivadas (do vetor escore e da matriz hessiana) tal que $d_i=-a_i.b_i$ e esse parâmetro d também possui uma atualização condicional. Se, por exemplo, esperamos que o parâmetro **a** varie em [0,2] e **b** varie em [-3,3], esperamos que o parâmetro **d** varie de -6 até 6. Logo, se for definido 6 para os limites desse parâmetro, valores estimados que estejam fora do intervalo [–6,6] serão rejeitados e o programa manterá a última estimativa compreendida no intervalo definido. Maiores informações sobre essa reparametrização podem ser encontradas em Baker (1992). Assim:

- lia: limite inferior para o parâmetro de discriminação (**a**);

- lsa: limite superior para o parâmetro de discriminação (**a**);
- ld: limite para o parâmetro **d**.

Depois, existem duas opções que compõem os critérios de parada do algoritmo EM:

- n.ciclo.EM: número máximo de ciclos EM executados antes de interromper o processo de estimação;
- prec.item.EM: precisão do item dentro do algoritmo EM. Essa precisão é calculada através da diferença entre os valores das estimativas no passo **t** e os valores no passo **t+1**. Caso a diferença seja menor que prec.item.EM, o algoritmo é interrompido.

Quando algum dos dois critérios for alcançado, o algoritmo EM é interrompido. Dentro do algoritmo EM, utilizou-se o algoritmo Newton-Raphson (NR) para se fazer a maximização da função de máxima verossimilhança. Neste programa, utilizou-se apenas 1 laço de NR, ou seja, a cada ciclo EM, o algoritmo NR é rodado apenas uma única vez. Optou-se por este método devido ao fato dele apresentar os melhores resultados (em relação à utilização de mais de um laço de NR) em simulações. Algumas sub-rotinas de erro foram implementadas da mesma forma que as descritas em A.2. Na saída, são gerados dois arquivos de texto chamados *saída-fase3-TRI-x.txt* e *parametrosx.txt*. O primeiro arquivo mostra na saída o número de ciclos do algoritmo EM rodados até se obter a convergência ou atingir o critério de parada, mostra o número de itens considerados e desconsiderados na análise e o total de respondentes. Podemos ver, também, qual método para se obter as estimativas iniciais foi utilizado, dentre os 4 possíveis apresentados. É apresentada uma tabela contendo quatro colunas: nome do item, estimativas dos parâmetros de discriminação (**a**), estimativas dos parâmetros de dificuldade (**b**) e estimativa dos parâmetros de acerto casual (**c**). Em cada linha, para cada um dos itens, são mostrados as estimativas dos itens e, na linha de baixo, os erros-padrão correspondentes. Além disso, é gerado um

segundo arquivo chamado *parametrosx.txt* (ou *parametrosy.txt*) que possuem os valores dos parâmetros **a**, **b** e **c** dos itens considerados na análise. Esse arquivo é criado com o único propósito de ser utilizado por outros programas como o de calibração de habilidades e equalizações pela TRI.

A.9. Programa de estimação na TRI – estimação das habilidades

Nome do programa em *EstatR.exe*:	Fase 3 – Estimação das habilidades – Teste X
Nome do arquivo:	fase3-estima habilidades-x.txt
Versão:	Teste X
Arquivos de entrada necessários:	entradax.txt parametrosx.txt
Arquivos de saída gerados:	saida-fase3-habil-x.txt habilidadesx.txt

Nome do programa em *EstatR.exe*:	Fase 3 – Estimação das habilidades – Teste Y
Nome do arquivo:	fase3-estima habilidades -y.txt
Versão:	Teste Y
Arquivos de entrada necessários:	entraday.txt parametrosx.txt
Arquivos de saída gerados:	saida-fase3-habil-y.txt habilidadesy.txt

Esta implementação realiza a estimação das habilidades dos respondentes quando os parâmetros dos itens são conhecidos. Utiliza, para isso, o método de máxima verossimilhança. Para rodar este programa, é necessário que o usuário realize algumas configurações nos arquivos *fase3-estima habilidades-x.txt* e/ou *fase3-estima habilidades -y.txt*, conforme podemos ver na Figura A.15.

```
# Configure o local onde estão os arquivos e onde será gravado o
arquivo de saída.
setwd("C:/TRI/exemplo")

# número máximo de iterações do algoritmo Newton Raphson
tmax<-50

# critério de parada para o algoritmo Newton Raphson (valor em
módulo)
crit<-0.01
```

Figura A.15: Configurações básicas do arquivo *fase3-estima habilidades-x.txt.*

O usuário deve definir apenas o local onde estão os arquivos de entrada e onde serão salvos os arquivos de saída e deve definir os critérios de parada do algoritmo Newton-Raphson:

- tmax: número máximo de iterações que podem ser realizadas pelo algoritmo Newton-Raphson;
- crit: o algoritmo é interrompido quando o valor, em módulo, da diferença entre a estimativa da habilidade no passo **t** e no passo **t+1** for menor que o valor definido em **crit**.

No arquivo de saída (*saida-fase3-habil-x.txt)*, é apresentado o número de respondentes do teste em questão e uma tabela contendo a identificação do respondente, sua habilidade estimada, o erro padrão associado e o número de ciclos do algoritmo Newton-Raphson até se atingir a estimativa. Observou-se, em algumas situações, problemas na estimação de alguns indivíduos, o que ocasiona, na saída, um valor igual a **NaN** (*Not a Number*). Isso indica que o programa obteve valores muito pequenos com os quais não conseguiu realizar todos os cálculos e, por isso, é informado que o valor obtido não é um número. Também observou-se que esse problema depende, não só das respostas do indivíduo, mas também do conjunto de dados. Ou seja, se num conjunto de dados um indivíduo recebeu **NaN**, em outro conjunto, o mesmo indivíduo pode ter sua habilidade calculada normalmente. Quanto a quantidade de casas decimais, pode ser que sejam exibidas mais do que 3 casas. Isso ocorre geralmente quando o valor é muito

pequeno, ou muito grande, e o **R**, automaticamente transforma esse valor para notação científica. É gerado também um arquivo de texto chamado *habilidadesx.txt* que contém os valores das habilidades estimadas para os indivíduos. Esta saída serve apenas para ser usada em outras implementações, como as de equalização.

A.10. Programa de equalização na TRI – 2 testes

Nome do programa em *EstatR.exe*:	Fase 4 – Equalização entre 2 testes
Nome do arquivo:	fase4-equalizacao 2 teste.txt
Versão:	Única
Arquivos de entrada necessários:	entradax.txt entraday.txt parametrosx.txt parametrosy.txt habilidadesx.txt (opcional) habilidadesy.txt (opcional)
Arquivos de saída gerados:	saida-fase4-param2testes.txt saida-fase4-graf.jpg saida-fase4-habil2testes.txt (opcional)

Este programa realiza a equalização a posteriori entre dois testes: transforma os parâmetros dos itens e as habilidades dos respondentes (opcional) da escala do teste $\mathbf{X}$ para a métrica do teste $\mathbf{Y}$. Admite-se o modelo ML3. Como configurações iniciais, o usuário deve informar o local onde estão os arquivos de entrada e onde estão os arquivos de saída. Deve decidir se deseja ou não que o programa faça a equalização das habilidades colocando **S** para equalizar ou **N** para não equalizar (opção **equahabi**). Deve definir, também, qual método deseja utilizar: **1** para média-desvio ou **2** para média-média (opção **método**). Finalmente, deve escolher o que deseja fazer com os valores dos parâmetros de itens comuns. Ou seja, para um item comum do teste $\mathbf{Y}$ temos os valores dos parâmetros; para o item correspondente no teste $\mathbf{X}$ temos outros valores que, quando equalizados, devem

se aproximar dos valores do teste **Y**. Temos, portanto, dois valores diferentes para os parâmetros de um mesmo item. O programa disponibiliza duas alternativas na opção **metcomum**:

1 – obter a média entre os valores dos parâmetros de ambos testes;

2 – descartar os valores equalizados e manter os valores do teste **Y**.

As configurações mencionadas podem ser vistas na Figura A.16.

```
# Configure o local onde estão os arquivos e onde será gravado o arquivo
de saída.
setwd("C:/TRI/pacote")

# Você deseja equalizar as habilidades (necessário arquivos
habildiadesx.txt
# e habilidadesy.txt) ? Configure "S" para sim ou "N" para não.
equahabi<-"N"

# escolha o método de equalização a posteriori:
# 1 para média-desvio ou 2 para média-média
metodo<-1

# Para os parâmetros dos itens comuns deve-se
# 1 - obter a média dos parâmetros de Y com os equalizados de X
# 2 - manter os parâmetros do teste Y
metcomum<-1
```

Figura A.16: Configurações iniciais do arquivo *fase4-equalizacao 2 teste.txt*.

O programa sempre gera a saída contendo os parâmetros equalizados (arquivo *saida-fase4-param2testes.txt*). Nesse arquivo, podemos encontrar os seguintes elementos:

Media (a): média dos parâmetros de discriminação (**a**) para os testes **X** e **Y**.

Media (b): média dos parâmetros de dificuldade (**b**) para os testes **X** e **Y**.

Desvio-padrão (b): desvio-padrão dos parâmetros de dificuldade (**b**) para os testes **X** e **Y**.

alfa: valor do parâmetro α utilizado na equalização e definido em 5.4.1.

beta: valor do parâmetro β utilizado na equalização e definido em 5.4.1.

Mais adiante, são mostrados o número de itens considerados na análise para o teste **X** e para o teste **Y** e o número de itens comuns a ambos testes. Depois é

apresentada uma tabela contendo, em suas colunas, os nomes dos itens e os valores já equalizados dos parâmetros **a**, **b** e **c**. Também é gerado um arquivo chamado *saida-fase4-graf.jpg* que contém os gráficos de dispersão dos parâmetros a e b entre os testes $\mathbf{X}$ e $\mathbf{Y}$. A idéia é que o usuário possa verificar se existe, a partir dos gráficos, uma relação linear entre os valores equalizados dos parâmetros para os dois testes. Opcionalmente, se definido pelo usuário, o programa pode gerar um arquivo contendo as habilidades dos respondentes equalizadas: primeiro aparecem as habilidades dos indivíduos que realizaram o teste X e, depois, que fizeram o teste Y, na mesma seqüência em que os indivíduos foram colocados no arquivo *entradax.txt* e *entraday.txt*. Foi implementada uma sub-rotina de erro que avisa, através de janelas, caso não seja encontrado nenhum item em comum nos dois testes. Como já foi dito, para que o programa reconheça os itens comuns é preciso que os nomes atribuídos aos itens sejam os mesmos nos arquivos de entrada, ou seja, mesmo item deve ter o mesmo nome.

A.11. Programa de equalização na TRI – escala pré-definida

Nome do programa em *EstatR.exe*:	Fase 4 – Equalização – escala X
Nome do arquivo:	fase4-equalizacao escala-x.txt
Versão:	Teste X
Arquivos de entrada necessários:	entradax.txt parametrosx.txt Itenscomunsx.txt habilidadesx.txt (opcional)
Arquivos de saída gerados:	saida-fase4-param-escala-x.txt saida-fase4-habil-escala-x.txt (opcional)

Nome do programa em *EstatR.exe*:	Fase 4 – Equalização – escala Y
Nome do arquivo:	fase4-equalizacao escala-y.txt
Versão:	Teste Y
Arquivos de entrada necessários:	entraday.txt parametrosy.txt Itenscomunsy.txt habilidadesy.txt (opcional)
Arquivos de saída gerados:	saida-fase4-param-escala-y.txt saida-fase4-habil-escala-y.txt (opcional)

Estes programas permitem a equalização a posteriori transformando os parâmetros dos itens e as habilidades dos respondentes (opcional) da escala de um dos testes para uma escala pré-definida (obtida a partir de uma equalização prévia entre 2 testes). Isso permite comparar um conjunto de itens com outro conjunto de itens que já haviam sido calibrados para uma determinada métrica. Admite-se o modelo ML3. Como configurações iniciais, o usuário deve informar o local onde estão os arquivos de entrada e onde estão os arquivos de saída. Deve decidir se deseja ou não que o programa faça a equalização das habilidades colocando **S** para equalizar ou **N** para não equalizar (opção **equahabi**). Deve definir, também, qual método deseja utilizar: **1** para média-desvio ou **2** para média-média (opção **método**). Deve escolher o que deseja fazer com os valores dos parâmetros de itens comuns, conforme explicado em A.10, na opção **metcomum**:
1 – obter a média entre os valores dos parâmetros de ambos testes;
2 – descartar os valores equalizados e manter os valores do teste **Y**.

Finalmente, devem ser configuradas informações relativas à métrica da escala pré-definida. Caso o método escolhido seja o média-desvio (opção **1** de **metodo**), deve-se configurar o valor da média do parâmetro **b** na opção **media.bpd** e do desvio-padrão desse parâmetro em **dp.bpd**, correspondentes aos itens comuns utilizados na composição da escala pré-definida. A equalização ajustará os parâmetros do teste para essa escala. Neste caso, deixe qualquer valor na opção **media.apd**, visto que ela não será utilizada nos cálculos. Caso o método

escolhido seja o média-média (opção **2** de **metodo**), deve-se definir o valor da média dos itens comuns que compõem a escala pré-definida do parâmetro **a** na opção **media.apd** e da média do parâmetro **b** na opção **media.bpd**. Neste caso, deixe qualquer valor na opção **desvio.bpd**, visto que ela não será utilizada nos cálculos. As configurações mencionadas podem ser vistas na Figura A.17.

```
# Configure o local onde estão os arquivos e onde será gravado o arquivo
de saída.
setwd("C:/TRI/exemplo")

# Você deseja equalizar as habilidades (necessário arquivos
habilidadesx.txt)?
# Configure "S" para sim ou "N" para não.
equahabi<-"N"

# escolha o método de equalização a posteriori:
# 1 para média-desvio ou 2 para média-média
metodo<-1

# Valor da média do parâmetros de discriminação (a) da métrica pré-
definida
# (necessário apenas no método média-média; caso o método escolhido seja
# média-desvio, deixar qualquer valor no campo abaixo).
media.apd<-1

# Valor da média do parâmetros de dificuldade (b) da métrica pré-definida
media.bpd<-0.5

# valor do desvio-padrão da métrica pré-definida
# (necessário apenas no método média-desvio; caso o método escolhido seja
# média-média, deixar qualquer valor no campo abaixo).
dp.bpd<-1
```

Figura A.17: Configurações iniciais do arquivo *fase4-equalizacao 2 teste.txt*.

O programa sempre gera a saída contendo os parâmetros equalizados (arquivo *saida-fase4-param-escala-x.txt* ou *saida-fase4-param-escala-y.txt*). Esses arquivos contém os seguintes elementos: em **Num.itens consider.** temos o número de itens considerados do teste inteiro ($\mathbf{X}$ ou $\mathbf{Y}$) e número de itens comuns (tais itens devem ser informados no arquivo *itenscomunsx.txt* ou *itencomunsy.txt*). Depois, aparecem informações sobre a escala pré-definida: média (**media b**) e desvio-padrão (**desvio b**) do parâmetro de dificuldade quando o método escolhido for média-desvio ou aparecem as médias dos parâmetros de discriminação e de dificuldade (**media a** e **media b**) quando o método escolhido for média-média. Por

fim, há uma tabela contendo, em suas colunas, os nomes dos itens e os valores equalizados dos parâmetros **a**, **b** e **c** para a escala pré-definida.

O arquivo *saída-fase4-habil-escala-x.txt* (ou *saída-fase4-habil-escala-y.txt*) possui os valores das habilidades equalizadas para a métrica pré-definida. A seqüência em que os valores aparecem corresponde a mesma seqüência de identificação dos indivíduos no arquivo de entrada.

A.12. Resumo dos arquivos

A seguir, mostramos a Tabela A.1 que poderá ser útil na hora de localizar os arquivos de entrada necessários para cada simulação e quais são os arquivos de saída. Nela estão disponíveis os nomes do programa dentro do software EstatR.exe, os nomes dos arquivos originais dos programas (arquivo mãe), os arquivos de entrada necessários para que o arquivo mãe rode e, finalmente, os arquivos de saída gerados.

Tabela A.1: Resumo dos arquivos das implementações

Nome em EstatR.exe	Arquivo mãe	Arquivos de entrada	Arquivos de saída
Fase 1 – Análise de itens – Teste X	fase1-x.txt	entradax.txt	saida-fase1-x.txt saida-fase1-hist-x.jpg
Fase 1 – Análise de itens – Teste Y	fase1-y.txt	entraday.txt	saida-fase1-y.txt saida-fase1-hist-y.jpg
Fase 2 – Equalização clássica - Tucker	fase2-Tucker.txt	entradax.txt entraday.txt	saida-fase2-Tucker.txt saida-fase2-graf-Tucker.jpg
Fase 2 – Equalização clássica - Levine	fase2-Levine.txt	entradax.txt entraday.txt	saida-fase2-Levine.txt saida-fase2-graf-Levine.jpg
Fase 2 – Equalização clássica - Equipercentil	fase2-Equipercentil.txt	entradax.txt entraday.txt	saida-fase2-Equipercentil.txt
Fase 2 – Equalização clássica - Braun-Holland	fase2-Braun-Holland.txt	entradax.txt entraday.txt	saida-fase2-BraunHol.txt saida-fase2-graf-BraunHol.jpg
Fase 3 – Estimação dos parâmetros – Teste X	fase3-estima parametros-x.txt	entradax.txt	saida-fase3-TRI-x.txt parametrosx.txt
Fase 3 – Estimação dos parâmetros – Teste Y	fase3-estima parametros-y.txt	entraday.txt	saida-fase3-TRI-y.txt parametrosy.txt
Fase 3 – Estimação das habilidades – Teste X	fase3-estima habilidades-x.txt	entradax.txt parametrosx.txt	saida-fase3-habil-x.txt habilidadesx.txt
Fase 3 – Estimação das habilidades – Teste Y	fase3-estima habilidades-y.txt	entraday.txt parametrosy.txt	saida-fase3-habil-y.txt habilidadesy.txt

Tabela A.1(Cont.): Resumo dos arquivos das implementações

Nome em EstatR.exe	Arquivo mãe	Arquivos de entrada	Arquivos de saída
Fase 4 – Equalização entre 2 testes	fase4-equalização 2 testes.txt	entradax.txt entraday.txt parametrosx.txt parametrosy.txt habilidadesx.txt (opcional) habilidadesy.txt (opcional)	saida-fase4-param2testes.txt saida-fase4-graf.jpg saida-fase4-habil2testes.txt (opcional)
Fase 4 – Equalização – escala X	fase4-equalização escala-x.txt	entradax.txt parametrosx.txt itenscomunsx.txt habilidadesx.txt (opcional)	saida-fase4-param-escala-x.txt saida-fase4-habil-escala-x.txt (opcional)
Fase 4 – Equalização – escala Y	fase4-equalização escala-y.txt	entraday.txt parametrosy.txt itenscomunsy.txt habilidadesy.txt (opcional)	saida-fase4-param-escala-y.txt saida-fase4-habil-escala-y.txt (opcional)