UNIVERSIDADE FEDERAL DO PARANÁ

HENRIQUE KREFER
RHUAN GABRIEL C. DE LIMA

IMPLEMENTAÇÃO DE UM TESTE ADAPTATIVO COMPUTADORIZADO (TAC) SOBRE A PLATAFORMA CONCERTO

CURITIBA
2013

HENRIQUE KREFER
RHUAN GABRIEL C. DE LIMA

IMPLEMENTAÇÃO DE UM TESTE ADAPTATIVO COMPUTADORIZADO (TAC) SOBRE A PLATAFORMA CONCERTO

Trabalho de Conclusão de Curso apresentado à disciplina Laboratório de Estatística do Curso de Graduação em Estatística da Universidade Federal do Paraná, como exigência parcial para obtenção do grau de Bacharel em Estatística.

Orientador: Prof. Adilson dos Anjos

CURITIBA
2013

AGRADECIMENTOS

Ao professor Adilson dos Anjos por instruir e confiar em nosso modo para desenvolver uma idéia.

Ao Hangouts e ao WhatsApp por permitir organizar as atividades independentemente do horário ou do local.

Ao Dropbox por compartilhar e manter em segurança todo o conteúdo.

Aos familiares e amigos por tolerar os horários impróprios os quais foram reservados para a experiência ímpar de realizar o trabalho.

À todos o nosso muito obrigado!

”Exploration is wired into our brains. If we can see the horizon, we want to know what’s beyond.”

– BUZZ ALDRIN

RESUMO

Implementação de um Teste Adaptativo Computadorizado (TAC) sobre a Plataforma Concerto

A sociedade moderna anseia por medir ou quantificar o conhecimento sobre um determinado traço latente, como o simples critério de seleção para ser admitido em uma universidade ou mesmo em uma pesquisa mercadológica para o lançamento de um novo produto ou serviço, o que tem provocado o desenvolvimento de métodos extremamente eficientes e eficazes para realizar a tarefa de administrar um teste e se adaptar ao seu próprio resultado. O objetivo deste trabalho é apresentar um método e um mecanismo para aplicar um Teste Adaptativo Computadorizado (TAC) baseado na Teoria da Resposta ao Item (TRI). Com o uso do *R* e com base no *package catR* foi desenvolvido e construído o *package stepCAT*, o qual apresenta funções para integrar uma tabela (base de dados) e os itens previamente calibrados pela TRI com o processo de administração de um teste adaptativo, interagindo com o examinando através dos mecanismos da *R Console*, a interface gráfica *Tcl/Tk* e o *Concerto Platform* (Plataforma Concerto) na *web*.

Palavras-chave: Teste Adaptativo Computadorizado, TAC, Teoria da Resposta ao Item, TRI, R software, R packages, catR, stepCAT, Concerto Platform, Plataforma Concerto.

ABSTRACT

Implementing a Computerized Adaptive Testing (CAT) on Concert Platform

The modern society yearns for measuring or quantifying knowledge about a particular latent trait, such as simple selection criterion for admission in an university or even a market research for launching a new product or service, which has led to the development of extremely efficient and effective methods to accomplish the task of administering a test and to adapt to own result. The objective of this paper is to present a method and mechanism for applying a Computerized Adaptive Testing (CAT) based on Item Response Theory (IRT). By using the *R* and based on *package catR* was developed and built the *package stepCAT*, which has functions to integrate a table (database) and previously calibrated items by TRI with the process for administering an adaptive test, by interacting with the examinee through the mechanisms of *R Console*, a graphical interface *Tcl/Tk* and *Concerto Platform* on web.

Key-words: Computerized Adaptive Testing, CAT, Item Response Theory, IRT, R software, R packages, catR, stepCAT, Concerto Platform.

LISTA DE FIGURAS

FIGURA 1 – CURVA CARACTERÍSTICA DO ITEM – CCI 17
FIGURA 2 – CURVA DE INFORMAÇÃO DO ITEM – CII 17
FIGURA 3 – FUNÇÃO DE INFORMAÇÃO DO TESTE 20
FIGURA 4 – PROCESSO DO TAC 21
FIGURA 5 – INTERATIVIDADE COM O USUÁRIO PARA O EXEMPLO 1 . . . 39
FIGURA 6 – INTERATIVIDADE COM O USUÁRIO PARA O EXEMPLO 2 . . . 40
FIGURA 7 – INTERATIVIDADE COM O USUÁRIO PARA O EXEMPLO 3 . . . 40
FIGURA 8 – RESULTADO DA FUNÇÃO *stepCAT* ATRIBUÍDO AO OBJETO . . 41
FIGURA 9 – TELA DO *BROWSER* PARA USAR O CONCERTO 42
FIGURA 10 – TELA DO *CONCERTO* EM *CREATE AN ACCOUNT* 43
FIGURA 11 – TELA DO *CONCERTO* EM *REGISTER NEW ACCOUNT* 43
FIGURA 12 – TELA DO *CONCERTO* EM *LOGIN* 44
FIGURA 13 – TELA DO *CONCERTO* EM *CREATE A TABLE* 45
FIGURA 14 – TELA DO *CONCERTO* EM *ADD NEW TABLE* 45
FIGURA 15 – TELA DO *CONCERTO* EM *IMPORT NEW TABLE* 45
FIGURA 16 – TELA DO *CONCERTO* EM *IMPORT NEW TABLE PROPERTIES* 46
FIGURA 17 – TELA DO *CONCERTO* EM *CREATE A TEST* 46
FIGURA 18 – TELA DO *CONCERTO* EM *ADD NEW TEST* 47
FIGURA 19 – TELA DO *CONCERTO* EM *NEW TEST LOGIC* 48
FIGURA 20 – TELA DO *CONCERTO* EM *NEW TEST SCREEN* 48
FIGURA 21 – TELA DO *CONCERTO* EM *NEW TEST SESSION* 49
FIGURA 22 – TELA DO *CONCERTO* EM *NEW TEST SAVE* 49
FIGURA 23 – TELA DO *CONCERTO* EM *CREATE A NEW TEST* 49
FIGURA 24 – TELA DO *CONCERTO* EM *EDIT A TEST* 49
FIGURA 25 – TELA DO *CONCERTO* EM *NEW TEST SAVE* 50
FIGURA 26 – TELA DO *CONCERTO* EM *RUN A TEST* 50
FIGURA 27 – TELA DO *CONCERTO* EM *RUN A TEST SCREEN* 50
FIGURA 28 – TELA DO *CONCERTO* EM *SCREEN RESULT* 51
FIGURA 29 – TELA DO *BROWSER* PARA TESTAR NO CONCERTO 51

LISTA DE QUADROS

QUADRO 1 – INSTALAÇÃO DO PACOTE *stepCAT* 23
QUADRO 2 – LEITURA DO PACOTE *stepCAT* COMO CÓDIGO-FONTE . . . 24
QUADRO 3 – SINTAXE DA FUNÇÃO *stepCAT.by* 24
QUADRO 4 – SINTAXE DA FUNÇÃO *stepCAT.items* 27
QUADRO 5 – SINTAXE DA FUNÇÃO *stepCAT.table* 29
QUADRO 6 – SINTAXE DA FUNÇÃO *stepCAT.bank* 30
QUADRO 7 – SINTAXE DA FUNÇÃO *stepCAT.start* 31
QUADRO 8 – SINTAXE DA FUNÇÃO *stepCAT.test* 33
QUADRO 9 – SINTAXE DA FUNÇÃO *stepCAT.stop* 34
QUADRO 10 – SINTAXE DA FUNÇÃO *stepCAT.choice* 36
QUADRO 11 – SINTAXE DA FUNÇÃO *stepCAT.final* 37
QUADRO 12 – CÓDIGO *R* DO EXEMPLO 1 DA FUNÇÃO *stepCAT.by* 38
QUADRO 13 – CÓDIGO *R* DO EXEMPLO 2 DA FUNÇÃO *stepCAT.by* 38
QUADRO 14 – CÓDIGO *R* DO EXEMPLO 3 DA FUNÇÃO *stepCAT.by* 39
QUADRO 15 – CÓDIGO *R* PARA *ADD NEW TEST* DO *CONCERTO* 47
QUADRO 16 – CÓDIGO *R* PARA *EDIT A TEST* DO *CONCERTO* 50

SUMÁRIO

1	**INTRODUÇÃO**	**10**
2	**MATERIAL E MÉTODOS**	**13**
2.1	MATERIAL	13
2.1.1	Base de dados	13
2.1.2	*R* e o *RStudio*	14
2.1.2.1	*R Packages*	14
2.1.2.2	*Package roxygen2*	15
2.1.3	Plataforma *Concerto*	15
2.2	MÉTODO	16
2.2.1	Teoria da Resposta ao Item (TRI)	16
2.2.2	Teste Adaptativo Computadorizado (TAC)	18
2.2.2.1	Calibração dos itens	19
2.2.2.2	Processo do TAC	20
3	**RESULTADOS**	**23**
3.1	OS PRINCÍPIOS DO TAC E O PACOTE *stepCAT*	23
3.1.1	*stepCAT*	23
3.1.1.1	*stepCAT.by*	24
3.1.2	TABELA DE ITENS	26
3.1.2.1	*stepCAT.items*	26
3.1.2.2	*stepCAT.table*	28
3.1.3	BANCO DE ITENS	30
3.1.3.1	*stepCAT.bank*	30
3.1.4	PASSO INICIAL	31
3.1.4.1	*stepCAT.start*	31
3.1.5	PROCESSO DO TESTE	32
3.1.5.1	*stepCAT.test*	32
3.1.5.2	*stepCAT.stop*	34
3.1.5.3	*stepCAT.choice*	35
3.1.6	PASSO FINAL	37
3.1.6.1	*stepCAT.final*	37
3.1.7	EXEMPLOS	37
3.2	O PACOTE *stepCAT* E A PLATAFORMA *CONCERTO*	42
3.2.1	ACESSAR O *CONCERTO*	42
3.2.2	CRIAR UMA CONTA DE USUÁRIO – *ACCOUNT*	42
3.2.3	INICIAR UM ESPAÇO DE TRABALHO – *WORKSPACE*	44
3.2.4	CRIAR E IMPORTAR UMA TABELA DE ITENS – *TABLES*	44

3.2.5 CRIAR UM TESTE COM O *stepCAT – TESTS* 46
3.2.6 EDITAR E EXECUTAR UM TESTE – *TESTS* 48
3.2.7 EXECUTAR UM TESTE NO *BROWSER* 51
4 CONSIDERAÇÕES FINAIS 52
REFERÊNCIAS 53
APÊNDICE 55

1 INTRODUÇÃO

O conhecimento e a habilidade são colocados a prova em diversas situações do cotidiano. Para tanto, a qualquer momento podemos estar sendo submetidos a algum tipo de exame, sendo esses informais, como a simples resposta verbal sobre a direção a se tomar feita à pessoa em uma rua da cidade; ou formais, como a resposta escrita apontada através de uma alternativa dada para a questão em um exame de qualificação profissional. Portanto, podemos pensar o princípio básico para um exame ou teste está no modo de avaliação destinado a medir o conhecimento, ou alguma habilidade, daquele que se candidata a ser, ou está sendo, examinado sob questões e respostas em um determinado assunto. Dessa forma, o resumo da avaliação através das evidências contidas nas respostas das questões (itens) em um teste configuram o escore desse teste, podendo ser um escore bruto (sem qualquer tipo de adaptação ou transformação baseado no simples número de itens respondidos corretamente) ou um escore padronizado em escala (resultado de uma transformação aplicada ao escore bruto) (THISSEN; WAINER, 2001).

Baseado nos escores obtidos, a Teoria Clássica do Teste (TCT) assume que os efeitos entre as respostas do examinando são somente devidas a variação do seu nível de habilidade - o seu traço latente, o que seria então chamado de um escore verdadeiro. Todas as outras fontes de variação se devem aos ensaios do teste e as condições externas ou internas ao examinando, quando essas influências podem ser por vezes positivas; ou por outras vezes negativas, em relação ao resultado final do teste. Em outras palavras, o escore verdadeiro é determinado pelo modo como o teste está desenhado e não tão somente pelo traço latente do examinando. Portanto, os escores verdadeiros em dois diferentes testes destinados a medir a mesma variável de habilidade são geralmente desiguais, mesmo que os testes envolvam a mesma padronização daquelas condições externas ou internas. A razão para isso é que cada teste implica seu próprio conjunto de itens e cada item tem a sua propriedade com uma dependência residual que escapa a padronização. Considerando isso, os modelos estatísticos passam a ser uma ótima alternativa no caso (DE KLERK, 2008).

Os métodos de modelagem estatística requerem uma parametrização explícita para a habilidade sob interesse, bem como para as propriedades dos itens, por meio de um modelo que relacione os dados aos valores das respostas obtidas através do teste. Se os parâmetros dos itens são conhecidos, o modelo ajusta os dados para as propriedades dos itens em teste, e, portanto, pode ser usado para produzir as medidas de habilidade que são então livres daquelas propriedades. Nesse contexto, os modelos estatísticos tem sido desenvolvidos na Teoria da Resposta ao Item (TRI), do inglês *Item Response Theory* (IRT). O mais conhecido modelo de TRI para res-

postas dicotômicas (resposta correta ou incorreta) por exemplo, considera que essas medidas de habilidade ou características latentes, chamadas traços, possuem relação probabilística com os itens utilizados no teste e os parâmetros de dificuldade, poder de discriminação e a probabilidade de acerto casual de cada item (VAN DER LINDEN; HAMBLETON, 1997).

Para os fins dos Testes Adaptativos Computadorizados (TAC), do inglês *Computer Adaptive Tests* ou *Computerized Adaptive Testing* (CAT), muito embora sejam úteis por si próprios, os três parâmetros - a dificuldade, o poder de discriminação e o acerto casual - de cada item em teste são combinados em uma Função de Informação do Item (FII), ao contrário da TCT onde a informação está completamente vinculada ao resultado final do teste. Assim, uma FII é calculada a partir dos parâmetros dos itens e descreve qual é a medida do item em cada nível de habilidade. Na TRI a habilidade é denotada como θ e é estimada durante o processo do teste. Dessa forma, a partir de um banco de itens com os parâmetros previamente calibrados pela TRI, um TAC seleciona cada item seguinte com base nas respostas marcadas nos itens anteriores, da seguinte forma: se o item é respondido corretamente a estimativa original de θ será aumentada; se o item é respondido incorretamente a estimativa será reduzida na mesma proporção. Com isso, depois de cada novo item ser administrado e respondido, a nova estimativa de θ é usada para selecionar o próximo item, que deve maximizar a informação fornecida ao examinando. O mesmo processo se repete e a cada passo um ou mais critérios de parada são consultados - normalmente o valor mínimo para o erro padrão ou um número máximo de itens administrados. Então, quando um dos critérios for cumprido, o teste está terminado e a estimativa final de θ e seu erro padrão são registrados para o examinando (THOMPSON; WEISS, 2011).

Diante disso, a principal vantagem de um TAC seria a capacidade de conceber e fornecer testes eficientes que medem os examinandos com igual nível de precisão. Isso significa dizer que em um TAC devidamente desenhado para este objetivo de medição, todos os examinandos serão medidos com um mesmo erro padrão ou uma duração mínima no teste, o que não seria facilmente conseguido com algum outro tipo de teste. Obviamente, devido ao uso extensivo de cálculos em tempo real necessários para implementar os TACs, eles não podem ser realizados por quaisquer outros meios que não sejam os computadores (WEISS, 2011).

Nesse aspecto computacional, o *R* como um *software* fornece através de sua linguagem de código aberto [1] um ambiente no qual as técnicas estatísticas mencionadas podem ser implementadas através de pacotes nativos ou obtidos adicionalmente. Para isso, o pacote adicional *catR* oferece regras e rotinas para a seleção de itens, estimação de habilidade e critérios de parada, os quais são os componentes principais

[1] O termo código aberto, do inglês *open source*, foi criado pela OSI (*Open Source Initiative*) e se refere a *software*, também conhecido por *software* livre (*free software*).

do algoritmo em um TAC (MAGIS; GILLES, 2012).

No entanto, apesar dos recursos computacionais permitirem que as técnicas desenvolvidas sob um TAC resultem em complexos algoritmos, o mesmo TAC precisa ser simples e atraente quando usado pelo examinador ou apresentado ao examinando. Para isso, a plataforma *online* e *open source* Concerto oferece a integração completa para o uso do *R*, a flexibilidade na apresentação do conteúdo em *HTML*[2] e a facilidade no compartilhamento dos resultados através da web[3] (CONCERTO TESTING PLATFORM, 2012), possibilitando com isso a conectividade para outros testes.

Entretanto, apesar da robusta integração sob a plataforma *Concerto*, o desenvolvimento do algoritmo para o TAC ainda seria um desafio, pois a ausência de uma literatura especializada, principalmente em língua portuguesa, somado ao obstáculo operacional da aplicação do processo, devido a necessidade de uma estrutura física composta computadores interligados através uma rede de dados segura, acabam colocando o TAC muito distante do público menos experiente, e dessa forma, impactando consideravelmente a sua ampla utilização em um nível profissional e menos didático. Isso se torna evidente observando que as mais conhecidas aplicações de TAC no Brasil, são principalmente desenvolvidas para os exames de proficiência em idiomas, ou ainda, em estudos teóricos, como a proposta de avaliação de novos condutores no Departamento Nacional de Trânsito (DETRAN) do estado de Santa Catarina (MOREIRA JUNIOR, 2011). Contudo, noticia-se que o Exame Nacional do Ensino Médio (ENEM) e outros vestibulares de instituições federais e estaduais, tem visto com bons olhos o potencial de aplicação de um TAC em suas avaliações. No entanto, o desenvolvimento é embrionário e ainda esbarra em inúmeras questões legislativas, práticas e tecnológicas.

Portanto, levando em consideração todos esses aspectos mas em um contexto menos amplo, o objetivo do trabalho está em flexibilizar o uso de um TAC ao usuário convencional, demonstrando como os princípios do TAC deram origem ao pacote *stepCAT*, como este é assistido pelo pacote *catR* e, finalmente, como se torna possível integrar todos sob a *Plataforma Concerto*, e assim, tornar possível a configuração e publicação de um teste adaptativo de uma forma simples e prática.

---

[2] O termo *HTML*, do inglês *Hyper Text Markup Language*, é uma linguagem de programação usada para produzir páginas para apresentação em navegadores (*web browsers*) para a *internet.*

[3] O termo *web*, derivado do inglês *World Wide Web*, é um sistema de páginas interligadas *online* sobre a *internet*

2 MATERIAL E MÉTODOS

2.1 MATERIAL

2.1.1 Base de dados

Os dados utilizados no trabalho foram elaborados através da coleta em atividade sugerida na disciplina de Teoria da Resposta ao Item, ministrada pelo Prof. Adilson dos Anjos durante o 1º semestre de 2013 para o curso de Estatística da UFPR.

O objetivo estava em medir o traço latente do ”Conhecimento sobre clubes e seleções de futebol”, através de cada item para uma resposta dicotômica em uma questão de múltipla-escolha composta por 5 alternativas, sendo uma o gabarito (resposta correta) e as outras os distratores. Os itens foram sugeridos seguindo uma matriz de referência com os seguintes temas (FORBELLONE; INÁCIO, 2013):

a) clubes nacionais;
b) clubes internacionais;
c) seleção nacional do Brasil;
d) outras seleções nacionais.

Nos temas acima foram propostos os seguintes descritores, ou seja, assuntos sobre o tema em questão que podiam ser abordados:

a) principais conquistas (títulos e campeonatos);
b) rivalidades entre clubes e seleções (objetivo);
c) datas (ano) das principais conquistas;
d) principais jogadores (exceto atuais);
e) principais técnicos (exceto atuais);
f) mascote, símbolos, uniforme (cores);
g) principais jogos realizados.

A base de dados está composta de 80 itens desenvolvidos pelos alunos do curso, tendo sido aplicados em um grupo de 710 examinandos entre os meses de maio e junho de 2013, através de um formulário eletrônico do *GoogleForms* [1].

[1] Consultar em *https://support.google.com/drive/answer/87809?hl=pt-BR*

2.1.2 *R* e o *RStudio*

O aspecto exploratório do trabalho sobre os métodos estatísticos está executado através do *R* e o *RStudio*.

O *R*, disponibilizado como um *software* livre (*free software*) sob os termos de *GNU - General Public License*[2], é um conjunto integrado de programas para a manipulação de dados, cálculos e exibição gráfica que formam uma linguagem e um ambiente para a computação estatística. O *R* permite ser extensível através do uso de pacotes desenvolvidos por seus usuários para funções ou áreas específicas de estudo (R DEVELOPMENT CORE TEAM, 2011).

O *RStudio*[3] é um ambiente integrado de desenvolvimento livre e de código aberto para *R*. Como um sistema multiplataforma é composto por ferramentas que aumentam produtividade no desenvolvimento de programas, como a interface com o destaque na sintaxe do código em uso, formas de auto-completar a codificação e a indentação inteligente (RSTUDIO, 2013).

O contexto do trabalho assume que o leitor saiba fazer uso do *R*, através de sua *Console* ou com o auxílio do *RStudio*. Portanto, cabe ao autor a sugestão de uma busca rápida em outras fontes sobre o assunto, ou partir de uma introdução ao *R* na *internet*, por exemplo em *http://cran.r-project.org/doc/manuals/R-intro.html* com conteúdo em língua inglesa.

2.1.2.1 *R Packages*

O sistema de pacotes do *R* é um dos principais diferencias do ambiente, por assim dizer, pois tornam o compartilhamento de artigos e programas algo simples e prático, tanto para o desenvolvedor, quanto para o meio de publicação e o usuário final.

O sistema de pacotes pode estar relacionado à um *software* de um artigo, onde a ferramenta que se fez uso para a exploração do tema é facilmente acessada, e as análises feitas naquele artigo podem ser replicadas e experimentadas pelo próprio leitor (LEISCH, 2009). Seguindo esse princípio da reutilização e colaboração sugerida com o uso de pacotes, é tomado como base o pacote pacote *catR*, especializado em TAC baseado na TRI(MAGIS; GILLES, 2012), para desenvolver e propor um processo mais robusto e flexível para a aplicação e manipulação de um teste adaptativo.

[2] Consultar em *http://www.r-project.org/COPYING*.
[3] Disponível em *http://www.rstudio.com*

2.1.2.2 *Package roxygen2*

O requisito fundamental para um pacote *R* é ter em todas as funções, objetos e dados uma documentação completa e clara. Para isso é possível usar ferramentas específicas do *RStudio* que facilitam essa tarefa, como é o caso do *roxygen2* [4].

O pacote *roxygen2* oferece um sistema completo de documentação de código fonte para *R*, que gera automaticamente os arquivos de documentação (extensão de arquivo .Rd) (WICKHAM; DANENBERG; EUGSTER, 2013). O arquivo .Rd, do termo "*R Documentation*", consiste em um cabeçalho com: um título - uma breve descrição textual e a instrução para o modo de usar o objeto documentado, um corpo - com os argumentos para o uso e os valores de retorno, e um rodapé opcional com as palavras-chave [5].

Pela coerência do formato proposto por *R Documentation* e para a conveniência do leitor, algumas seções seguintes utilizam uma organização semelhante no seu conteúdo.

2.1.3 Plataforma *Concerto*

O trabalho implementa os métodos estatísticos, as técnicas e os algoritmos de um TAC sob a Plataforma *Concerto*.

O *Concerto* é uma plataforma de código-aberto (*open-source*) disponibilizada para o uso livre em nível acadêmico e comercial, que permite ao seu usuário criar e hospedar *online* através da *web* vários tipos de testes.

O mecanismo do *R* e o uso dos pacotes *R* totalmente integrado ao *Concerto* permitem desenvolver, implementar e apresentar um TAC baseado na TRI (CONCERTO TESTING PLATFORM, 2012).

---

[4] Disponível em *http://cran.r-project.org/web/packages/roxygen2/index.html*

[5] Consultar em *http://cran.r-project.org/doc/manuals/R-exts.htmlDocumenting-packages*

2.2 MÉTODO

2.2.1 Teoria da Resposta ao Item (TRI)

Conceitualmente a TRI é um grupo de modelos probabilísticos, os quais são utilizadas para descrever características de indivíduos que não podem ser diretamente observadas, podendo também ser chamada de Teoria do Traço Latente. Na TRI, o procedimento de medida do avaliando considera que essas características latentes, chamadas traços, possuem relação probabilística com os itens utilizados no teste (VENDRAMINI; SILVA, 2004).

Na TRI cada item (questão) apresenta características/parâmetros que junto da proficiência dos examinandos seguem uma função de probabilidade de acerto (BAKER, 2001). Nos testes que se utilizam de itens objetivos, tem se como o modelo mais utilizado o Modelo Logístico de 3 Parâmetros Unidimensional, como, por exemplo, o aplicado em avaliações do Exame Nacional do Ensino Médio (ENEM).

O Modelo de 3 Parâmetros pode ser ilustrado através da Curva Característica do Item (CCI) na FIGURA 1, onde: o parâmetro "a" determina a discriminação (inclinação) do item i, com valor proporcional à inclinição no ponto "b_i"; o parâmetro "b" expressa a dificuldade (posição) do item na mesma escala da habilidade; o parâmetro "c" considera a probabilidade de um acerto casual - ou um indivíduo de baixa habilidade acertar o item ao acaso; e ainda, o valor θ representa a habilidade (traço latente) do examinando. Dessa forma, a função que define a probabilidade de acerto $P(U_{ij} = 1)$ de um item "i", dado um examinando "j" com a habilidade θ_j (ANDRADE; TAVARES; VALLE, 2000), é definida por:

$$P\left(U_{ij} = 1 \mid \theta_j, a_i, b_i, c_i\right) = c_i + \left(1 - c_i\right) \frac{1}{1 + e^{-a_i(\theta_j - b_i)}}$$

onde:

a_i – parâmetro de discriminação;

b_i – parâmetro de dificuldade;

c_i – probablidade de acerto casual;

θ_j – habilidade do examinando.

Por sua vez, a Curva de Informação do Item (CII) (FIGURA 2) permite analisar a quantidade de informação em um item para a medida do traço latente. Portanto, se pode dizer que quanto maior for a informação do item melhor será sua qualidade

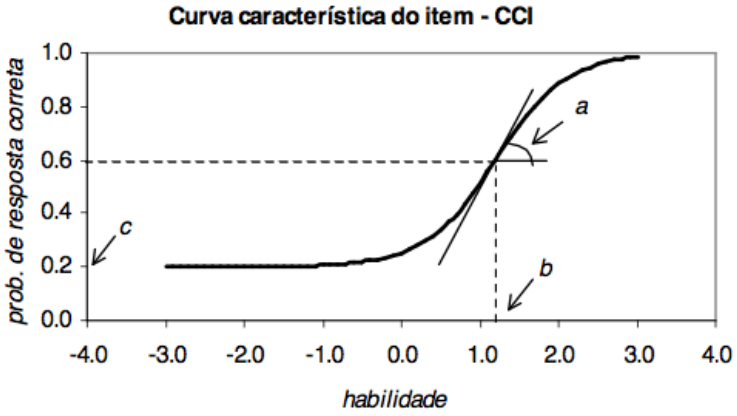

FIGURA 1 – CURVA CARACTERÍSTICA DO ITEM – CCI
Fonte: (ANDRADE; TAVARES; VALLE, 2000)

ou, em outras palavras, em um Modelo de 3 Parâmetros: quanto maior for o valor de "a" e menor o valor de "c", a informação do item será maximizada em uma certa habilidade "b". Dessa forma, se pode perceber como a CCI, ou mais especificamente, quando calculada à partir da Função de Informação do Item (FII), pode constituir-se em métodos para alguns tipos de TACs (THOMPSON; WEISS, 2011), onde podem, por vezes, usar internamente em seu algoritmo a expressão dada por:

$$I_i(\theta) = a^2 \left[\frac{Q_i(\theta)}{P_i(\theta)}\right] \left[\frac{P_i(\theta) - c^2}{(1-c^2)}\right]$$

onde:

$$P_i(\theta) = c + (1-c)\left(\frac{1}{(1+EXP(-L))}\right); L = a(\theta - b); Q_i = 1.0 - P_i(\theta);$$

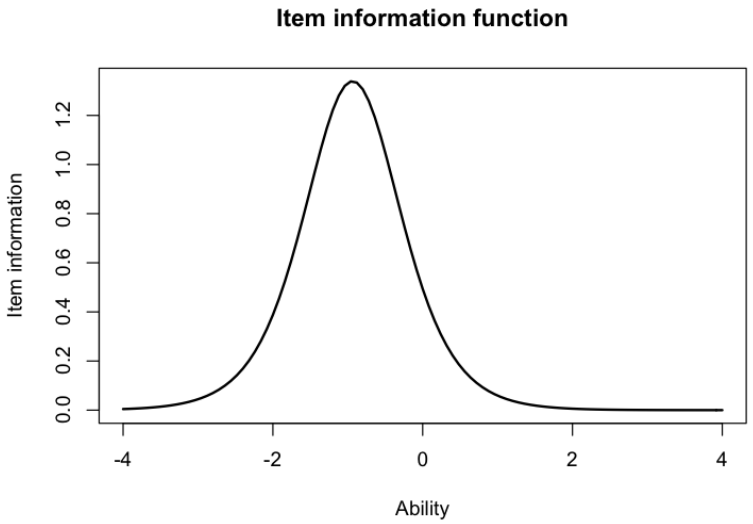

FIGURA 2 – CURVA DE INFORMAÇÃO DO ITEM – CII
Fonte: O autor (2013)

2.2.2 Teste Adaptativo Computadorizado (TAC)

Os TACs permitem implementar uma seleção inteligente para o item de avaliação em um teste (WAINER; DORANS, 2000). Ao contrário dos testes sequenciais nos quais são usados tipicamente uma ordem fixa dos itens e um mesmo teste é aplicado para todos os examinados, o TAC permite que cada examinando possa começar o seu teste a partir de um diferente item em relação a um outro examinando, e assim continue a receber diferentes itens de um conjunto de itens estabelecidos para o teste. No entanto, apesar de haver uma grande variedade de métodos para TACs, em algum momento todos eles selecionam dinamicamente os itens para serem administrados à cada examinado, tendo se baseado na resposta dada para o item anterior do teste (WEISS, 1985).

Sob ponto de vista construtivo, o TAC é composto por 5 componentes. O primeiro componente é um banco de itens calibrado, e é portanto, desenvolvido como o conteúdo do teste. Os quatro componentes restantes são psicométricos em detrimento ao conteúdo, e se referem a algoritmos usados no sistema de TAC.

a) tabela e/ou banco de itens calibrados;
b) critério de partida;
c) seleção do próximo item;
d) estimador de habilidade;
e) critério de parada.

Um TAC opera tomando os dois primeiros componentes (*a* e *b*) tal como são dados, em seguida processa ciclicamente através dos próximos componentes (*c*, *d* e *e*) até que o critério de parada seja satisfeito (WEISS, 2011).

Apesar de existirem vários métodos diferentes para calcular as estatísticas de cada um destes três componentes, Weiss (2011) em um dos seus artigos menciona que os TACs mais flexíveis e, portanto, mais eficientes são os TACs totalmente adaptativos baseados em Teoria da Resposta ao Item (TRI). Tais TACs são baseados na FII da TRI, calculada a partir dos parâmetros do item - dificuldade ("a"), poder de discriminação ("b") e acerto casual ("c") (RUDNER, 1998), como mencionado anteriormente.

O uso da FII permite que cada examinando possa ter o critério de partida do seu teste através de um item diferente, se uma estimativa provisória de proficiência ("a priori") esteja disponível (MOREIRA JUNIOR, 2011). Então, com base em sua resposta a esse item, a estimação de habilidade é calculada para aquele examinando e expressa na escala do traço latente (θ) da TRI, utilizando os métodos de estimação

que levam em conta qual resposta o examinando deu ao item mediante os parâmetros definidos para esse item. A estimação atualizada é então utilizada para selecionar outro item ainda não administrado a partir do banco de itens calibrados, sendo aquele item que proporciona o máximo de informação para o examinando, o qual também é o item que maximamente reduz a incerteza associada com a estimativa θ, conforme expresso no erro padrão individualizado associado com a estimativa do θ. Então, um ou mais critérios de parada são consultados - esses normalmente são o valor mínimo para o erro padrão ou um número máximo de itens administrados. Se o examinando não cumpriu um dos critérios de parada, a atual estimativa de θ é usada para selecionar o próximo melhor item e assim o processo continua. Quando um critério de parada for cumprido, seja pela quantidade de itens administrados ou pela precisão na estimativa da habilidade, o teste está terminado e a estimativa final do θ e seu erro padrão são registrados para aquele examinando (THOMPSON; WEISS, 2011).

2.2.2.1 Calibração dos itens

O desenvolvimento do trabalho, considerando os princípios do TAC, necessita de uma calibração nos itens à partir da base de dados. Para isso, utilizou-se um Modelo Logístico de 3 Parâmetros Unidimensional para respostas dicotômicas, onde foram aplicados os seguintes filtros:

a) correlação bisserial inferior a 0.2, demonstrando assim uma baixa relação do item com o escore;

b) parâmetro de discriminação (parâmetro a) inferior a 0.6, demonstrando um baixo poder de discriminação do item;

Com os filtros aplicados foram eliminados 17 itens e mantidos 63 itens calibrados. Com isso, ao analisar a curva formada pela Função de Informação do Teste (FIT) (FIGURA 3), se percebe no ponto de máximo da função um pequeno deslocamento em relação à zero, assim como uma simetria. Também, se pode observar, que os itens permitem avaliar a informação de examinandos que possuem um traço latente entre 0,5 desvios abaixo da média e 1 desvio acima da média.(FORBELLONE; INÁCIO, 2013)

Os itens agora calibrados constituem, com os seus parâmetros, o ”banco de itens” (ou *item bank*). Então cada item, e seus parâmetros, é associado ao texto da questão, às alternativas para a questão e à um índice da alternativa correta, configurando assim a ”tabela de itens” disponibilizada on-line em *https://raw.github.com/hkrefer/stepCAT/master/data/futebol.csv*, que pode ser usada em alguns exemplos, mesmo no computador do leitor, como a tabela ”futebol.csv”.

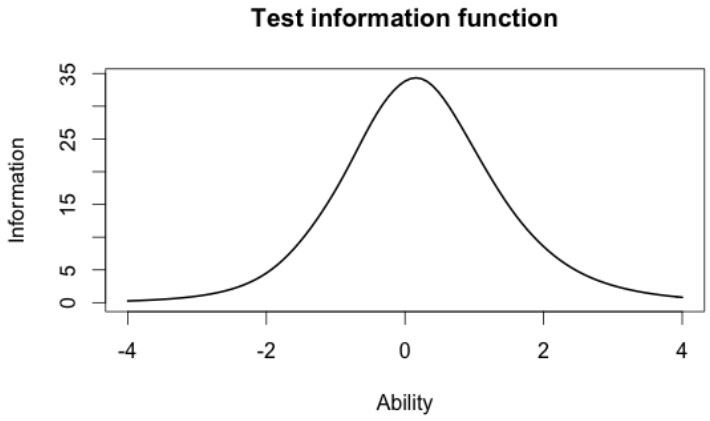

FIGURA 3 – FUNÇÃO DE INFORMAÇÃO DO TESTE
Fonte: O autor (2013)

2.2.2.2 Processo do TAC

Além da teoria estatística, O TAC exige um fluxo operacional para sua correta aplicação prática. Para isso, as especificidades devem ser consideradas desde a publicação do teste pelo examinador, passando pelas respostas do examinando através do processamento da parte sistêmica e até uma possível consulta posterior daquele teste FIGURA 4.

Entretanto, o TAC pode ser simplificado em um sub-processo composto de 3 passos importantes para a sua definição:

a) Passo inicial

No passo inicial é feita a seleção dos primeiros itens do teste. Como nesse momento ainda não temos nenhuma informação da habilidade do examinando, o teste deve seguir os critérios estipulados pelo examinador. O principal critério é definir quantos itens devem ser aplicados nesse passo e qual o critério de seleção deve ser usado, seja através de uma lista pré-definida ou da estimativa de um θ inicial. Utiliza-se, em exemplos mais adiante, uma seleção inicial com 3 itens equidistantes de um θ com valor 0, selecionados através do critério de Máxima Informação de Fisher, padrão nas funções *startItems* e *nextItems* do pacote *catR* pelo argumento *startSelect* ou *criterion* como ”MFI” (de *Maximum Fisher Information*). Com uma aplicação prática, nota-se que a seleção inicial é basicamente a mesma, devido aos poucos itens disponíveis no banco de itens didático.

b) Processo do Teste

Após a aplicação dos itens iniciais é feito a primeira estimação de habilidade, dado o desempenho do examinando na primeira etapa. Utiliza-se, em exemplos mais adiante, utiliza-se o método de Bayes, padrão na função

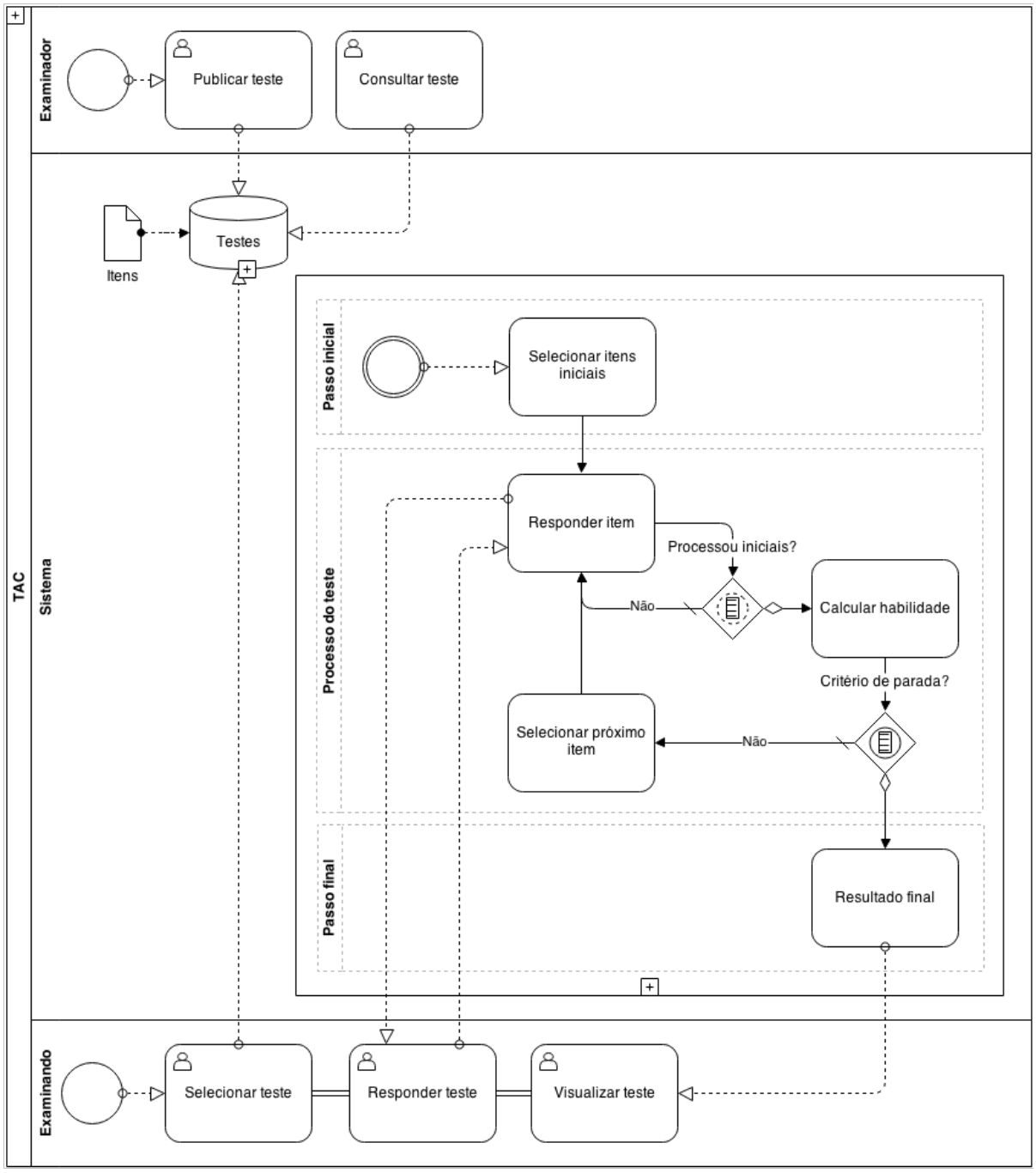

FIGURA 4 – PROCESSO DO TAC
Fonte: O autor (2013)

thetaEst do pacote *catR* pelo argumento *method* como ”BM” (de *Bayes Modal*). Outros critérios e métodos podem ser utilizados para a seleção dos itens (MAGIS; GILLES, 2012), e podem ser vistos tanto nas funções do pacote *catR*, por exemplo em *http://cran.r-project.org/web/packages/catR/*, como na literatura disponível.

– o critério de parada pode ser feito tanto sobre a quantidade de itens aplicados, garantindo que o teste tenha um tamanho único para todos

os examinandos; ou pelo intervalo de confiança do θ estimado, garantindo assim que o teste utilize a precisão como critério para medir o traço latente. Com um princípio didático, adotou-se como padrão um tamanho de teste limitado em 10 itens administrados, e assim evitar um teste prolongado.

c) Passo Final

Após o teste ter sido aplicado e o critério de parada satisfeito, o resultado pode ser analisado através de uma lista com:

- os parâmetros dos itens administrados;
- o valor das respostas (como 0 ou 1);
- o valor da estimativa da habilidade;
- o valor do erro padrão médio.

3 RESULTADOS

3.1 OS PRINCÍPIOS DO TAC E O PACOTE *stepCAT*

Construído sob os princípios do TAC e seus componentes, o pacote *stepCAT* para o uso no *R* está estruturado à partir do comando *stepCAT.by*, descrito na seção 3.1.1.1, e tem seu algoritmo assistido pelo pacote *catR* (MAGIS; GILLES, 2013).

No contexto do trabalho o pacote está apresentado nas seções seguintes para cada comando (ou função), o seu modo de usar através da sintaxe, os argumentos requeridos e opcionais, os detalhes desses argumentos e os valores retornados pelo comando como resultado. Demonstrando assim como o pacote *stepCAT* pode ser usado para a aplicação e simulação de um TAC, ou através de suas partes integradas em algoritmos particulares para *softwares* mais sofisticados, como a proposta de combiná-lo com a plataforma *Concerto*.

3.1.1 *stepCAT*

O pacote *stepCAT* está publicado no GitHub[1], em um repositório colaborativo para a construção de *software*, e pode ser usado sob os termos e condições da licença *GPLv2 – GNU General Public License version 2)*[2] no R a partir de:

a) instalado com o auxílio do pacote *devtools*[3] (QUADRO 1).

```
library(devtools)
install_github("stepCAT", "hkrefer")
```

QUADRO 1 – INSTALAÇÃO DO PACOTE *stepCAT*
FONTE: O autor (2013)

[1] Consultar em *https://github.com/about*
[2] Consultar em *http://www.gnu.org/licenses/gpl-2.0.html*
[3] Disponível em *http://cran.r-project.org/web/packages/devtools/index.html*

b) carregado como código-fonte na sessão atual com o auxílio do pacote *RCurl*[4] (QUADRO 2).

```
library(RCurl)
url <- "https://raw.github.com/hkrefer/stepCAT/master/R/stepCAT.R"
code <- textConnection(getURL(url, followlocation = TRUE, ssl.verifypeer =
    FALSE))
source(code)
```

QUADRO 2 – LEITURA DO PACOTE *stepCAT* COMO CÓDIGO-FONTE
FONTE: O autor (2013)

3.1.1.1 *stepCAT.by*

A função *stepCAT.by* atua como um integrador para construir e aplicar um teste adaptativo. Dado um banco de itens calibrados à partir de uma tabela de dados e usando algumas das funções no pacote *catR*, aquela função administra o teste através de uma das opções de interface gráfica do usuário.

O modo de usar a função, através da sua sintaxe (QUADRO 3), está definido pelo método a ser usado à partir da classe do objeto (*method for class*) e dos argumentos listados entre os parênteses e separados por vírgulas, considerando o elemento:

a) *table* - como um objeto e sua classe para selecionar o método;

b) *...* - para informar os argumentos adicionais nas funções.

```
stepCAT(table, ...)

## S3 method for class 'data.frame'
 stepCAT(table, ...)

## S3 method for class 'character'
 stepCAT(table, ...)

 ## S3 method for class 'numeric'
 stepCAT(table, ...)
```

QUADRO 3 – SINTAXE DA FUNÇÃO *stepCAT.by*
FONTE: O autor (2013)

Os argumentos adicionais são usados nas funções similarmente aos princípios do TAC, portanto, podem ser considerados como cinco etapas fundamentais:

[4] Disponível em *http://cran.r-project.org/web/packages/RCurl/index.html*

a) tabela de itens:
   - para o objeto *table* da classe *data.frame* ou da classe *numeric*, usa os argumentos na função *stepCAT.items*, como descrito na seção 3.1.2.1;
   - para o objeto *table* da classe *character*, usa os argumentos na função *stepCAT.table*, como descrito na seção 3.1.2.2;

b) banco de itens:
   - usa os argumentos na função *stepCAT.bank*, como descrito na seção 3.1.3.1;

c) passo inicial:
   - usa os argumentos na função *stepCAT.start*, como descrito na seção 3.1.4.1;

d) processo do teste:
   - usa os argumentos na função *stepCAT.test*, como descrito na seção 3.1.5.1;

e) passo final:
   - usa os argumentos na função *stepCAT.final*, como descrito na seção 3.1.6.1.

O resultado da função é retornado em uma lista e cada valor atribuído em um objeto, descrito como:

a) *start* (o vetor de elementos usados no passo inicial);
   - *start items* (o índice de cada item);
   - *start par* (uma matriz dos parâmetros de cada item);
   - *start par "a"* (parâmetro de discriminação);
   - *start par "b"* (parâmetro de dificuldade);
   - *start par "c"* (parâmetro de acerto casual);
   - *start par "d"* (parâmetro de distração);
   - *start thStart* (sequência do valor de habilidade para o item inicial);
   - *start startSelect* (caracter para o critério de seleção do item inicial);

b) *test* (vetor de elementos usados no processo do teste):
   - *test items* (índice de cada item);
   - *test par* (matriz dos parâmetros de cada item);
   - *test par "a"* (parâmetro de discriminação);
   - *test par "b"* (parâmetro de dificuldade);
   - *test par "c"* (parâmetro de acerto casual);

- *test par "d"* (parâmetro de distração);
- *test* x (respostas com valores 0 ou 1);
- *test theta* (estimativa da habilidade);
- *test SE* (estimativa do erro padrão);
- *test CI* (estimativa do intervalo de confiança);

3.1.2 TABELA DE ITENS

3.1.2.1 *stepCAT.items*

A função *stepCAT.items* configura uma "tabela de itens" pelo nome das colunas (parâmetros) e os tipos dos seus valores - ou cria para um determinado número de itens, usando a função *createItemBank* do pacote *catR*.

O modo de usar a função, através da sua sintaxe (QUADRO 4), está definido pelo método a ser usado à partir da classe do objeto (*method for class*) e dos argumentos listados entre os parênteses e separados por vírgulas, considerando o elemento:

a) *items* - como um objeto e sua classe para selecionar o método;

b) *...* - para informar os argumentos adicionais na função;

c) *id* - como um valor *TRUE* (verdadeiro), se o índice dos itens está na primeira coluna do objeto *items*;

d) *choices* - como a quantidade de colunas no objeto *items*, contendo as alternativas para a questão do item;

e) *model* - como um texto pré-definido, indicando o Modelo Logístico da TRI para criar o objeto *items*;

A função genérica *stepCAT.items* possui dois métodos, definidos à partir de um:

a) objeto *items* da classe *numeric*, assumindo que:
   - o argumento *items* deve ser um inteiro positivo, ou então o valor padrão (50) é assumido para gerar um banco de itens;
   - o argumento *choices* deve ser um número inteiro positivo, ou então o valor padrão (5) é assumido para gerar valores *NA*;

```
 stepCAT.items(items, ...)

 ## S3 method for class 'numeric'
stepCAT.items(items = 50, id = FALSE,
   choices = 5, model = "3PL", ...)

 ## S3 method for class 'data.frame'
stepCAT.items(items, id = FALSE,
   choices = 5, ...)
```

QUADRO 4 – SINTAXE DA FUNÇÃO *stepCAT.items*
FONTE: O autor (2013)

- o argumento *model* e os argumentos adicionais (em ...) podem ser consultados na função *createItemBank* do pacote *catR*.

b) objeto *items* da classe *data.frame*, assumindo que:

- o argumento *items* deve ser um *data frame* com um item por linha e, no mínimo, 7 colunas nomeadas na seguinte ordem e sugestão: "a", "b", "c", "d", "r", "q" e "$ch[1:n]$ ", respectivamente para (os parâmetros de) discriminação, dificuldade, acerto casual, distração, o valor da resposta, o texto da questão e as alternativas (uma por coluna até o valor "n", dado pelo argumento *choices*);
- se o argumento *id* assume o valor *TRUE* (verdadeiro), então a primeira coluna deve ser um vetor de índices;
- o número de colunas na tabela de dados deve ser igual as colunas descritas mais as alternativas, ou então a função é interrompida por um erro;
- o argumento *choices* deve ser um número inteiro positivo, ou então o valor padrão (5) é assumido.

O resultado da função é retornado com cada valor atribuído em um elemento, descrito como:

a) *id* (índice opcional pelo argumento *id*);

b) *a* (parâmetro de discriminação);

c) *b* (parâmetro de dificuldade);

d) *c* (parâmetro de acerto casual);

e) *d* (parâmetro de distração);

f) *r* (número da alternativa correta - entre 1 e *n*);

g) *q* (texto da questão);

h) *ch1* (texto da alternativa *1*);

i) *ch:* (texto das alternativas entre 1 e *n*);

j) *chn* (texto da alternativa *n* - valor dado pelo argumento *choices*).

3.1.2.2 *stepCAT.table*

O *stepCAT.table* lê os dados de uma tabela e configura em uma ”tabela de itens” usando a função *stepCAT.items*, como descrito na seção 3.1.2.1.

O modo de usar a função, através da sua sintaxe (QUADRO 5), está definido pelo método a ser usado à partir da classe do objeto (*method for class*) e dos argumentos listados entre os parênteses e separados por vírgulas, considerando o elemento:

a) *table* - como o nome da tabela, com os dados que devem ser lidos;

b) *DB* - como um objeto e sua classe para selecionar o método;

c) *...* - para informar os argumentos adicionais na função;

d) *id* - como um valor *TRUE* (verdadeiro), se o índice dos itens está na primeira coluna do objeto *table*;

e) *choices* - como a quantidade de colunas no objeto *items*, contendo as alternativas para a questão do item;

f) *header* - como o valor *TRUE* (verdadeiro), para usar os nomes das colunas como está na primeira linha do objeto *table*.

g) *sep* - como o separador de valores nas colunas no objeto *table*;

h) *encoding* - como a codificação padrão, para lidar com seqüências codificadas no objeto *table*;

i) *tableWorkspaceID* - como o identificador (ID) do espaço de trabalho na plataforma *Concerto*, onde está localizado o objeto *table* (se aplicável ao método selecionado);

j) *file.force* - como o valor *TRUE* (verdadeiro), para forçar a leitura dos dados do objeto *table*, como no método *file*.

A função genérica *stepCAT.table* possui dois métodos, definidos à partir de um:

a) objeto *DB* da classe *file*, assumindo que:

   – o argumento *choices* deve ser um numero inteiro positivo, caso contrário o valor padrão será assumido;

```
 stepCAT.table(table, DB = "file", ...)

 ## S3 method for class 'file'
stepCAT.table(table, id = FALSE,
   choices = 5, header = TRUE, sep = ";",
   encoding = "UTF-8", ...)

 ## S3 method for class 'concerto'
stepCAT.table(table, id = FALSE,
   choices = 5, tableWorkspaceID = NULL,
   file.force = FALSE, ...)
```

QUADRO 5 – SINTAXE DA FUNÇÃO *stepCAT.table*
FONTE: O autor (2013)

- os argumentos adicionais (em *...*) podem ser consultados na função *read.table* do pacote *utils*;
- a especificação para a tabela de dados pode ser consultada na função *stepCAT.items*, na seção 3.1.2.1 nos termos da classe *data.frame*.

b) objeto *DB* da classe *concerto*, assumindo que:

- o argumento *choices* deve ser um numero inteiro positivo, caso contrário o valor padrão (5) será assumido;
- o argumento *tableWorkspaceID* deve ser um numero inteiro positivo, caso contrário o valor *NULL* será assumido;
- se o argumento *tableWorkspaceID* tiver o valor *NULL*, então a variável de sessão *concerto$workspaceID* da plataforma *Concerto* é assumido;
- a especificação para a tabela de dados pode ser consultada na função *stepCAT.items*, na seção 3.1.2.1 nos termos da classe *data.frame*.

O resultado da função é retornado com cada valor atribuído em um elemento, descrito como:

a) *id* (índice opcional pelo argumento *id*);

b) *a* (parâmetro de discriminação);

c) *b* (parâmetro de dificuldade);

d) *c* (parâmetro de acerto casual);

e) *d* (parâmetro de distração);

f) *r* (número da alternativa correta - entre 1 e *n*);

g) *q* (texto da questão);

h) *ch1* (texto da alternativa *1*);

i) *ch:* (texto das alternativas entre 1 e *n*);

j) *chn* (texto da alternativa *n* - valor dado pelo argumento *choices*).

3.1.3 BANCO DE ITENS

3.1.3.1 *stepCAT.bank*

A função *stepCAT.bank* extrai os elementos de um *data frame* pelo nome das colunas, à partir de uma ”tabela de itens”, criando um ”banco de itens” (*item bank*) usando a função *createItemBank* do pacote *catR*.

O modo de usar a função, através da sua sintaxe (QUADRO 6), está definido pelos argumentos listados entre os parênteses e separados por vírgulas, considerando o elemento:

a) *items* - como um *data frame* por colunas nomeadas como ”a”, ”b”, ”c” e ”d” para os parâmetros;

b) *...* - para informar os argumentos adicionais na função.

```
stepCAT.bank(items, ...)
```

QUADRO 6 – SINTAXE DA FUNÇÃO *stepCAT.bank*
FONTE: O autor (2013)

Os argumentos são usados na função considerando:

a) a especificação para o *data frame* na função *stepCAT.items*, na seção 3.1.2.1 nos termos da classe *data.frame*;

b) os argumentos adicionais (em *...*) na função *createItemBank* do pacote *catR*.

O resultado da função é retornado em um objeto, como uma lista com a classe *itBank*, contendo os elementos *itemPar*, *theta*, *infoTab* e *cbGroup*, os quais podem ser consultados os detalhes na função *createItemBank* do pacote *catR*.

3.1.4 PASSO INICIAL

3.1.4.1 *stepCAT.start*

A função *stepCAT.start* seleciona os itens iniciais para o teste adaptativo, à partir de um ”banco de itens”, usando a função *startItems* do pacote *catR*.

O modo de usar a função, através da sua sintaxe (QUADRO 7), está definido pelos argumentos listados entre os parênteses e separados por vírgulas, considerando o elemento:

a) *itemBank* - como o objeto da classe *itBank*, retornado pela função *createItemBank*;

b) *nrItems* - como o numero de itens selecionados do objeto *itemBank*;

c) *theta* - como o nível de habilidade inicial, para os itens selecionados do objeto *itemBank*;

d) *halfRange* - como intervalo de habilidade, para seleção do item inicial do objeto *itemBank*;

e) *startSelect* - como o critério de seleção, para os itens iniciais do objeto *itemBank*;

f) *seed* - para fixar a seleção aleatória, pela origem de um número aleatório;

g) *...* - para informar os argumentos adicionais na função.

```
stepCAT.start(itemBank, nrItems = 1, theta = 0,
    halfRange = 2, startSelect = "MFI", seed = NULL, ...)
```

QUADRO 7 – SINTAXE DA FUNÇÃO *stepCAT.start*
FONTE: O autor (2013)

Os argumentos são usados na função considerando:

a) o argumento *NrItems* deve ser um número inteiro positivo, ou então o valor padrão (1) é assumido e a seleção aleatória do número de origem é fixado como 1;

b) o valor mínimo entre o argumento *nrItems* e o valor do elemento *itemBank$itemPar* é assumido, para evitar selecionar itens fora do limite;

c) o argumento *StartSelect* e alguns argumentos adicionais (em *...*), podem ser consultados na função *startItems* do pacote *catR*.

O resultado da função é retornado em um objeto, como uma lista com a classe *list*, contendo os valores *items*, *par*, *thStart* e *startSelect*, os quais podem ser consultados os detalhes na função *startItems* do pacote *catR*.

3.1.5 PROCESSO DO TESTE

3.1.5.1 *stepCAT.test*

A função *stepCAT.test* administra um teste adaptativo, à partir de uma ”tabela de itens” e um ”banco de itens”, calculando a estimativa da habilidade e selecionando os próximos itens usando as funções *thetaEst*, *semTheta*, *stepCAT.stop* e *NextItem* do pacote *catR*.

O modo de usar a função, através da sua sintaxe (QUADRO 8), está definido pelos argumentos listados entre os parênteses e separados por vírgulas, considerando o elemento:

a) *items* - como o objeto *data frame* com colunas nomeadas como ”a”, ”b”, ”c” e ”d” para os parâmetros e ”r”, ”q” e ”$ch[1:n]$” para o conteúdo;

b) *itemBank* - como o objeto com o ”banco de itens” da classe *itBank*, dos valores retornados na função *createItemBank* (geralmente a partir do objeto *items*);

c) *item* - como um índice (vetor) para os itens iniciais;

d) *GUI* - como uma das opções de interface do usuário (”*console*”, ”*concerto*” ou ”*tcltk*”), para mostrar a questão e escolher uma resposta para o item;

e) *show.id* - como um valor *TRUE* (verdadeiro) para mostrar o índice do item;

f) *method* - como um método para o estimador da habilidade;

g) *criterion* - como um critério para selecionar o próximo item;

h) *rule* - como uma regra (”*length*”, ”*precision*” ou ”*classification*”) de parar o teste;

i) *thr* - como o limite para regra de parar o teste;

j) *alpha* - como o nível de significância, para o intervalo de confiança da estimativa da habilidade;

k) *out* - como o índice (vetor) de cada item previamente administrado;

l) *x* - como a resposta (vetor com valores 0 ou 1) de cada item previamente administrado;

m) *test* - como os elementos *items*, (*out*), *x*, *theta*, *SE* e *CI* (vetores) de cada item previamente administrado;

n) *...* - para informar os argumentos adicionais na função.

```
stepCAT.test(items, itemBank, item = 1, GUI = "console",
    show.id = FALSE, method = "BM", criterion = "MFI",
    rule = "length", thr = 10, alpha = 0.05, out = NULL,
    x = NULL, test = NULL, ...)
```

QUADRO 8 – SINTAXE DA FUNÇÃO *stepCAT.test*
FONTE: O autor (2013)

Os argumentos são usados na função considerando:

a) o argumento *items* na função *stepCAT.items*, como descrito na seção 3.1.2.1 nos termos da classe *data.frame*;

b) o argumento *itemBank* na função *stepCAT.bank*, como descrito na seção 3.1.3.1;

c) o argumento *item* na função *stepCAT.start*, como descrito na seção 3.1.4.1;

d) o argumento *GUI* ou alguns argumentos adicionais (em *...*) na função *stepCAT.choice*, como descrito na seção 3.1.5.3; se o argumento *show.id* não tiver o valor *TRUE* (verdadeiro) ou *FALSE* (falso), nenhum índice é mostrado;

e) o argumento *method* ou alguns argumentos adicionais (em *...*), como descrito na função *thetaEst* do pacote *catR*;

f) o argumento *criterion*, *out*, *x* ou alguns argumentos adicionais (em *...*), como descrito na função *NextItem* do pacote *catR*;

g) se o comprimento do valor do argumento *out* é igual ao comprimento do valor de *itemBank$itemPar*, o teste é interrompido para evitar que seja selecionado um item fora do limite;

h) o argumento *rule* e *thr*, como descrito na seção 3.1.5.2;

i) o argumento *test*, como descritos nos valores retornados por essa função.

O resultado da função é retornado em uma lista e cada valor atribuído em um objeto, descrito como:

a) *test* (vetor de elementos usados no processo do teste);

   – *test items* (índice de cada item);

- *test par* (matriz dos parâmetros de cada item);
- *test par "a"* (parâmetro de discriminação);
- *test par "b"* (parâmetro de dificuldade);
- *test par "c"* (parâmetro de acerto casual);
- *test par "d"* (parâmetro de distração);
- *test* x (respostas com valores 0 ou 1);
- *test theta* (estimativa da habilidade);
- *test SE* (estimativa do erro padrão);
- *test CI* (estimativa do intervalo de confiança);

3.1.5.2 *stepCAT.stop*

A função *stepCAT.stop* verifica se a regra para terminar o teste adaptativo tem sido satisfeita.

O modo de usar a função, através da sua sintaxe (QUADRO 9), está definido pelos argumentos listados entre os parênteses e separados por vírgulas, considerando o elemento:

a) *test* - como um objeto com os valores de *items*, *param*, *x*, *theta*, *SE* e *CI*, retornados pela função *stepCAT.test*;

b) *rule* - como uma regra ("*length*", "*precision*" ou "*classification*");

c) *thr* - como um limite para a regra;

d) *...* - para informar os argumentos adicionais na função.

```
stepCAT.stop(test = NULL, rule = "length", thr = 10, ...)
```

QUADRO 9 – SINTAXE DA FUNÇÃO *stepCAT.stop*
FONTE: O autor (2013)

Os argumentos são usados na função considerando:

a) a regra "*length*" retorna o valor *TRUE* (verdadeiro), se o comprimento do parâmetro *test$itens* (itens administrados) é maior ou igual ao argumento *thr* (limite), que deve assumir um número inteiro positivo ou então o valor padrão (10) é assumido;

b) a regra "*precision*" retorna o valor *TRUE* (verdadeiro), se o valor de *test$SE* (erro padrão da estimativa da habilidade) é menor ou igual ao argumento *thr* (limite);

c) a regra ”*classification* retorna o valor *TRUE* (verdadeiro), se o valor de *test$CI* (intervalo de confiança da estimativa da habilidade) não contém o argumento *thr* (limite);

d) somente se a regra é satisfeita, o valor *TRUE* (verdadeiro) é retornado.

O resultado da função é retornado em um objeto, com o valor *TRUE* (verdadeiro) ou *FALSE* (falso).

3.1.5.3 *stepCAT.choice*

A função *stepCAT.choice* mostra as alternativas para a questão e lê uma resposta do usuário.

O modo de usar a função, através da sua sintaxe (QUADRO 10), está definido pelos argumentos listados entre os parênteses e separados por vírgulas, considerando o elemento:

a) *GUI* - como um objeto e sua classe para selecionar o método;

b) *item* - como o índice do item;

c) *question* - como a descrição da questão;

d) *choices* - como descrição de cada alternativa (indexado como vetor);

e) *...* - para informar os argumentos adicionais na função;

f) *templateID* - como o identificador (ID ou nome) do modelo da página *HTML* (*template*) na plataforma *Concerto* (se aplicável ao método selecionado);

g) *templateWorkspaceID*- como o identificador (ID) do espaço de trabalho na plataforma *Concerto*, onde está localizado o objeto *templateID* (se aplicável ao método selecionado);

h) *submit* - como o texto para exibir no botão para enviar a resposta selecionada;

i) *console.force* - como um valor *TRUE* (verdadeiro), para forçar a leitura da resposta a partir do *console*, como no método *console*.

A função genérica *stepCAT.choice* possui três métodos, definidos à partir de um:

a) objeto *GUI* da classe *console*, assumindo que:

- se o argumento *item* tem valor *NULL* (nulo), não é unido ao argumento *question* para ser exibido;
- o argumento *choices* deve ser um vetor de pelo menos um elemento, ou valor padrão (5) é assumido.

```
stepCAT.choice(GUI = "console", item = NULL,
    question = NA, choices = rep(NA, 5), ...)

  ## S3 method for class 'console'
 stepCAT.choice(item = NULL,
    question = NA, choices = rep(NA, 5), ...)

  ## S3 method for class 'concerto'
 stepCAT.choice(item = NULL,
    question = NA, choices = rep(NA, 5), templateID = NULL,
    templateWorkspaceID = NULL, submit = "Submit",
    console.force = FALSE, ...)

  ## S3 method for class 'tcltk'
 stepCAT.choice(item = NULL,
    question = NA, choices = rep(NA, 5), ...)
```

QUADRO 10 – SINTAXE DA FUNÇÃO *stepCAT.choice*
FONTE: O autor (2013)

b) objeto *GUI* da classe *concerto*, assumindo que:
   - o argumento *choices* deve ser um número inteiro positivo, ou então o valor padrão (5) é assumido;
   - se o argumento *TemplateID* tem valor *NULL* (nulo), o código em *HTML* é gerado e o argumento *templateWorkspaceID* é ignorado;
   - o argumento *templateWorkspaceID* deve ser um número inteiro positivo, ou então o valor *NULL* (nulo) é assumido;
   - se o argumento *templateWorkspaceID* tem valor *NULL* (nulo), o valor da variável de sessão *concerto$workspaceID* da plataforma *Concerto* é assumido.

c) objeto *GUI* da classe *tcltk*, assumindo que:
   - se o argumento *item* tem valor *NULL* (nulo), não é unido ao argumento *question* para ser exibido;
   - o argumento *choices* deve ser um vetor de pelo menos um elemento, ou então o valor padrão (5) é assumido.

O resultado da função é retornado em um objeto com um valor numérico, contendo o índice da alternativa selecionada.

3.1.6 PASSO FINAL

3.1.6.1 *stepCAT.final*

A função *stepCAT.final* calcula as estimativas finais para o teste adaptativo.

O modo de usar a função, através da sua sintaxe (QUADRO 11), está definido pelos argumentos listados entre os parênteses e separados por vírgulas, considerando o elemento:

a) *test* - como um objeto com os valores de *items*, *param*, *x*, *theta*, *SE* e *CI*, retornados pelas função *stepCAT.test*;

b) *...* - para informar os argumentos adicionais na função.

```
stepCAT.final(test, ...)
```

QUADRO 11 – SINTAXE DA FUNÇÃO *stepCAT.final*
FONTE: O autor (2013)

A versão atual da função *stepCAT.final* somente devolve o objeto, como informado no argumento *test*.

3.1.7 EXEMPLOS

Os trechos em código *R* no QUADRO 12, QUADRO 13 e QUADRO 14 são usados como exemplos para demonstrar como construir e aplicar um teste adaptativo, seguindo as etapas e os argumentos de cada função operada internamente.

O *R* usa a interatividade com o usuário para iniciar e processar o teste adaptativo através de uma das opções de interfaces, como o *console* (FIGURA 5), o mecanismo *Tcl/Tk* (FIGURA 6) e a plataforma *Concerto*. No passo final os valores são atribuídos no objeto com o resultado, então podem ser reutilizados em outras funções ou mesmo como argumentos para um novo teste. (FIGURA 7).

```
# stepCAT.by Exemplo 1
# Construindo e aplicando um teste adaptativo interagindo em Console

# carregando pacote
library(stepCAT)

# atribuindo local da tabela em objeto como classe 'character'
table <- "/Users/Henrique/Documents/GitHub/stepCAT/data/futebol.csv"

# demonstrando etapas e argumentos na funcao usada internamente
#   tabela de itens    stepCAT.table  table, id = TRUE
#   passo inicial      stepCAT.start  nrItems = 3, halfRange = 1
#   processo do teste  stepCAT.test   rule = "length", thr = 7
#   passo final        stepCAT.final

# usando argumentos na funcao stepCAT.by e armazenando resultado em objeto
result <- stepCAT.by(table, id = TRUE, nrItems = 3, halfRange = 1, rule = "
    length", thr = 7)
```

QUADRO 12 – CÓDIGO *R* DO EXEMPLO 1 DA FUNÇÃO *stepCAT.by*
FONTE: O autor (2013)

```
# stepCAT.by Exemplo 2
# Construindo e aplicando um teste adaptativo interagindo em Tcl/Tk

# carregando pacote
library(stepCAT)

# atribuindo local da tabela e lendo em objeto como classe 'data.frame'
table.file <- "/Users/Henrique/Documents/GitHub/stepCAT/data/futebol.csv"
table <- read.table(table.file, header = TRUE, sep = ";", encoding = "UTF-8")

# demonstrando etapas e argumentos na funcao usada internamente
#   banco de itens     stepCAT.items  table, id = TRUE
#   passo inicial      stepCAT.start  nrItems = 3, halfRange = 1
#   processo do teste  stepCAT.test   GUI = "tcltk", rule = "length", thr = 7
#   passo final        stepCAT.final

# usando argumentos na funcao stepCAT.by e armazenando resultado em objeto
result <- stepCAT.by(table, id = TRUE, nrItems = 3, halfRange = 1, GUI = "
    tcltk", rule = "length", thr = 7)
```

QUADRO 13 – CÓDIGO *R* DO EXEMPLO 2 DA FUNÇÃO *stepCAT.by*
FONTE: O autor (2013)

```
# stepCAT.by Exemplo 3
# Construindo um teste adaptativo básico simulando o uso do Concerto

# Carregando o código-fonte
library(RCurl)
url <- "https://raw.github.com/hkrefer/stepCAT/master/R/stepCAT.R"
code <- textConnection(getURL(url, followlocation = TRUE, ssl.verifypeer =
    FALSE))
source(code)

# Carregando o local da tabela em objeto como classe 'character'
table <- "/Users/Henrique/Documents/GitHub/stepCAT/data/futebol.csv"

# Definindo e passando os argumentos nas funções
# stepCAT.table  <- table, id = TRUE, file.force = TRUE
# stepCAT.start  <- nrItems = 3, theta = 0, halfRange = 1
# stepCAT.test   <- GUI = "concerto", rule = "length", thr = 7
# stepCAT.choice <- console.force = TRUE
result <- stepCAT.by(table, id = TRUE, file.force = TRUE, nrItems = 3,
    halfRange = 1, GUI = "concerto", rule = "length", thr = 7, console.force =
     TRUE)
```

QUADRO 14 – CÓDIGO *R* DO EXEMPLO 3 DA FUNÇÃO *stepCAT.by*
FONTE: O autor (2013)

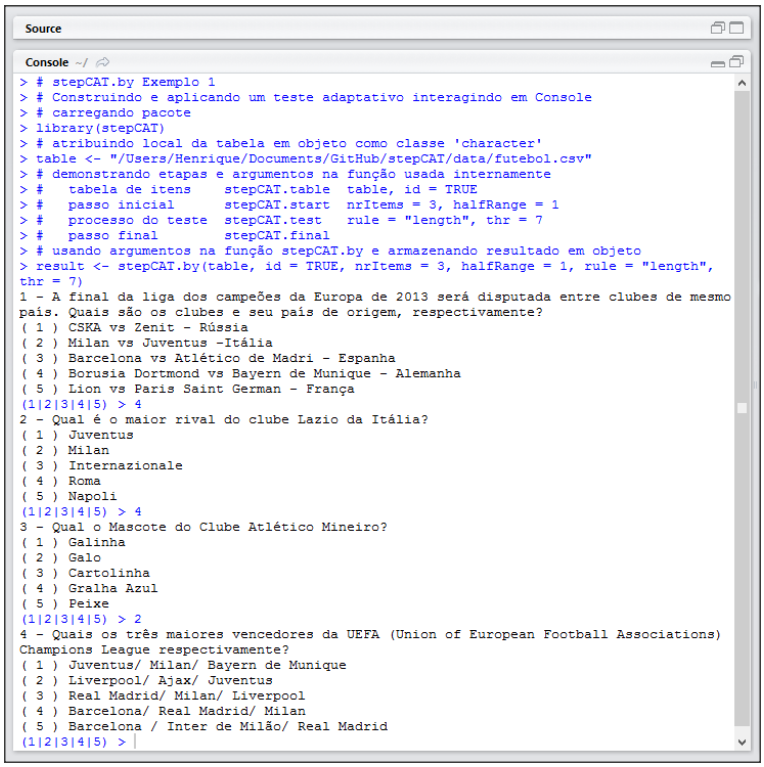

FIGURA 5 – INTERATIVIDADE COM O USUÁRIO PARA O EXEMPLO 1

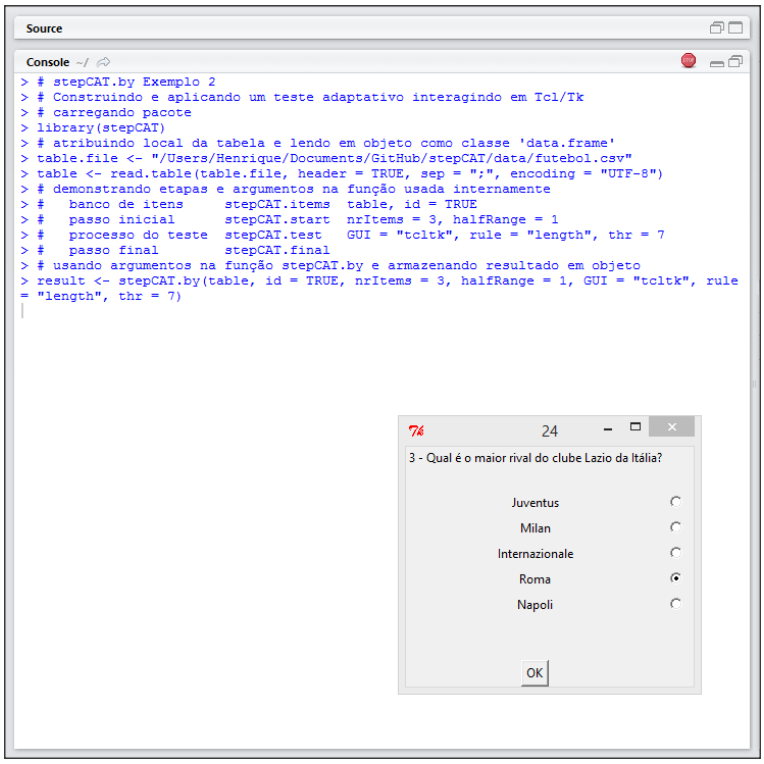

FIGURA 6 – INTERATIVIDADE COM O USUÁRIO PARA O EXEMPLO 2

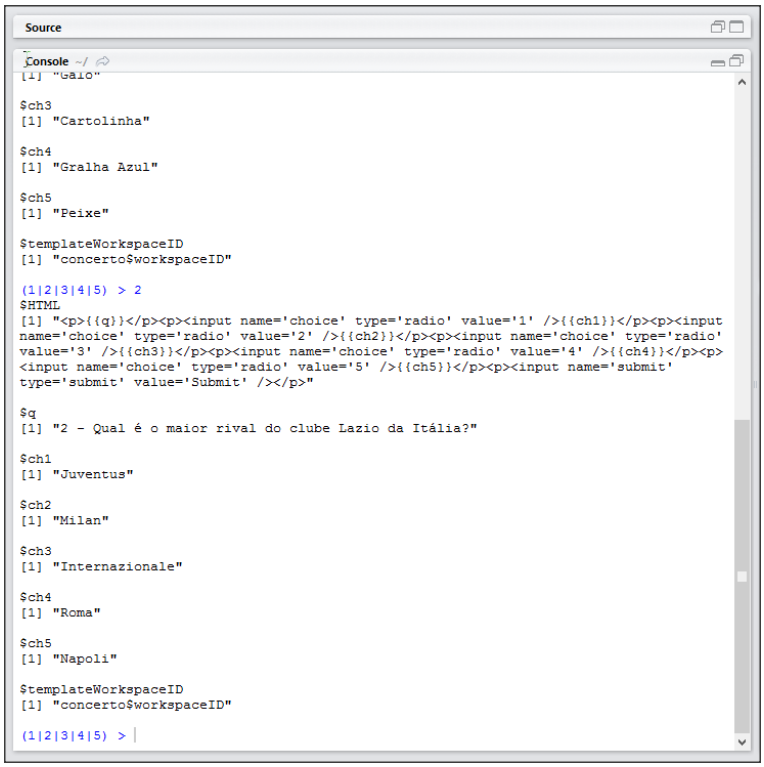

FIGURA 7 – INTERATIVIDADE COM O USUÁRIO PARA O EXEMPLO 3

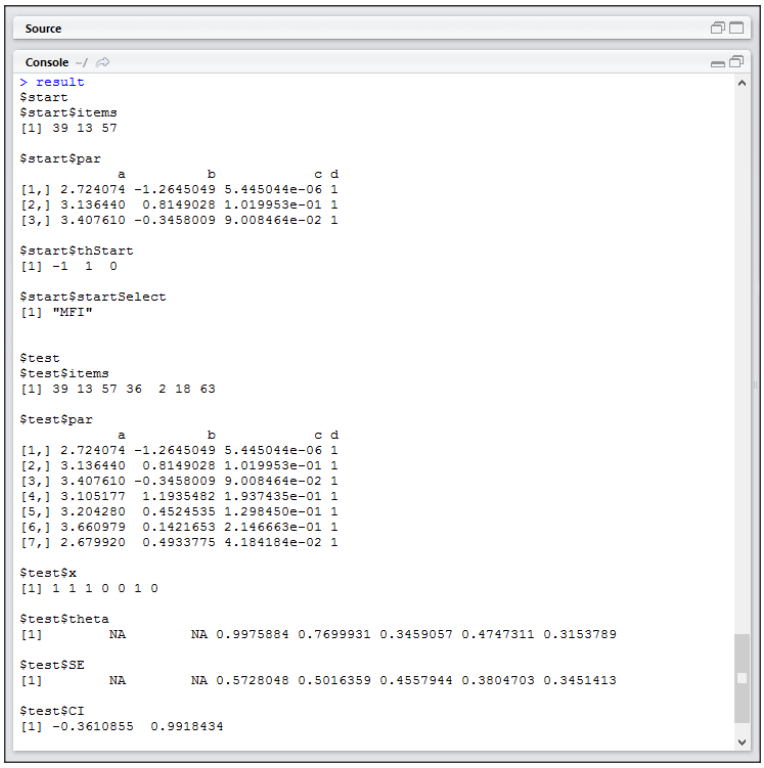

FIGURA 8 – RESULTADO DA FUNÇÃO *stepCAT* ATRIBUÍDO AO OBJETO

3.2 O PACOTE *stepCAT* E A PLATAFORMA *CONCERTO*

O *Concerto* é denominado pelos seus criadores como uma plataforma de teste. Conceitualmente é desenvolvida para integrar o mecanismo do *R*, um gerenciador de banco de dados e as capacidades de apresentar usando o *HTML* algum resultado *on-line* na *web*.

Colaborativamente, com um conceito similar, o pacote *stepCAT* para o uso no *R* provém funcionalidades para flexibilizar o uso na plataforma *Concerto*.

No contexto do trabalho, é apresentado em cada uma das seções seguintes, em um formato ilustrado e simplificado de um passo-a-passo, o modo de usar o espaço de trabalho do *Concerto* com as funcionalidades do pacote *stepCAT* apresetando em detalhes nas seções anteriores, demonstrando como viabilizar a construção e administração de um teste adaptativo.

3.2.1 ACESSAR O *CONCERTO*

a) abrir o navegador de internet – *browser*;

b) acessar o endereço: *http://concerto4.e-psychometrics.com/cms/* (FIGURA 9)

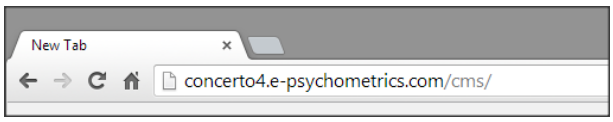

FIGURA 9 – TELA DO *BROWSER* PARA USAR O CONCERTO

3.2.2 CRIAR UMA CONTA DE USUÁRIO – *ACCOUNT*

a) clicar em ***create an account***, para registrar uma conta de usuário (FIGURA 10);

b) preencher os espaços (FIGURA 11) como:

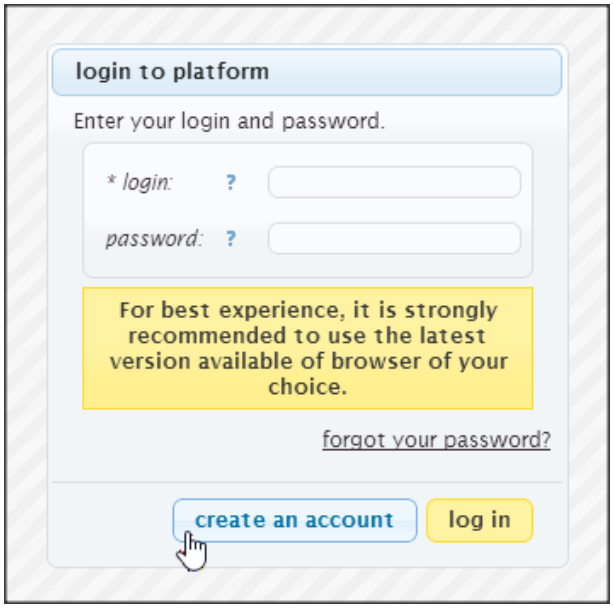

FIGURA 10 – TELA DO *CONCERTO* EM *CREATE AN ACCOUNT*

- *login* - identificador do usuário;
- *password* - senha do usuário;
- *pass. conf.* - repetir a senha do usuário;
- *first name* - nome do usuário;
- *last name* - sobrenome do usuário;
- *inst. type* - tipo da instituição do usuário;
- *inst. name* - nome da instituição do usuário;
- *email* - endereço eletrônico do usuário;

c) clicar em ***register***.

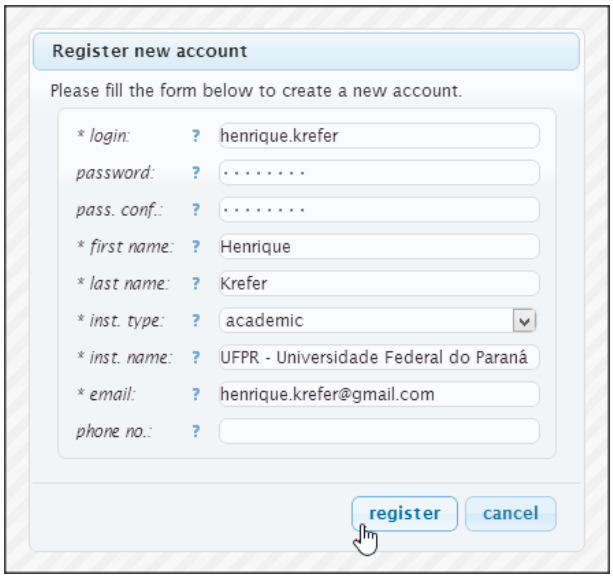

FIGURA 11 – TELA DO *CONCERTO* EM *REGISTER NEW ACCOUNT*

d) clicar em **OK**.

3.2.3 INICIAR UM ESPAÇO DE TRABALHO – *WORKSPACE*

a) acessar o *Concerto* (como na seção 3.2.1);
b) preencher os espaços (FIGURA 12) como:
   – *login* - identificador do usuário;
   – *password* - senha do usuário;
c) clicar em **log in**.

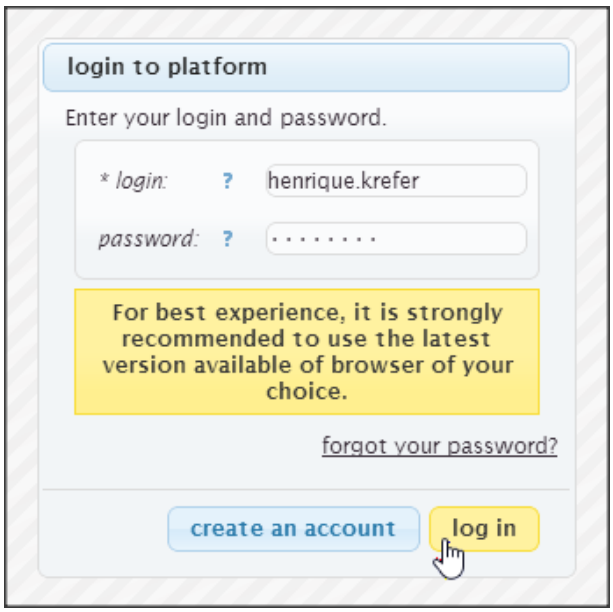

FIGURA 12 – TELA DO *CONCERTO* EM *LOGIN*

3.2.4 CRIAR E IMPORTAR UMA TABELA DE ITENS – *TABLES*

a) acessar o *Concerto* (como na seção 3.2.1);
b) na barra superior do **current workspace** clicar em ***tables*** (FIGURA 13);
c) na aba interna ***adding new object available objects*** clicar em ***+ add new object*** (FIGURA 14);

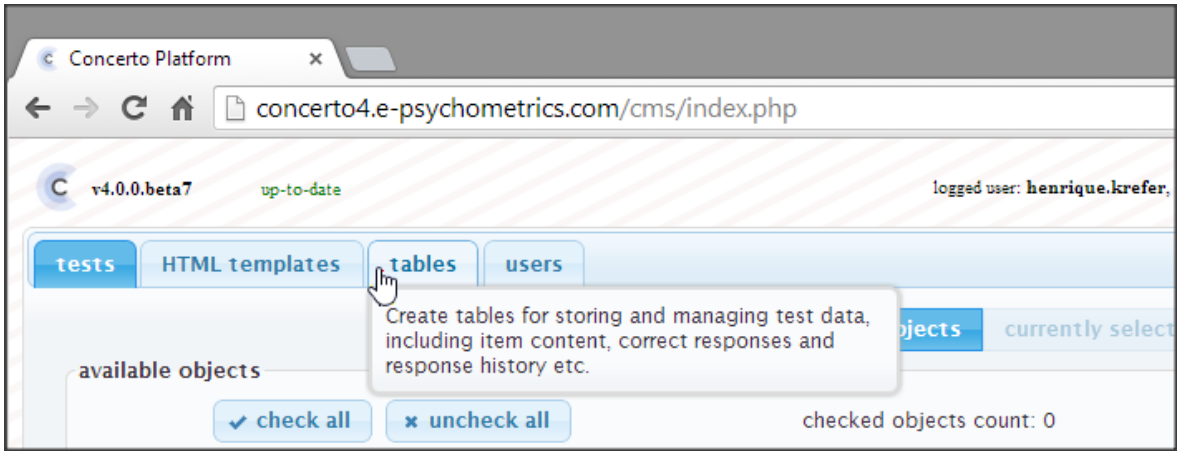

FIGURA 13 – TELA DO *CONCERTO* EM *CREATE A TABLE*

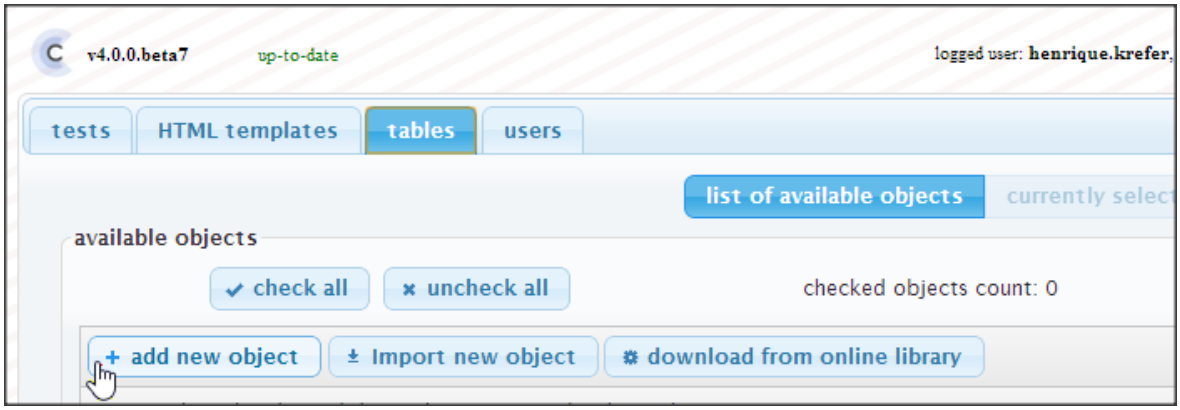

FIGURA 14 – TELA DO *CONCERTO* EM *ADD NEW TABLE*

d) na janela ***adding new object*** na guia ***creating new table*** preencher *name* - nome da tabela;
   - no exemplo é utilizado o nome da tabela como *exemplo_1*;

e) clicar em ***Save***;

f) na janela ***Saving object*** clicar em ***ok***;

g) na barra inferior do *Concerto* clicar em ***import table from CSV file*** (FIGURA 15);

FIGURA 15 – TELA DO *CONCERTO* EM *IMPORT NEW TABLE*

h) na janela ***Importing table*** clicar em ***Yes***;

i) na janela ***Importing table from CSV file*** (FIGURA 16) considerar:
   - marcar ***add 'id' column*** - para importar a coluna nomeada como *id*;
   - marcar ***header row*** - para importar a primeira linha como os nomes para as colunas;

- escolher ***field delimiter*** como ; - para ponto-e-vírgula como o separador de colunas;

j) clicar em ***file*** para selecionar o arquivo:

- no exemplo é utilizado o arquivo ”futebol.csv” disponível em *https://raw.github.com/hkrefer/stepCAT/master/data/futebol.csv*;

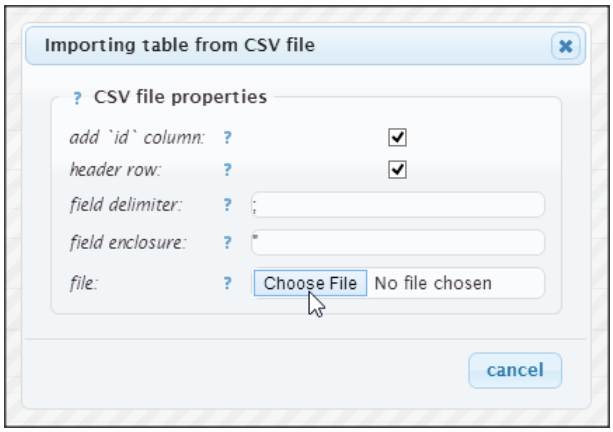

FIGURA 16 – TELA DO *CONCERTO* EM *IMPORT NEW TABLE PROPERTIES*

k) na janela ***File uploaded*** clicar em ***Yes***;

l) na janela ***Importing table*** clicar em ***Ok***;

m) na barra inferior do ***Concerto*** clicar ***save***;

n) na janela ***Saving object*** clicar em ***ok***.

3.2.5 CRIAR UM TESTE COM O *stepCAT – TESTS*

a) acessar o *Concerto* (como na seção 3.2.1);

b) na barra superior do ***Current workspace*** clicar em ***tests*** (FIGURA 17);

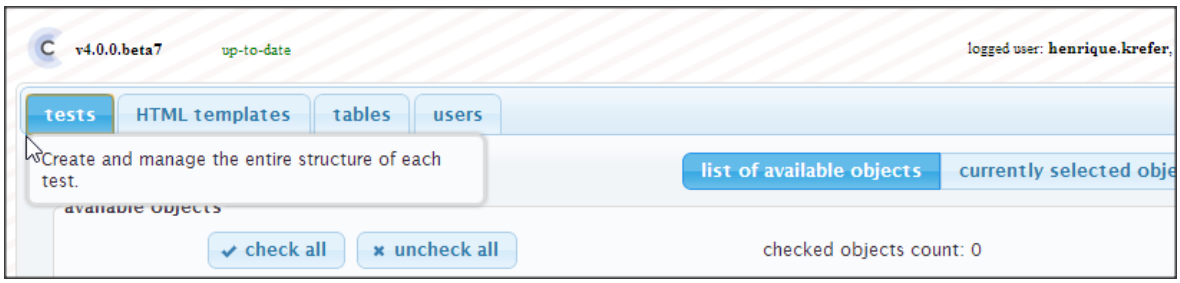

FIGURA 17 – TELA DO *CONCERTO* EM *CREATE A TEST*

c) na aba interna ***adding new object available objects*** clicar em ***+ add new object*** (FIGURA 18);

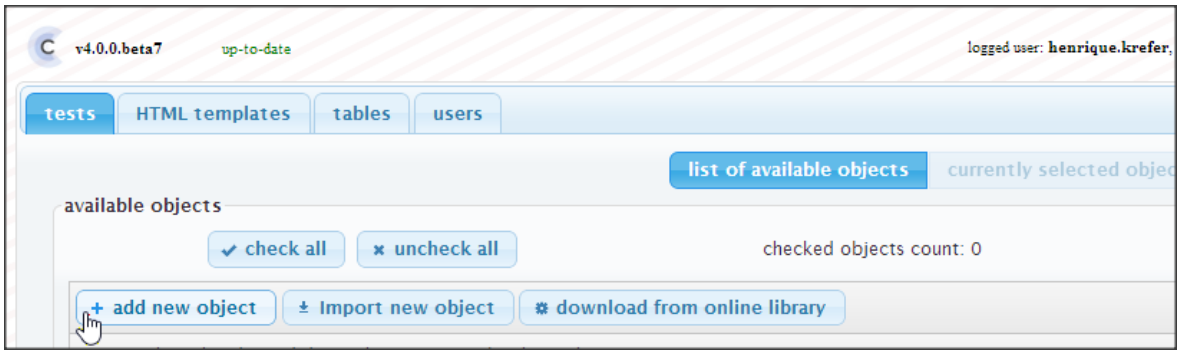

FIGURA 18 – TELA DO *CONCERTO* EM *ADD NEW TEST*

d) na janela ***adding new object*** na guia ***creating new test*** preencher *name* – nome do teste;

e) na janela ***Saving object*** clicar em ***Ok***;

f) inserir o código R como no exemplo do QUADRO 15 na guia ***Test logic*** (FIGURA 19);

```
# stepCAT source
library(RCurl)
url <- "https://dl.dropboxusercontent.com/s/eoypbii7tdcm5qj/stepCAT.R"
code <- textConnection(getURL(url, followlocation = TRUE, ssl.verifypeer =
    FALSE))
source(code)

# stepCAT arguments
result <- stepCAT.by(table = "exemplo_1", id = TRUE, DB = "concerto", GUI =
"concerto", nrItems = 3, rule = "length", thr = 10)
```

QUADRO 15 – CÓDIGO *R* PARA *ADD NEW TEST* DO *CONCERTO*
FONTE: O autor (2013)

g) na guia ***test session state*** clicar em ***begin*** para iniciar uma sessão para validar o teste:

   – observe que abrirá uma nova aba ou tela no *browser* (FIGURA 20);

h) na aba ou tela anterior ***Concerto Platform*** do *browser* uma sessão do teste é mostrada (FIGURA 21);

i) na guia ***test session state*** clicar em ***stop*** para interromper o teste;

j) na barra inferior do *Concerto* clicar em ***save*** (FIGURA 22);

k) na janela ***Saving object*** clicar em ***Ok***.

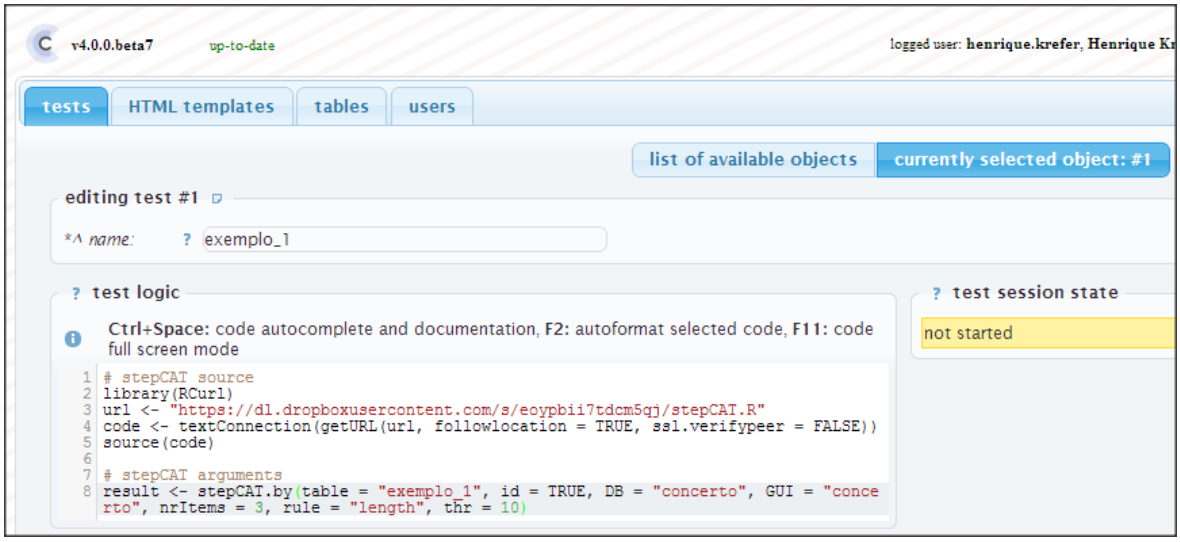

FIGURA 19 – TELA DO *CONCERTO* EM *NEW TEST LOGIC*

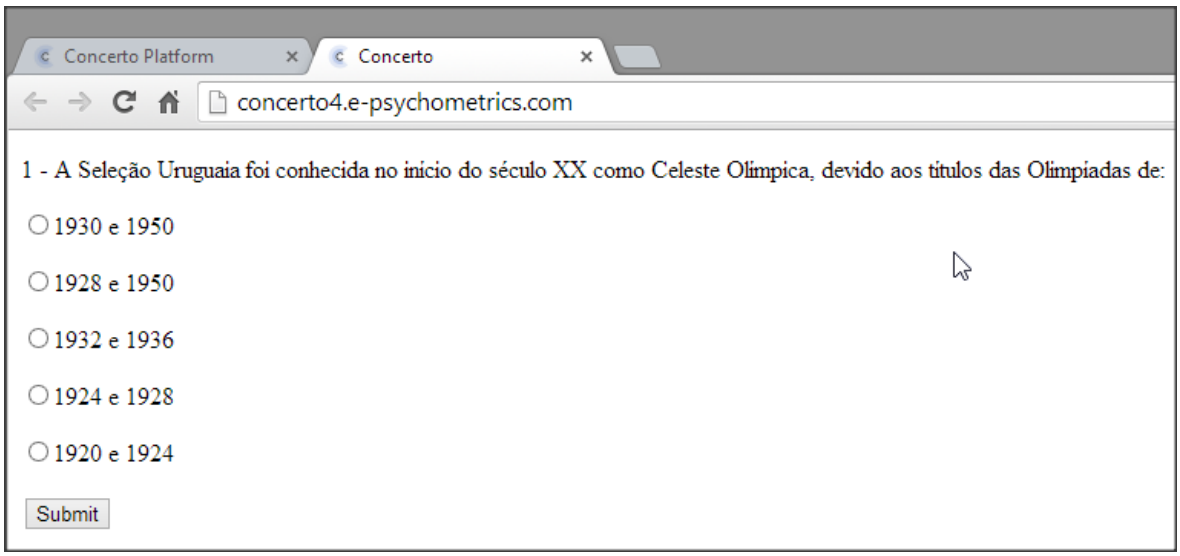

FIGURA 20 – TELA DO *CONCERTO* EM *NEW TEST SCREEN*

3.2.6 EDITAR E EXECUTAR UM TESTE – *TESTS*

a) acessar o *Concerto* (como na seção 3.2.1);

b) na barra superior do ***current workspace*** clicar em ***tests*** (FIGURA 23);

c) na aba interna ***available objects*** clicar em ***edit object*** na linha do teste desejado (FIGURA 24);

d) na guia ***test logic*** editar o código *R* como no exemplo do QUADRO 16;

e) na barra inferior do *Concerto* clicar em ***save*** (FIGURA 25);

f) na janela ***Saving object*** clicar em ***ok***;

g) na barra inferior do *Concerto* clicar em ***run test*** (FIGURA 26):

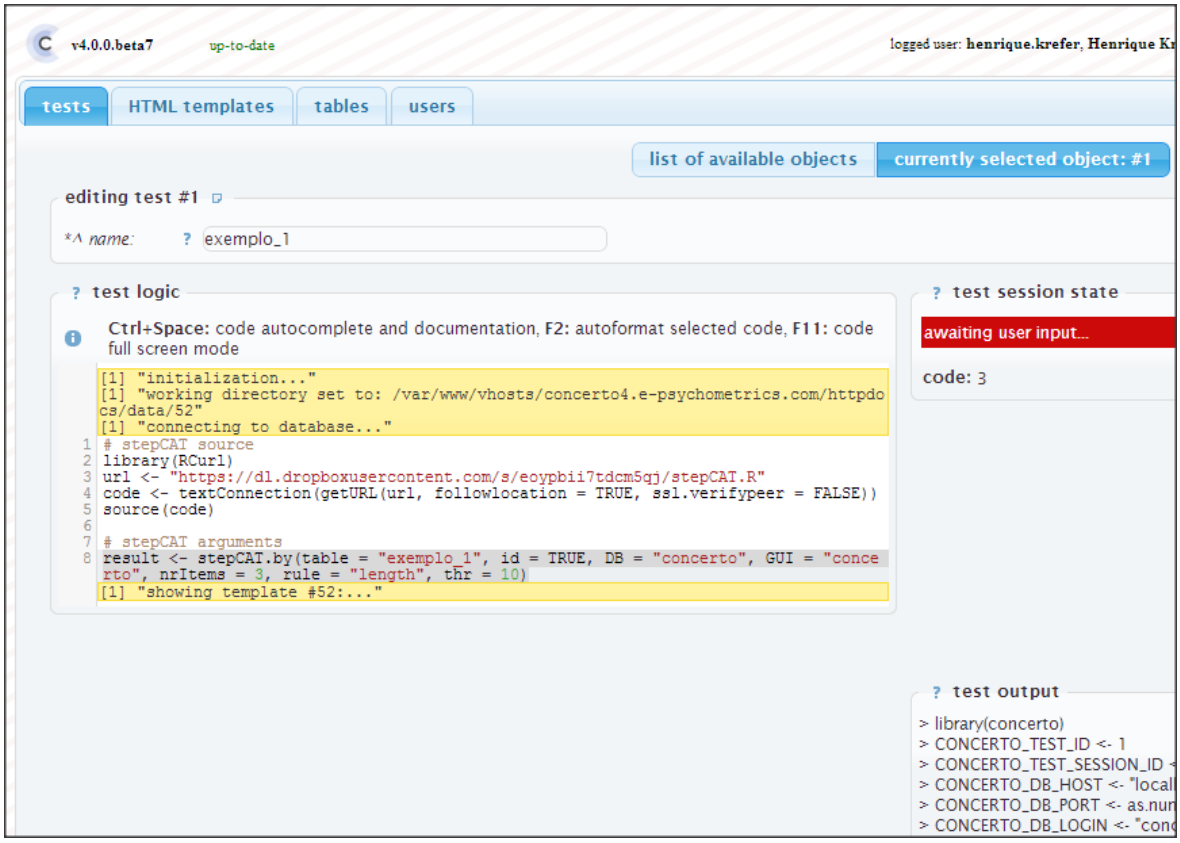

FIGURA 21 – TELA DO *CONCERTO* EM *NEW TEST SESSION*

FIGURA 22 – TELA DO *CONCERTO* EM *NEW TEST SAVE*

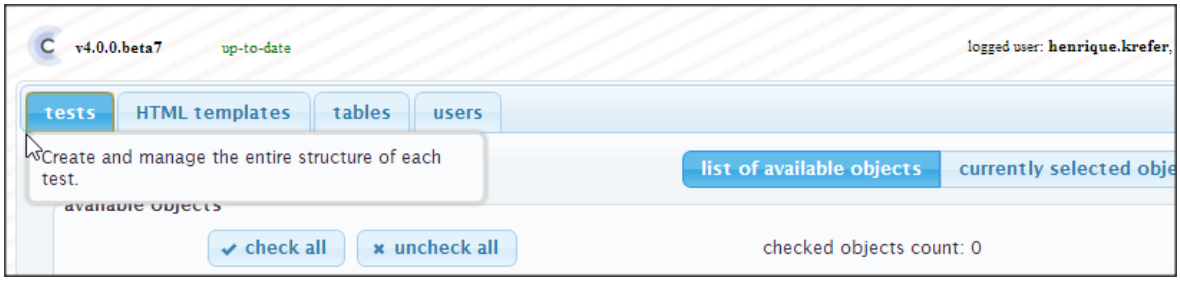

FIGURA 23 – TELA DO *CONCERTO* EM *CREATE A NEW TEST*

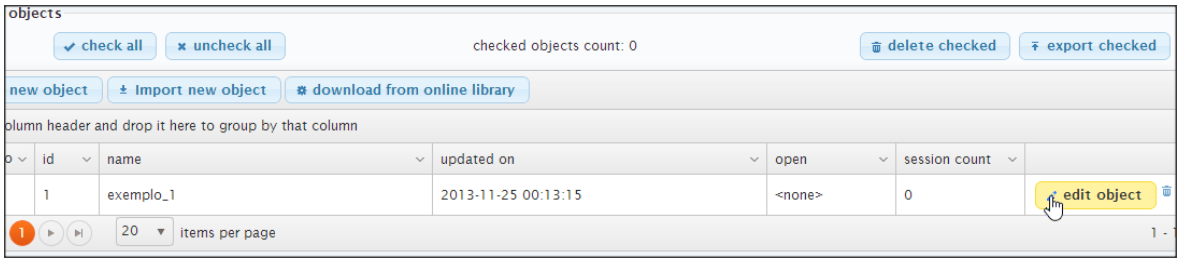

FIGURA 24 – TELA DO *CONCERTO* EM *EDIT A TEST*

```
# stepCAT source
library(RCurl)
url <- "https://raw.github.com/hkrefer/stepCAT/master/R/stepCAT.R"
code <- textConnection(getURL(url, followlocation = TRUE, ssl.verifypeer =
    FALSE))
source(code)

# stepCAT arguments
result <- stepCAT.by(table = "exemplo_1", id = TRUE, DB = "concerto", GUI =
"concerto", nrItems = 3, halfRange = 1, rule = "length", thr = 10)

concerto.template.show(
HTML="theta = {{theta}}", params=list(theta = tail(result$test$theta, 1)),
finalize=TRUE,
)
```

QUADRO 16 – CÓDIGO *R* PARA *EDIT A TEST* DO *CONCERTO*
FONTE: O autor (2013)

FIGURA 25 – TELA DO *CONCERTO* EM *NEW TEST SAVE*

– observe que abrirá uma nova aba ou tela no *browser*;

FIGURA 26 – TELA DO *CONCERTO* EM *RUN A TEST*

h) na nova aba ou tela no *browser* clique em uma escolha e em *Submit* enquanto não finalizar o teste (FIGURA 27):;

FIGURA 27 – TELA DO *CONCERTO* EM *RUN A TEST SCREEN*

– ao final do teste uma nova aba ou tela no *browser* será exibida (FIGURA 28).

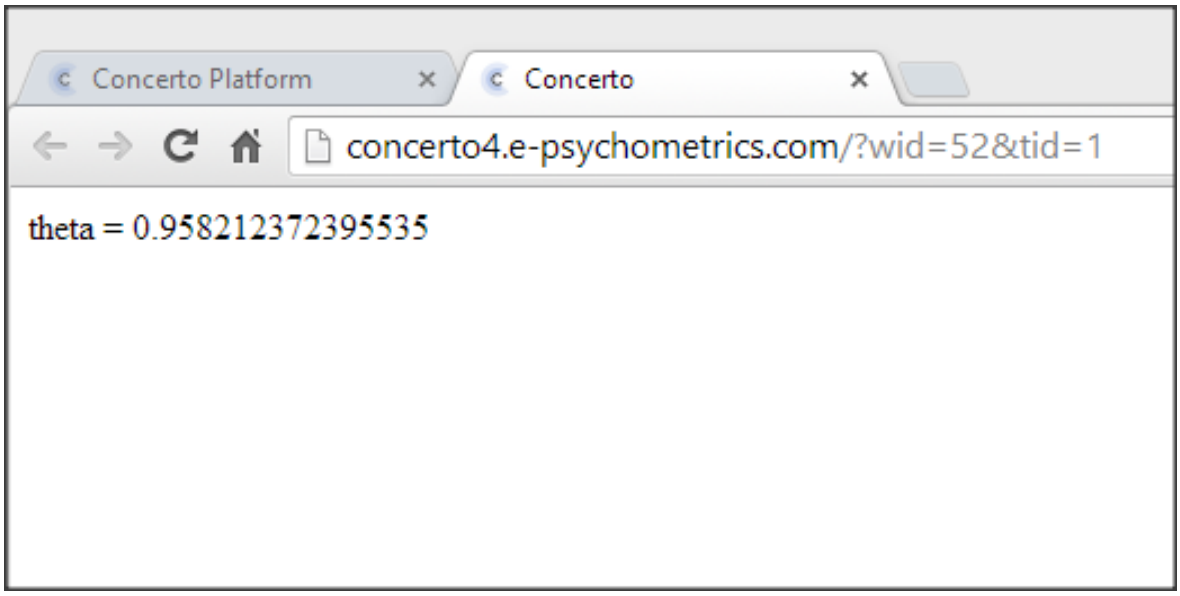

FIGURA 28 – TELA DO *CONCERTO* EM *SCREEN RESULT*

3.2.7 EXECUTAR UM TESTE NO *BROWSER*

a) abrir o navegador de internet - *browser*;

b) acessar o endereço:
http://concerto4.e-psychometrics.com/?wid=52tid=1

– observar que o campo *wid* é o *Workspace ID* e o *tid* é o *ID* do teste criado anteriormente (FIGURA 29).

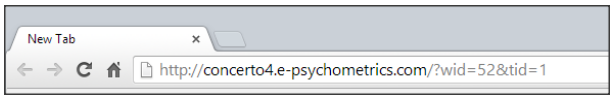

FIGURA 29 – TELA DO *BROWSER* PARA TESTAR NO CONCERTO

4 CONSIDERAÇÕES FINAIS

O desafio inicial estava em compreender e elaborar o fluxo do processo em um TAC, como isso deveria ser tratado e quais seriam as possibilidades para sua aplicação e apresentação. O *Concerto*, principal recurso tecnológico elencado, realmente pode ser considerado uma plataforma com essa finalidade, considerando a sua robustez na integração entre o mecanismo do *R*, um gerenciador de banco de dados e a capacidade de apresentação do *HTML* na *web*, porém não é concebido para fazer uma simples transição da teoria para a prática em um TAC. Preencher essa lacuna seria preciso.

Para isso a solução criada com o desenvolvimento, construção e publicação do pacote *stepCAT* para o uso no *R* atende de forma consideravelmente ampla a aplicação de um TAC, partindo de um algoritmo que permite desde a conexão com os dados, esssências no processo, a apresentação em outras interfaces amigáveis para o usuário como o *R Console* e o *Tcl/Tk* e, por fim, a incorporação completa ao *Concerto*.

O resultado do trabalho pode ainda servir de referência teórica para o TAC e as ferramentas utilizadas, como os softwares, o mecanismo principal do pacote *catR* e os pacotes adicionais como o *roxygen2*.

Com uma análise futura pode ser feita a implementação de novas funcionalidades no pacote, como métodos nativos para a conexão com outro gerenciador de banco de dados e outra interface, codificação da resposta armazenada em arquivo, inclusão de item não-calibrado no meio do teste afim de obter novo item calibrado, análise adicional do resultado com uma apresentação especializada entre outros recursos que podem ser implementados colaborativamente, enriquecendo o processo do TAC e consolidando a sua aplicação com base em métodos estatísticos.

REFERÊNCIAS

ANDRADE, D. F. et al. Teoria da Resposta ao Item: Conceitos e Aplicações. In: **Simpósio Nacional de Probabilidade e Estatística - SINAPE**. [S.l.]: Associação Brasileira de Estatística - ABE, 2000. 16, 17

BAKER, F. B. **The Basics of Item Response Theory**. 2nd.. ed. Wisconsin: ERIC Clearinghouse on Assessment and Evaluation, 2001. ISBN 1886047030. 16

CONCERTO TESTING PLATFORM. **Concerto Testing Platform: Open-source Online R-based Adaptive Testing Platform**. 2012. Disponível em: ⟨http://www.psychometrics.cam.ac.uk/newconcerto⟩. Acesso em: 09/09/13. 12, 15

DE KLERK, G. **Classical Test Theory (CTT)**. 2008. Disponível em: ⟨http://www.intestcom.org/Publications/ORTA/Classical+test+theory.php⟩. Acesso em: 08/09/13. 10

FORBELLONE, A. A.; INÁCIO, L. d. V. Construção e validação de um instrumento de mensuração para avaliar o conhecimento sobre futebol utilizando a teoria da resposta ao item. 2013. 13, 19

LEISCH, F. Creating R Packages : A Tutorial. **Compstat 2008-Proceedings in Computational Statistics**, n. 36, p. 1–19, 2009. Disponível em: ⟨http://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf⟩. Acesso em: 13/11/13. 14

MAGIS, D.; GILLES, R. Random Generation of Response Patterns under Computerized Adaptive Testing with the R Package catR. **Journal of Statistical Software**, v. 48, n. 8, 2012. Disponível em: ⟨http://www.jstatsoft.org/v48/i08/⟩. Acesso em: 09/09/13. 12, 14, 21

MAGIS, D.; GILLES, R. **Package 'catR'**. 2013. Disponível em: ⟨http://cran.r-project.org/web/packages/catR/catR.pdf⟩. Acesso em: 09/09/13. 23

MOREIRA JUNIOR, F. d. J. **Sistemática para a implantação de testes adaptativos informatizados baseados na teoria da resposta ao item**. 334 p. Tese (Doutorado) — Universidade Federal de Santa Catarina - Centro Tecnológico, 2011. 12, 18

R DEVELOPMENT CORE TEAM. **R: A Language and Environment for Statistical Computing**. Vienna, Austria.: R Foundation for Statistical Computing, 2011. 409 p. (R Foundation for Statistical Computing, 2.11.1). Disponível em: ⟨http://www.r-project.org⟩. Acesso em: 09/09/13. 14

RSTUDIO. **RStudio: Integrated development environment for R**. Boston, MA.: [s.n.], 2013. 14

RUDNER, L. M. **Computer Adaptive Testing Tutorial**. 1998. Disponível em: ⟨http://echo.edres.org:8080/scripts/cat/catdemo.htm⟩. Acesso em: 09/09/13. 18

THISSEN, D.; WAINER, H. **Test Scoring**. Psychology Press, 2001. ISBN 9781410604729. Disponível em: ⟨http://books.google.com.br/books?id=G1I_Ygj_-1wC⟩. Acesso em: 09/09/13. 10

THOMPSON, N. A.; WEISS, D. J. A Framework for the Development of Computerized Adaptive Tests. **Practical Assessment, Research & Evaluation (PARE)**, v. 16, n. 1, 2011. Disponível em: ⟨http://pareonline.net/pdf/v16n1.pdf⟩. Acesso em: 09/09/13. 11, 17, 19

VAN DER LINDEN, W. J.; HAMBLETON, R. K. **Handbook of Modern Item Response Theory**. Springer, 1997. ISBN 9780387946610. Disponível em: ⟨http://books.google.com.br/books?id=aytUuwl4ku0C⟩. Acesso em: 12/09/13. 11

VENDRAMINI, C. M.; SILVA, M. C. Análise de itens de uma prova de raciocínio estatístico. **Psicologia em Estudo**, v. 9, n. 3, p. 487–498, 2004. Disponível em: ⟨http://www.scielo.br/pdf/pe/v9n3/v9n3a16.pdf⟩. Acesso em: 12/09/13. 16

WAINER, H.; DORANS, N. J. **Computerized Adaptive Testing: A Primer**. [S.l.]: Lawrence Erlbaum Associates, 2000. ISBN 9780805835113. 18

WEISS, D. J. Adaptive testing by computer. **Journal of Consulting and Clinical Psychology**, US: American Psychological Association, v. 53, n. 6, p. 774–789, 1985. ISSN 1939-2117(Electronic);0022-006X(Print). Disponível em: ⟨http://psycnet.apa.org/journals/ccp/53/6/774/⟩. Acesso em: 12/09/13. 18

WEISS, D. J. Item Banking , Test Development , and Test Delivery. In: GEISINGER, K. F. (Ed.). **APA Handbook of Testing and Assessment in Psychology**. Minneapolis MN: American Psychological Association, 2011. cap. 1. 11, 18

WICKHAM, H. et al. **Package 'roxygen2'**. 2013. Disponível em: ⟨http://cran.r-project.org/web/packages/roxygen2/roxygen2.pdf⟩. Acesso em: 20/10/13. 15

APÊNDICE
INSCRIÇÃO DO TRABALHO NO CONBRATRI

Belém, 08 de Novembro de 2013

CARTA DE ACEITE

Prezado(s) Senhor(es),
Henrique Krefer, Rhuan Gabriel C. de Lima, Adilson dos Anjos, Bruno Henrique Correa da Silva

É com satisfação que comunicamos a aceitação de seu trabalho intitulado **"stepCAT: uma proposta de pacote para implementação de testes adaptativos computadorizados com uso do R"**, para apresentação no III Congresso Brasileiro de Teoria da Resposta ao Item, a ocorrer em Belém-PA, dias 04 a 06/12, nas dependências do Hotel Hilton.

Informamos ainda que a apresentação será feita em **Comunicação POSTER.** Todos os trabalhos enviados para fins de Concurso de IC ou Concurso de Dissertação, tendo sido aceitos, serão obrigatoriamente encaminhados à Comunicação Oral.

Caso o apresentador ainda não tenha feito sua inscrição no III CONBRATRI, solicitamos que a faça com a maior brevidade.

Atenciosamente,

Prof. Dr. Héliton Ribeiro Tavares
Coordenador