Abstract
Computerized adaptive tests are tests in which items are selected and presented to examinees in real time, according to proficiency estimations taken after each new item is answered. Assisted by the mathematical models of Item Response Theory, items and examinees can be represented by a set of parameters, which in turn allows for the execution of test simulations, providing means to analyze different characteristics of an adaptive test’s life cycle. This paper presents catsim, a package written in the Python language, the first package in this language specialized in computerized adaptive tests and the logistical models of Item Response Theory. catsim provides functions for generating item and examinees parameters, simulating tests and plotting results, as well as enabling end users to create new procedures for proficiency initialization, item selection, proficiency estimation and test stopping criteria. The simulator keeps a record of the items selected for each examinee as well as their answers and also enables the simulation of linear tests, in which all examinees answer the same items.

Keywords: computerized adaptive testing, item response theory, Python.

1. Introduction
Assessment instruments are widely used to measure individuals latent traits, that is, internal characteristics that cannot be directly measured. An example of such assessment instruments are educational and psychological tests. Each test is composed of a series of items and an examinee’s answers to these items allow for the measurement of one or more of his or hers latent traits. When a latent trait is expressed in numerical form, it is called an ability or proficiency.

Ordinary tests, hereon called linear tests, are applied using the orthodox paper and pencil strategy, in which tests are printed and all examinees are presented with the same items in the same order. One of the drawbacks of this methodology is that individuals with either high or low proficiencies must answer all items in order to have their proficiency estimated. An individual with high proficiency might get bored of answering the whole test if it only contains items that he or she considers easy; on the other hand, an individual of low proficiency might get frustrated if he is confronted by items considered hard and might give up on the test or answer the items without paying attention.

With these concerns in mind, a new paradigm in assessment emerged in the 70s. Initially named tailored testing by Lord, Novick, and Birnbaum (1968), these were tests in which items were chosen to be presented to the examinee in real time, based on the examinee’s responses.
to previous items. The name was changed to computerized adaptive testing (CAT) due to the advances in technology that facilitated the application of such a testing methodology using electronic devices, like computers and tablets.

In a CAT, the examinee’s proficiency is evaluated after the response of each item. The new proficiency is then used to select a new item, theoretically closer to the examinee’s real proficiency. This method of test application has several advantages compared to the traditional paper-and-pencil method, since examinees of high proficiency are not required to answer all the low difficulty items in a test, answering only the items that actually give some information regarding his or hers true knowledge of the subject at matter. A similar, but inverse effect happens for those examinees of low proficiency level.

Finally, the advent of CAT allowed for researchers to create their own variant ways of starting a test, choosing items, estimating proficiencies and stopping the test. Fortunately, the mathematical formalization provided by Item Response Theory (IRT) allows for tests to be computationally simulated and the different methodologies of applying a CAT to be compared under different constraints. Packages with these functionalities already exist in the R language (Magis and Raïche 2012) but not yet in Python, to the best of the authors’ knowledge. catsim was created to fill this gap, using the facilities of established scientific packages such as numpy and scipy, as well as the object-oriented programming paradigm supported by Python to create a simple, comprehensive and user-extendable CAT simulation package.

2. Related Work

The work on catsim is comparable to similar open-source CAT simulation packages such as catR (Magis and Raïche 2012), catIrt (Nydick 2015) and MAT (Choi and King 2015). While catsim does not yet implement Bayesian examinee proficiency estimation (like the three aforementioned packages) and simulation of tests under Multidimensional Item Response Theory (like MAT), it contributes with a range of item selection methods from the literature, as well as by taking a broader approach by providing users with IRT-related functions, such as bias, item and test information, as well as functions that assist plotting item curves and test results. Furthermore, catsim is the only package of its kind developed in the Python language, to the best of the authors’ knowledge.

Similarly, another open-source package that works with adaptive testing is mirtCAT (Chalmers 2016), which provides users an interface that enables the creation of both adaptive and non-adaptive tests. While catsim’s main purpose is not the construction of tests, it allows users to simulate tests under different circumstances and anticipate possible issues before creating a full-scale test.

CATSim, a homonymous program by Assessment Systems1, is a commercial CAT analysis tool created with the purpose of assisting in the development of adaptive testing. While it provides further constraints on item selection as well as the inclusion of polytomous models, it is supposed to be used as a standalone application and not as an extensible software library and thus does not allow users to create their own item selection methods (constrained or otherwise) as well as integrate with other software as catsim allows.

1http://www.assess.com/catsim
3. Item Response Theory

As a CAT simulator, catsim borrows many concepts from Item Response Theory (IRT) (Lord et al. 1968; Rasch 1966), a series of models created in the second part of the 20th century with the goal of measuring latent traits. catsim makes use of Item Response Theory one-, two-, three- and four-parameter logistic models, in which examinees and items are represented by a set of numerical values (the models’ parameters). Item Response Theory itself was created with the goal of measuring latent traits as well as assessing and comparing individuals’ proficiencies by allocating them in proficiency scales, inspiring as well as justifying its use in adaptive testing.

The logistic models of Item Response Theory are unidimensional, which means that a given assessment instrument only measures a single proficiency (or dimension of knowledge). The instrument, in turn, is composed of items in which examinees manifest their latent traits when answering them.

In unidimensional IRT models, an examinee’s proficiency is represented as $\theta$. Usually, $-\infty < \theta < \infty$, but since the scale of $\theta$ is up to the individuals creating the instrument, it is common for the values to be around the normal distribution $N(0; 1)$, such that $-4 < \theta < 4$. Additionally, $\hat{\theta}$ is the estimate of $\theta$. Since a latent trait can’t be measured directly, estimates need to be made, which tend to get closer to the theoretically real $\theta$ as the test progresses in length.

Under the logistic models of IRT, an item is represented by the following parameters

- $a$ represents an item’s discrimination parameter, that is, how well it discriminates individuals who answer the item correctly (or, in an alternative interpretation, individuals who agree with the idea of the item) and those who don’t. An item with a high $a$ value tends to be answered correctly by all individuals whose $\theta$ is above the items difficulty level and wrongly by all the others; as this value gets lower, this threshold gets blurry and the item starts not to be as informative. It is common for $a > 0$;

- $b$ represents an item’s difficulty parameter. This parameter, which is measured in the same scale as $\theta$, shows at which point of the proficiency scale an item is more informative, that is, where it discriminates the individuals who agree and those who disagree with the item. Since $b$ and $\theta$ are measured in the same scale, $b$ follows the same distributions as $\theta$. For a CAT, it is good for an item bank to have as many items as possible in all difficulty levels, so that the CAT may select the best item for each individual in all ability levels;

- $c$ represents an item’s pseudo-guessing parameter. This parameter denotes what is the probability of individuals with low proficiency values to still answer the item correctly. Since $c$ is a probability, $0 < c \leq 1$, but the lower the value of this parameter, the better the item is considered;

- $d$ represents an item’s upper asymptote. This parameter denotes what is the probability of individuals with high proficiency values to still answer the item incorrectly. Since $d$ represents an upper limit of a probability function, $0 < d \leq 1$, but the lower the value of this parameter, the better the item is considered;

For a set of items $N$, when $\forall i \in N, d_i = 1$, the four-parameter logistic model is reduced to the three-parameter logistic model; when $\forall i \in N, c_i = 0$, the three-parameter logistic model
is reduced to the two-parameter logistic model; if all values of \(a\) are equal, the two-parameter logistic model is reduced to the one-parameter logistic model. Finally, when \(\forall i \in N, a_i = 1\), we have the Rasch model (Rasch 1966). Thus, \texttt{catsim} is able of treating all of the logistic models presented above, since the underlying functions of all logistic models related to test simulations are the same, given the correct item parameters.

Under the four-parameter logistic model of IRT, the probability of an examinee with a given \(\theta\) value to answer item \(i\) correctly, given the item parameters, is given by (Magis 2013)

\[
P(X_i = 1|\theta) = c_i + \frac{d_i - c_i}{1 + e^{a_i(\theta - b_i)}}
\]

The information \(i\) gives is calculated as (Magis 2013)

\[
I_i(\theta) = \frac{a^2[(P(\theta) - c)^2(d - P(\theta))]^2}{(d - c)^2(1 - P(\theta))P(\theta)}
\]

In the one-parameter and two-parameter logistic models, an item is most informative when its difficulty parameter is close the examinee’s proficiency. In the three-parameter and four-parameter logistic models, the \(\theta\) value where each item maximizes its information is given by (Magis 2013)

\[
\arg\max_{\theta} I(\theta) = b + \frac{1}{a} \log \left( \frac{x^* - c}{d - x^*} \right)
\]

where

\[
x^* = 2 \sqrt{\frac{-u}{3}} \cos \left( \frac{1}{3} \log \left( \frac{1}{2} \sqrt{\frac{27}{-u^3}} + \frac{4\pi}{3} \right) \right) + 0.5
\]

\[
u = -\frac{c + d - 1}{4}
\]

The item characteristic curve, given by Equation 1; the item information curve, given by Equation 2; and the maximum information point, obtained with Equation 3 are graphically represented in figure 1 for a randomly generated item.

Equations 1, 2 and 3 are available in \texttt{catsim} under module \texttt{catsim.irt} as \texttt{icc()}, \texttt{inf()} and \texttt{max_info()}, respectively. High performance versions of the same functions, which return the response probability and information for all items in the bank, given a \(\theta\) value, or return the \(\theta\) values that maximize information for the entire item bank at once, are also available in the same module as \texttt{icc_hpc()}, \texttt{inf_hpc()} and \texttt{max_info_hpc()}. These versions were developed using the \texttt{numexpr} (Cooke, Hochberg, Alted, Vilata, Thalhammer, Wiebe, de Menten, Valentino, and Cox 2016) package.

The sum of the information of all items in a test is called \textit{test information} (De Ayala 2009):

\[
I(\theta) = \sum_{j \in J} I_j(\theta).
\]
The amount of error in the estimate of an examinee’s proficiency after a test is called the standard error of estimation (De Ayala 2009) and it is given by

\[ SEE = \sqrt{\frac{1}{I(\theta)}}. \]  

(4)

Since the denominator in the right-hand side of the \( SEE \) is \( I(\theta) \), it is clear to see that the more items an examinee answers, the smaller \( SEE \) gets. Additionally, the variance in the estimation of \( \theta \) is given by \( Var = SEE^2 \).

Another measure of internal consistency for the test is test reliability, a value with similar applicability to Cronbach’s \( \alpha \) in Classical Test Theory. It is given by (Thissen 2000)

\[ Rel = 1 - \frac{1}{I(\theta)}. \]

Its value is always lower than 1, with values close to 1 indicating good reliability. Note, however, that when \( I(\theta) < 1, Rel < 0 \). In such cases, the use of \( Rel \) does not make sense, but this situation may be fixed with the application of additional items to the examinee. catsim provides these functions in the catsim.irt module.

### 3.1. Item bank

In catsim, a collection of items is represented as a numpy.ndarray whose rows and columns represent items and their parameters, respectively. Thus, it is referred to as the item matrix. The most important features of the items are situated in the first three columns of the matrix, which represent the parameters \( a, b \) and \( c \), respectively. Item matrices can be generated via the catsim.cat.generate_item_bank function as follows:

```python
items = generate_item_bank(5, '1PL')
```
items = generate_item_bank(5, '2PL')
items = generate_item_bank(5, '3PL')
items = generate_item_bank(5, '4PL')
items = generate_item_bank(5, '4PL', corr=0.5)

These examples depict the generation of an array of five items according to the different logistic models. In the last example, parameters $a$ and $b$ have a correlation of 0.5, an adjustment that may be useful in case simulations require it (Chang, Qian, and Ying 2001). Parameters are extracted from the following probability distributions: $a \sim N(1.2, 0.25)$, $b \sim N(0, 1)$, $c \sim N(0.25, 0.02)$ (Barrada, Olea, Ponsoda, and Abad 2010) and $d \sim U(0.94, 1)$, generated using *numpy*'s random module.

After the simulation, *catsim* adds a fifth column to the item matrix, containing each item’s exposure rate, commonly denoted as $r$. Its value denotes how many times an item has been used and it is calculated as follows:

$$r_i = \frac{q_i}{N}$$

Where $q_i$ represents the number of tests item $i$ has been used on and $N$ is the total number of tests applied.

*catsim* also validates and normalizes item matrices by checking if item parameters are within their minimum and maximum bounds and creating additional columns in the matrix. $1 \times n$, $2 \times n$ and $3 \times n$ matrices can be transformed into a $4 \times n$ using the `irt.normalize_item_bank()` function. The function assumes existent columns represent parameters $b$, $a$, $c$ and $d$, in that order, and adds any missing columns with default values, in such a way that the matrix still represents items in their original logistic models.

### 4. Computerized adaptive testing

Unlike linear tests, in which items are sequentially presented to examinees and their proficiency estimated at the end of the test, in a computerized adaptive test (CAT), an examinees’ proficiency is calculated after the response of each item (Lord 1977, 1980). The updated knowledge of an examinee’s proficiency at each step of the test allows for the selection of more informative items *during* the test itself, which in turn reduce the standard error of estimation of their proficiency at a faster rate.

In general, a computerized adaptive test has a very well-defined lifecycle, presented in figure 2.

1. The examinee’s initial proficiency is estimated;
2. An item is selected based on the current proficiency estimation;
3. The proficiency is reestimated based on the answers to all items up until now;
4. **If** a stopping criterion is met, stop the test. **Else** go back to step 2.

In *catsim*, each phase is separated in its own module, which makes it easy to create simulations combining different methods for each phase. Each module will be explained separately.
Figure 2: The iterative process of a computerized adaptive test.
catsim was built using an object-oriented architecture, an approach which introduces benefits for the package maintenance and expansion. Each phase in the CAT lifecycle is represented by a different abstract class in the package. Each of these classes represent either a proficiency initialization procedure, an item selection method, a proficiency estimation procedure or a stopping criterion. The implementations of these procedures are made available in dedicated modules. Users and contributors may independently implement their own methods for each phase of the CAT lifecycle, or even an entire new CAT iterative process while still using catsim and its features to simulate tests and plot results, they respect the contracts of the provided abstract classes. Modules and their corresponding abstract classes are presented on table 1.

5. Package architecture

catsim was built using an object-oriented architecture, an approach which introduces benefits for the package maintenance and expansion. Each phase in the CAT lifecycle is represented by a different abstract class in the package. Each of these classes represent either a proficiency initialization procedure, an item selection method, a proficiency estimation procedure or a stopping criterion. The implementations of these procedures are made available in dedicated modules. Users and contributors may independently implement their own methods for each phase of the CAT lifecycle, or even an entire new CAT iterative process while still using catsim and its features to simulate tests and plot results, they respect the contracts of the provided abstract classes. Modules and their corresponding abstract classes are presented on table 1.

5.1. Proficiency initialization

The initialization procedure is done only once during each examinee’s test. In it, the initial value of an examinee’s proficiency $\hat{\theta}_0$ is selected. Ideally, the closer $\hat{\theta}_0$ is to $\theta$, the faster it converges to $\theta$, the examinee’s true proficiency value (Sukamolson 2002). The initialization procedure may be done in a variety of ways. A standard value can be chosen to initialize all examinees (FixedInitializer); it can be chosen randomly from a probability distribution (RandomInitializer); the place in the item bank with items of more information can be chosen to initialize $\hat{\theta}_0$ (Dodd 1990) etc.

In catsim, initialization procedures can be found in the catsim.initialization module.

5.2. Item selection

At time step $t$ and a given value for $\hat{\theta}_t$, an item $i_t$ is chosen from the item bank and presented to the examinee. The answers to all $i_0 \rightarrow i_t$ items are then used to estimate $\hat{\theta}_{t+1}$.

Table 2 lists the available selection methods provided by catsim. Selection methods can be found in the catsim.selection module.

MaxInfoSelector selects $i_{t+1}$ as the item that maximizes information gain, that is, to select $\arg\max_{i \in N} I_i(\hat{\theta}_t)$. This method guarantees faster decrease of standard error. However, it has been shown to have drawbacks, like causing overexposure of few items in the item bank while ignoring items with smaller information values.

AStratifiedSelector, MaxInfoStratificationSelector, AStratifiedBBlockingSelector and MaxInfoBBlockingSelector are classified (Georgiadou, Triantafillou, and Economides 2007) as stratification methods. In the first one, the item bank is sorted in ascending order according to the items discrimination parameter and then separated into $K$ strata ($K$ being
Table 2: Item selection methods implemented in catsim.

| Class                     | Author                                      |
|---------------------------|---------------------------------------------|
| AStratifiedBBlockingSelector | Chang et al. (2001)                       |
| AStratifiedSelector       | Chang and Ying (1999)                      |
| ClusterSelector           | Meneghetti (2015)                          |
| IntervalIntegrationSelector | —                                           |
| LinearSelector            | —                                           |
| MaxInfoBBlockingSelector  | Barrada, Mazuela, and Olea (2006)          |
| MaxInfoSelector           | Lord (1977)                                |
| MaxInfoStratificationSelector | —                                           |
| RandomSelector            | Barrada et al. (2006)                      |
| RandomesqueSelector       | Kingsbury and Zara (1989)                  |
| The54321Selector          | McBride and Martin (1983)                  |

The test size), each stratum containing gradually higher average discrimination. The alpha-stratified selector then selects the first non-administered item from stratum $k$, in which $k$ represents the position in the test of the current item the examinee is being presented. The second method separates items in strata according to their maximum $I(\theta)$ value. The last two are variants in which item difficulty is guaranteed to have a distribution close to uniform among the $K$ strata, after the discovery that there is a positive correlation between $a$ and $b$ (Chang et al. 2001).

RandomSelector, RandomesqueSelector and The54321Selector are methods that apply randomness to item selection. The first one selects items randomly from the item bank; the second selects an item randomly from the $n$ most informative items at that step of the test, $n$ being a user-defined variable. The last method also selects between the $n$ most informative items, but at each new item that is selected, $n = n - 1$, guaranteeing that more informative items are selected as the test progresses.

LinearSelector is a selector that mimics a linear test, that is, item indices are predefined and all examinees are presented the same items. This method is useful for the simulation of an ordinary paper and pencil test.

ClusterSelector clusters items according to their parameter values and selects items from the cluster that contains either the most informative item or the highest average information.

IntervalIntegrationSelector is an experimental selection method that selects the item whose integral around an interval $\delta$ of the peak of information is maximum, like so

$$\arg \max_{i \in N} \int_{\hat{\theta} - \delta}^{\hat{\theta} + \delta} I_i(\hat{\theta}).$$

It uses scipy.integrate quad function for the process of integration.

5.3. Proficiency estimation

Proficiency estimation occurs whenever an examinee answers a new item. After an item is selected, catsim simulates an examinee’s response to a given item by sampling a binary value from the Bernoulli distribution, in which the value of $p$ is given by Equation 1. Given a dichotomous (binary) response vector and the parameters of the corresponding items that
were answered, it is the job of an estimator to return a new value for the examinee’s \( \hat{\theta} \). This value reflects the examinee’s proficiency, given his or hers answers up until that point of the test.

In Python, an example of a list that may be used as a valid dichotomous response vector is as follows:

```python
response_vector = [True, True, True, False, True, False, True, False, False]
```

Estimation techniques are generally separated between maximum-likelihood estimation procedures (whose job is to return the \( \hat{\theta} \) value that maximizes a **likelihood** function, presented in `catsim.irt.log_likelihood`) and Bayesian estimation procedures, which use prior information of the distributions of examinee’s proficiencies to estimate new values for them.

The likelihood function of a given \( \hat{\theta} \) value given the answers to \( I \) items is given by

\[
L(X_{Ij}|\theta, a_I, b_I, c_I) = \prod_{i=1}^{I} P_{ij}(\theta)^{X_{ij}} Q_{ij}(\theta)^{1-X_{ij}}
\]

For mathematical reasons, finding the maximum of \( L(X_{Ij}) \) includes using the product rule of derivations. Since \( L(X_{Ij}) \) has \( j \) parts, it can be quite complicated to do so. Also, for computational reasons, the product of probabilities can quickly tend to 0, so it is common to use the log-likelihood in maximization/minimization problems, transforming the product of probabilities in a sum of probabilities. This function, presented in `catsim` as `catsim.irt.log_likelihood`, is given by

\[
\log L(X_{Ij}|\theta, a_I, b_I, c_I) = \sum_{i=1}^{I} \{ x_{ij} \log P_{ij}(\theta) + (1 - x_{ij}) \log Q_{ij}(\theta) \}.
\]

`catsim` provides a maximum likelihood estimation procedure based on the hill climbing optimization technique, called `HillClimbingEstimator`. Since maximum likelihood estimators tend to fail when all observed values are equal, the provided estimation procedure employs the method proposed by Dodd (1990) to select the value of \( \hat{\theta} \) as follows:

\[
\hat{\theta}_{t+1} = \begin{cases} 
\hat{\theta}_t + \frac{b_{max} - \hat{\theta}_t}{2} & \text{if } X_t = 1 \\
\hat{\theta}_t - \frac{\hat{\theta}_t - b_{min}}{2} & \text{if } X_t = 0 
\end{cases},
\]

where \( b_{max} \) and \( b_{min} \) are the maximum and minimum \( b \) values in the item bank, respectively.

The second estimation method uses `scipy` (Jones, Oliphant, and Peterson 2001) optimization module, more specifically the `differential_evolution()` function, to find the global maximum of the likelihood function. These proficiency estimation procedures can be found in the `catsim.estimation` module.

Figure 3 shows the performance of the available estimators. `HillClimbingEstimator` evaluates the log-likelihood function an average of 7 times before reaching a step-size smaller than \( 10^{-5} \), while the `scipy`-based `DifferentialEvolutionEstimator` evaluates the function four times more.

Estimators may benefit from prior knowledge of the estimate, such as known lower and upper bounds for \( \theta \) or the value of \( \hat{\theta}_t \) when estimating \( \hat{\theta}_{t+1} \).
Figure 3: Average number of function evaluations and graphical representation of function maxima found by the available estimation methods in catsim.

5.4. Stopping criterion

Since items in a CAT are selected during the test, a stopping criterion must be chosen such that, when achieved, no new items are presented to the examinee and the test is deemed finished. These stopping criteria might be achieved when the test reaches a fixed number of items or when the standard error of estimation from Equation 4 (De Ayala 2009) reaches a lower threshold (Lord 1977), among others. Both of the aforementioned stopping criteria are implemented as MaxItemStopper and MinErrorStopper, respectively, and are located in the catsim.stopping module.

6. The simulation process

The simulation process is configured using the Simulator class from the catsim.simulation module. A Simulator is created by passing an Initializer, a Selector, an Estimator and a Stopper object to it, as well as either an empirical or generated item matrix and a list containing examinees $\theta$ values or a number of examinees. If a number of examinees is passed, catsim generates their $\theta$ values from the normal distribution, using as $\mu_\theta = \mu_b$ and $\sigma_\theta = \sigma_b$, where $b$ represents the difficulty parameter of all items in the item matrix. If an item matrix is not provided, $\mu_\theta = 0$ and $\sigma_\theta = 1$.

When used in conjunction with the Simulator, all four abstract classes (Initializer, Selector, Estimator and Stopper) as well as their implementations have complete access to the current and past states of the simulation, which include:

1. item parameters;
2. indexes of the administered items to all examinees;
3. current and past proficiency estimations of all examinees;
4. each examinee’s response vector.
from catsim.cat import generate_item_bank
from catsim.initialization import RandomInitializer
from catsim.selection import MaxInfoSelector
from catsim.estimation import HillClimbingEstimator
from catsim.stopping import MaxItemStopper
from catsim.simulation import Simulator

initializer = RandomInitializer()
selector = MaxInfoSelector()
estimator = HillClimbingEstimator()
stopper = MaxItemStopper(20)
s = Simulator(generate_item_bank(100), 10)
s.simulate(initializer, selector, estimator, stopper)

During the simulation, the Simulator calculates and maintains $\hat{\theta}$, SEE and Var after every item is presented to each examinee, as well as the indexes of the presented items in the order they were presented. These values are exposed as properties of the Simulator and are accessed as s.estimations, s.see, s.var, s.administered_items and so on.

6.1. Validity measures

A series of validity measures have been implemented, in order to evaluate the results of the adaptive testing simulation. The first two are measurement bias and mean squared error (Chang et al. 2001), while the last two are the root mean squared error and test overlap rate (Barrada et al. 2006; Barrada, Olea, and Ponsoda 2007; Barrada, Olea, Ponsada, Abad, Ponsoda, and Abad 2009; Barrada et al. 2010; Barrada, Abad, and Olea 2014).

$$Bias = \frac{\sum_{j=1}^{J}(\hat{\theta}_j - \theta_j)}{J}$$

$$MSE = \frac{\sum_{j=1}^{J}(\hat{\theta}_j - \theta_j)^2}{J}$$

$$RMSE = \sqrt{\frac{\sum_{j=1}^{J}(\theta_j - \hat{\theta}_j)^2}{J}}$$

$$T = \frac{N}{Q} S^2 + \frac{Q}{N}$$

The first three are indicators of the error of measurement between the J examinees’ true proficiency $\theta_j$ and the proficiency estimated at the end of the CAT, $\hat{\theta}_j$. Test overlap rate, according to Barrada et al. (2009), “provides information about the mean proportion of items shared by two examinees”, where $N$ indicates the number of items in the item bank, $Q$ is the...
number of items in the test and $S_r^2$, the variance in the exposure rates of all items. A value of $T = 0.3$ indicates that two examinees share, on average, 30% of the same items in a test. It is possible to see that, since $Q/N$ is a constant, $T$ reaches its lowest value when $S_r^2 \approx 0$. For that, all items in the item bank must be used in a homogeneous way.

After the simulation, the Simulator calculates the aforementioned validity measures and exposes them as properties, which are accessed as $s.bias, s.mse, s.rmse$ and $s.overlap\_rate$.

6.2. Plotting items and results

catsim provides a series of plotting functions in the catsim.plot module, which uses matplotlib (Hunter 2007) to plot item curves, as well as plotting proficiency estimations and validity measures for each item answered by every examinee during the simulation.

The following example shows how to plot an item’s characteristic curve, its information curve as well as both curves in the same figure, as presented in figure 1.

```python
item = generate_item_bank(1)[0]
plot.item_curve(item[0], item[1], item[2], item[3], ptype='icc')
plot.item_curve(item[0], item[1], item[2], item[3], ptype='iic')
plot.item_curve(item[0], item[1], item[2], item[3], ptype='both')
```

Given a simulator $s$, the test progress of an examinee can be plotted using the function `test_progress()` and passing the examinee’s index, like so:

```python
plot.test_progress(simulator=s, index=0, true_theta=s.examinees[0],
                   info=True, var=True, see=True)
```

These commands generate the graphics depicted in figure 4.

At last, item exposure rates can be plotted via the `item_exposure()` function. Given a simulator $s$ or an item matrix $i$ with the last column representing the items exposure rate, item exposure can be plotted as the following example shows:

```python
plot.item_exposure(simulator=s, par='b')
plot.item_exposure(items=i, ptype='line')
```

Both of these commands generate graphics similar to the one shown in figure 5. The `par` parameter allows exposure rates to be sorted by one of the item parameters before plotting. The `ptype` parameter allows the chart to be plotted using either bars or lines.

7. Independent usage

Aside from being used in the simulation process, implementations of Initializer, Selector, Estimator and Stopper may also be used independently. Given the appropriate parameters, each component is able to return results in an accessible way, allowing catsim to be used in other software programs. Table 3 shows the main functions of each component, the parameters they commonly use and their return values.

The code below exemplifies the independent usage of CAT components outside of the simulator.
Figure 4: Graphics showing the progress of certain measurements during an examinee’s test.

Figure 5: Chart presenting item exposure rates of an item bank.
Table 3: Main functions of each component and the parameters they commonly use.

| Class       | Main method | Common parameters                                      | Returns                           |
|-------------|-------------|-------------------------------------------------------|-----------------------------------|
| Initializer | initialize  | None                                                  | $\theta_0$                        |
| Selector    | select      | item bank, indexes of administered items, $\theta_t$  | Index of the next item           |
| Estimator   | estimate    | item bank, indexes of administered items, $\theta_t$, response vector | $\theta_{t+1}$                   |
| Stopper     | stop        | parameters of administered items, $\theta_t$         | True to stop the test, otherwise False |

8. Examples

This section presents different examples of running simulations through catsim. Figure 6 presents a simulation with 10 examinees, a random $\theta_0$ initializer, a maximum information item selector, a hill climbing estimator and a stopping criterion of 20 items.

```python
bank_size = 5000
items = generate_item_bank(bank_size)
responses = [True, True, False, False]
administered_items = [1435, 3221, 17, 881]

initializer = RandomInitializer()
selector = MaxInfoSelector()
estimator = HillClimbingEstimator()
stopper = MaxItemStopper(20)

est_theta = initializer.initialize()
item_index = selector.select(items=items,
    administered_items=administered_items,
est_theta=est_theta)
new_theta = estimator.estimate(items=items,
    administered_items=administered_items,
    response_vector=responses, est_theta=est_theta)
_stop = stopper.stop(administered_items=items[administered_items], theta=est_theta)
```

Figure 7 presents a way of using predetermined proficiencies by passing either a list or a one-dimensional numpy.ndarray. It also uses the minimum error stopping criterion, making each examinee have tests of different sizes.
catsim: Computerized Adaptive Testing Simulator

Figure 6: CAT results of an examinee. Items were selected by the maximum information selector and test stops at 20 items.

```python
evaluatees = numpy.random.normal(size=10)
s = Simulator(items, evaluatees)
s.simulate(RandomInitializer(), MaxInfoSelector(),
HillClimbingEstimator(), MinErrorStopper(.3))
catplot.test_progress(simulator=s,index=0, info=True)
```

The last example shows a linear item selector, whose item indexes are selected in order from a list. The example shows indexes being randomly selected from the item bank, but they can also be passed directly, such as `indexes = [1, 10, 12, 27, 116]`.

```python
s = Simulator(items, 10)
indexes = numpy.random.choice(items.shape[0], 50, replace=False)
s.simulate(RandomInitializer(), LinearSelector(indexes),
HillClimbingEstimator(), MaxItemStopper(50))
catplot.test_progress(simulator=s,index=0, info=True, see=True)
```

9. Discussion

This paper has presented catsim, a computerized adaptive testing simulator written in Python. catsim separates each phase in the CAT iterative process into different components and provides implementations of different methods from the literature, as well as interfaces which can be used by end users to create their own methods. The simulator provides a complete history of the test progress for each examinee, including items chosen, answers given and measures taken after each answer, as well as validity measures for the whole test. Simulation results, item parameters and exposure rates can then be plotted with the provided functions.
Figure 7: CAT results of an examinee. Items were selected by the maximum information selector and test stops when a minimum error threshold is reached.

Figure 8: Test results of an examinee. Items were selected linearly and test stops at 50 items.
**catsim**: Computerized Adaptive Testing Simulator

**catsim** fills a gap in the Python scientific community, being the only package specialized in computerized adaptive tests and the logistic models of Item Response Theory. Other functionalities that may benefit the community include item parameter and examinee proficiency estimation through a dichotomous data matrix and the implementation of Bayesian estimators, whose behavior contrasts with the maximum likelihood estimators already provided.

**References**

Barrada JR, Abad FJ, Olea J (2014). “Optimal Number of Strata for the Stratified Methods in Computerized Adaptive Testing.” *The Spanish Journal of Psychology, 17*. ISSN 1138-7416. doi:10.1017/sjp.2014.50. URL http://www.journals.cambridge.org/.

Barrada JR, Mazuela P, Olea J (2006). “Maximum Information Stratification Method for Controlling Item Exposure in Computerized Adaptive Testing.” *Psicothema, 18*(1), 156–159. ISSN 02149915. URL http://156.35.33.98/reunido/index.php/PST/article/view/8411.

Barrada JR, Olea J, Ponsada V, Abad FJ, Ponsoda V, Abad FJ (2009). “Test Overlap Rate and Item Exposure Rate as Indicators of Test Security in CATs.” In *Proceedings of the 2009 {GMAC} Conference on Computerized Adaptive Testing*. URL http://iacat.org/sites/default/files/biblio/cat09barrada.pdf.

Barrada JR, Olea J, Ponsoda V (2007). “Methods for Restricting Maximum Exposure Rate in Computerized Adaptive Testing.” *Methodology, 3*(1), 14–23. ISSN 16141881. doi:10.1027/1614-2241.3.1.14. URL http://psycnet.apa.org/journals/med/3/1/14/.

Barrada JR, Olea J, Ponsoda V, Abad FJ (2010). “A Method for the Comparison of Item Selection Rules in Computerized Adaptive Testing.” *Applied Psychological Measurement, 34*(6), 438–452. ISSN 0146-6216. doi:10.1177/0146621610370152. URL http://apm.sagepub.com/cgi/doi/10.1177/0146621610370152.

Chalmers RP (2016). “Generating Adaptive and Non-Adaptive Test Interfaces for Multidimensional Item Response Theory Applications.” *Journal of Statistical Software, 71*(5). ISSN 1548-7660. doi:10.18637/jss.v071.i05. URL http://www.jstatsoft.org/v71/i05/.

Chang HH, Qian J, Ying Z (2001). “a-Stratified Multistage Computerized Adaptive Testing with b-Blocking.” *Applied Psychological Measurement, 25*(4), 333–341. URL http://apm.sagepub.com/content/25/4/333.short.

Chang HH, Ying Z (1999). “a-Stratified Multistage Computerized Adaptive Testing.” *Applied Psychological Measurement, 23*(3), 211–222. ISSN 0146-6216. doi:10.1177/0146621992031338. URL http://apm.sagepub.com/content/25/4/333.shorthttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.264&rep=rep1&type=pdf.

Choi SW, King DR (2015). “MAT: Multidimensional Adaptive Testing.”

Cooke D, Hochberg T, Alted F, Vilata I, Thalhammer G, Wiebe M, de Menten G, Valentino A, Cox D (2016). “numexpr: Fast Numerical Expression Evaluator for NumPy.”
De Ayala RJ (2009). *The Theory and Practice of Item Response Theory*. Guilford Press, New York. ISBN 1-59385-869-8. URL http://books.google.com/books?id=-k36zb0Ba28C&pgis=1.

Dodd BG (1990). “The Effect of Item Selection Procedure and Stepsize on Computerized Adaptive Attitude Measurement Using the Rating Scale Model.” *Applied Psychological Measurement*, 14(4), 355–366. ISSN 0146-6216. doi:10.1177/014662169001400403. URL http://apm.sagepub.com/content/14/4/355.shorthttp://iacat.org/sites/default/files/biblio/v14n4p355.pdf.

Georgiadou EG, Triantafillou E, Economides AA (2007). “A Review of Item Exposure Control Strategies for Computerized Adaptive Testing Developed from 1983 to 2005.” *The Journal of Technology, Learning and Assessment*, 5(8).

Hunter JD (2007). “Matplotlib: a 2D Graphics Environment.” *Computing In Science & Engineering*, 9(3), 90–95. doi:10.1109/MCSE.2007.55.

Jones E, Oliphant T, Peterson P (2001). *SciPy: Open source scientific tools for Python*. URL http://www.scipy.org/.

Kingsbury GG, Zara AR (1989). “Procedures for Selecting Items for Computerized Adaptive Tests.” *Applied Measurement in Education*, 2(4), 359–375.

Lord FM (1977). “A Broad-Range Tailored Test of Verbal Ability.” *Applied Psychological Measurement*, 1(1), 95–100. ISSN 0146-6216. doi:10.1177/014662167700100115. URL http://apm.sagepub.com/content/1/1/95.short.

Lord FM (1980). *Applications of Item Response Theory to Practical Testing Problems*. Routledge. ISBN 978-0-89859-006-7.

Lord FM, Novick MR, Birnbaum A (1968). *Statistical Theories of Mental Test Scores*. Addison-Wesley, Oxford, England.

Magis D (2013). “A Note on the Item Information Function of the Four-Parameter Logistic Model.” *Applied Psychological Measurement*, 37(4), 304–315. ISSN 0146-6216, 1552-3497. doi:10.1177/0146621613475471. URL http://apm.sagepub.com/cgi/doi/10.1177/0146621613475471.

Magis D, Raiche G (2012). “Random Generation of Response Patterns under Computerized Adaptive Testing with the R Package catR.” *Journal of Statistical Software*, 48(8), 1–31. URL http://www.jstatsoft.org/v48/i08/.

McBride JR, Martin JT (1983). “Reliability and Validity of Adaptive Ability Tests in a Military Setting.” In *New horizons in testing: Latent trait test theory and computerized adaptive testing*, pp. 224–236. Academic Press New York.

Meneghetti DR (2015). *Metodologia de Seleção de Itens em Testes Adaptativos Informatizados Baseada em Agrupamento por Similaridade*. Mestrado, Centro Universitário da FEI. URL https://www.researchgate.net/publication/283944553_Metodologia_de_selecao_de_itens_em_Testes_Adaptativos_Informatizados_baseada_em_Agrupamento_por_Similaridade.
Nydick SW (2015). “catIrt: An R Package for Simulating IRT-Based Computerized Adaptive Tests.”

Rasch G (1966). “An Item Analysis Which Takes Individual Differences into Account.” *British Journal of Mathematical and Statistical Psychology*, 19(1), 49–57. ISSN 2044-8317. doi: 10.1111/j.2044-8317.1966.tb00354.x. URL http://onlinelibrary.wiley.com/doi/10.1111/j.2044-8317.1966.tb00354.x/abstract.

Sukamolson S (2002). “Computerized Test/Item Banking And Computerized Adaptive Testing for Teachers and Lecturers.” *Information Technology and Universities in Asia – ITUA*. URL http://www.stc.arts.chula.ac.th/ITUA/Papers_for_ITUA_Proceedings/Suphat2.pdf.

Thissen D (2000). “Reliability and Measurement Precision.” In H Wainer, NJ Dorans, R Flaugher, BF Green, RJ Mislevy (eds.), *Computerized adaptive testing: A primer*. Routledge.

**Affiliation:**
Department of Graduate Electrical Engineering  
Centro Universitário FEI  
3972 Humberto de Alencar Castelo Branco Avenue, Sao Paulo, Brazil

Douglas De Rizzo Meneghetti  
E-mail: douglasrizzo@fei.edu.br  
URL: http://douglasrizzo.com.br/

Plinio Thomaz Aquino Junior  
E-mail: plinio.aquino@fei.edu.br  
URL: http://fei.edu.br/~plinio.aquino/