Behavior Research Methods
2006, 38 (3), 532-541

Using the open-source statistical language R to analyze the dichotomous Rasch model

YUELIN LI
Memorial Sloan-Kettering Cancer Center, New York, New York

R, an open-source statistical language and data analysis tool, is gaining popularity among psychologists currently teaching statistics. R is especially suitable for teaching advanced topics, such as fitting the dichotomous Rasch model—a topic that involves transforming complicated mathematical formulas into statistical computations. This article describes R's use as a teaching tool and a data analysis software program in the analysis of the Rasch model in item response theory. It also explains the theory behind, as well as an educator's goals for, fitting the Rasch model with joint maximum likelihood estimation. This article also summarizes the R syntax for parameter estimation and the calculation of fit statistics. The results produced by R is compared with the results obtained from MINISTEP and the output of a conditional logit model. The use of R is encouraged because it is free, supported by a network of peer researchers, and covers both basic and advanced topics in statistics frequently used by psychologists.

The open-source software movement has changed how behavioral scientists conduct research, carry out data analysis, and teach students the general principles of computation (Allbritton, 2003). This has resulted in the availability of a free statistical programming language called R. R's code base is drawn from S, a widely accepted statistical language (Becker, Chambers, & Wilks, 1988; R Development Core Team, 2004). R provides a computational environment and highly extensible software libraries for statistical analysis and graphics. These libraries include a variety of basic and advanced statistical methods used in the behavioral sciences. Additionally, R and S/Splus include open-source implementations of item response theory (IRT), an important topic usually covered in advanced undergraduate- or graduate-level psychometric classes. However, access to these programs remains mostly limited to researchers who are proficient in the S/Splus and R language (Fox, 2002; Rizopoulos, 2005; Waller, 1998), Bayesian IRT methods (Fox, 2002; Karabatsos & Batchelder, 2003), and Markov chain Monte Carlo simulation techniques (de la Torre, Stark, & Chernyshenko, 2006; Fox, 2002; Karabatsos & Batchelder, 2003; Martin & Quinn, 2006; Patz & Junker, 1999a, 1999b; Roberts & Huang, 2006). A readily available beginner-level solution in R to the basics of IRT would help students transfer to these sophisticated and powerful tools. This article provides such a tutorial. It explains the basics of the dichotomous Rasch model, thus aiding student novices and instructors who may use R as a teaching tool.

R's pedagogical value makes it well suited for teaching the underlying logic of and computations for statistical methods. R's design philosophy emphasizes data visualization, computational simulations, and data manipulation (Venables & Ripley, 2000). These design characteristics not only help students understand the critical abstract theoretical concepts at their foundation (Marasinghe, Meeker, Cook, & Shin, 1996), they also help students connect abstract statistical concepts with computations (Bradstreet, 1996; also see Mills, 2002, for a literature review). The close resemblance between the R syntax and the statistical formulas promotes a better grasp of these fundamental concepts.

This article describes how to use R to perform maximum likelihood estimation in the analysis of the dichotomous Rasch model. The main purpose is to supplement the technical materials available to instructors using popular texts such as Wright and Stone (1979), Wright and Masters (1982), Embretson and Reise (2000) and Bond and Fox (2001) to teach psychology students an introductory session on Rasch analysis. Thus, the emphasis is on showing how statistical formulas and their conceptual foundations can be understood through data visualization and data object manipulation. A lesson plan similar to the one in Chapter 8 of Embretson and Reise (2000) is summarized in eight sections. Section 1 provides a rationale for encouraging the use of R in IRT. Sections 2–6 describe how instructors can use R for pedagogical purposes. The materials progress as follows: Section 2 covers the R syntax basics. Section 3 explains basic concepts about the dichotomous Rasch model. Section 4 provides a visual simulation of the maximum likelihood estimation (MLE). Section 5 covers a simplified quantitative MLE solution. Finally, Section 6 presents a full joint maximum likeli-

This study was supported in part by Grant R03DC04486 from the National Institute on Deafness and Other Communication Disorders. The author thanks Jonathan Baron at the Department of Psychology, University of Pennsylvania, for his guidance, and Justin Mathew and Joanna Lautenberger for their help in manuscript preparation. Correspondence concerning this article should be addressed to Y. Li, Department of Psychiatry & Behavioral Sciences, Memorial Sloan-Kettering Cancer Center, 1275 York Ave., New York, NY 10021 (email: liy12@mskcc.org).

Copyright 2006 Psychonomic Society, Inc.

hood (JML) solution. The discussions focus on fitting a dichotomous Rasch model with the JML algorithm used in popular proprietary software programs such as WINSTEPS (Linacre, 2004). In Sections 7–8, the results obtained from R are compared with those from MINISTEP (a free but reduced version of WINSTEPS) and the output of a conditional logit model.

The example used herein is the Knox Cube Test in MINISTEP, also described in Wright and Stone (1979). Therefore, the present tutorial can be used to provide an open-source alternative for students' computational needs. Additionally, this article attempts to inspire experienced researchers to use R in IRT and to encourage further development of open-source IRT programs.

1. Rationale for Using R in Rasch Analysis

Students, instructors, and researchers benefit from R for slightly different reasons. R is attractive to students primarily because it can be freely downloaded from the R Web site. The R core programming language, its extensions, its documentations, and its user community reside at the R Web site (www.r-project.org).

Although R is free, using R does not require any sacrifice of performance or capacity. R runs on many hardware platforms and covers a wide range of statistical methods frequently used by psychologists. Free texts and tutorials are available electronically. Students can often further boost their proficiency in R programming by using program templates posted online by experts.

Instructors benefit from R's collection of powerful and highly customizable graphics routines. These routines can be used to visualize data patterns prior to data analysis. The syntax of R more closely resembles the actual formulas used in the computations. These features, combined with the powerful random sample generation subroutines, may help students better appreciate the connection between abstract formulas and concrete data objects, thus acquiring a more continuous and logical understanding of complicated statistical concepts.

For researchers, R provides a computational environment for implementing cutting edge statistical methods such as the advanced IRT methods cited above. Analyses not supported by proprietary software packages can be programed relatively easily in the highly customizable R language. R has a large online user support group that promptly provides peer-monitored solutions to posted questions.

Proprietary statistical software packages can certainly be programmed to accomplish the same feats as R. Data analysis has become more of a matter of style than conformity to any one statistical package. R promises to be a general-purpose statistical language in an area where special-purpose software packages dominate how students learn Rasch modeling (for reviews of special-purpose software packages, see Bond & Fox, 2001, and Embretson & Reise, 2000; see also Sheu, Chen, Su, & Wang, 2005, for a solution in SAS).

A limitation of R is that it is not yet equipped with many features found in special-purpose statistical programs. The somewhat steep learning curve can be intimidating for beginners. Nevertheless, the object-oriented design of R makes it relatively easy to add new functions to existing statistical libraries. This tutorial is actually built on the existing `stats4` library developed by the R Development Core Team and contributors worldwide. As beginners gain experience, they may benefit from the programmability of R in complicated computations such as JML. This tutorial outlines some of these advantages.

2. R Basics and Typographical Conventions

This section covers some of the basics of the R syntax in preparation for the more advanced Rasch analysis in Sections 3–6. It also summarizes the typographical conventions used in this article. More detailed tutorials on basic and advanced use of R can be found online, and are not covered here (see a list of downloadable texts at www.r-project.org/doc/bib/R-publications.html; and introductory textbooks, e.g., Dalgaard, 2002, and Maindonald & Braun, 2003).

In the R command-line environment, a user enters one command at a time and receives the output after R has interpreted and processed the command. In this article, R commands are in plain text font, as in `exp(0.5) / (1 + exp(0.5))`. This example R code calculates the inverse log odds of a probability of .5, as in $e^{0.5}/(1 + e^{0.5})$. Texts that follow a pound/number sign, such as `# calculate inverse logit`, represent comments for annotations.

The remainder of this section covers two features of R used extensively in this article: R's built-in matrix calculation capabilities and customizable functions. The iterative calculations in Rasch analysis are almost exclusively performed on data matrices, which necessitate a more thorough introduction on matrix computations.

The built-in `apply()` function, which helps students visualize complex matrix calculations, is an important part of R's matrix calculation capabilities. Suppose a mathematical formula calls for the sums of the columns in a data matrix x_{ij}. The mathematical notation is

$$\sum_{i=1}^{n} x_{ij} = x_{1j} + x_{2j} + \ldots + x_{nj}.$$

This can be done in one line of R code, using the `apply(x, 2, sum)` function call. Shown in Figure 1, `apply(x, 2, sum)` takes the matrix x_{ij} (the rectangle on top) and applies the `sum()` function on each column from the top row to the bottom row.

The sums of j columns are reduced into a vector of j numbers (the rectangle at the bottom). In R, a two-dimensional matrix is indexed by rows and columns (identified as dimensions 1 and 2, respectively). Thus, in the `apply()` example above, the `sum()` function is applied over dimension 2. Abstract mathematical calculations can be visualized as operations performed on data objects, thereby simplifying the somewhat complex mathematical computation.

Another extensively used feature of R involves building customized functions from basic elements. For example, the following function calculates the inverse log odds of probability x:

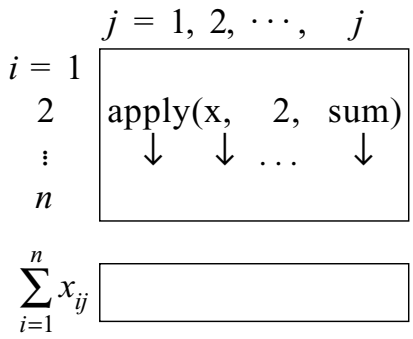

Figure 1. A diagram showing how R's `apply()` function sums across rows of a matrix into a vector.

```
> ilog <- function(x)
          { exp(x) / ( 1 + exp(x)) }.
```

The above defines a function `ilog(x)` that takes only one parameter `x` and outputs the inverse log odds of a probability. For example,

```
> ilog(0.5)
[1] 0.6224593
```

Customized functions make repetitive calculations less tedious. They can also be nested within other more advanced data analysis routines for complex computations such as Rasch modeling. In Sections 3–6, a Rasch model is defined in a customized function and analyzed in an iterative model-fitting process that relies on the matrix computation capabilities of R.

3. Basic Principles of the Dichotomous Rasch Model

In the 1960s, Georg Rasch proposed the dichotomous Rasch model to measure a person's latent trait level from a probabilistic perspective (Rasch, 1960/1980). The probability of a person answering an item correctly depends on the person's underlying ability and item difficulty. The simplest Rasch model begins with the following symbolic form:

$$\ln\left[\frac{\Pr_{si}}{\left(1-\Pr_{si}\right)}\right] = \theta_s - \beta_i, \tag{1}$$

where $\Pr_{si}$ is the probability of person s answering item i correctly. The difference between the person's underlying ability (θ_s) and the item's difficulty (β_i) determines the log odds of a person answering the item correctly. Note that if the person's ability equals the item's difficulty (e.g., $\theta_s = \beta_i = .25$), then the right-hand side of the equation is zero. It means that the person has a zero log odds; in other words, he or she has a 50–50 chance of correctly answering an item that matches his or her ability.

The θ_s and β_i parameters are to be estimated from the pass/fail response data. In textbooks about Rasch modeling, Equation 1 is often transformed into a different form. By applying the inverse logit transformation, we get

$$\exp\left(\ln\left[\frac{\Pr_{si}}{\left(1-\Pr_{si}\right)}\right]\right) = \exp\left(\theta_s - \beta_i\right),$$

and the exponent and natural log on the left-hand side cancel out. After some rearrangement, the formula becomes

$$\Pr\left(x_{si} = 1 \mid \theta_s, \beta_i\right) = \frac{\exp\left(\theta_s - \beta_i\right)}{1 + \exp\left(\theta_s - \beta_i\right)}, \tag{2}$$

which models the probability of test performance given the parameters. A computer program for Rasch analysis makes iterative adjustments until the estimated parameters fit the observed data as well as possible. In technical terms, the computer program performs MLE to determine the most probable estimates of examinee ability and item difficulty. In the next section, the MLE process is explained with a visual aid.

4. Maximum Likelihood Estimation by Visualization

This section aims to provide students with an intuitive understanding of the basic MLE technique and prepares students for a more in-depth, quantitative MLE analysis in the next section. For simplicity, the example involves only one examinee and three items.

Maximum likelihood estimation, as the name implies, is a procedure for finding the most probable parameters in a stochastic model. The term *likelihood* describes the probability of observing the data pattern given a set of candidate parameters. When fitting a Rasch model, MLE is used to find the values of examinees' ability levels and item difficulties that are most likely to produce the observed data. This procedure is called *joint MLE* (JML) because trait levels and item difficulties are estimated jointly by two separate steps of MLE. The following simplistic example shows how the MLE procedure determines which set of candidate parameters wins out.

Suppose a student has a trait level of 1.0 and is presented with three items to solve. The difficulties of the three items are 0, 1.0, and 2.0. Moreover, the student solves items 1 and 2, but not item 3. An immediate question is, what is the probability that a student with a 1.0 ability level solves an item with a difficulty of 0 and another item with a difficulty of 1.0 but fails to solve an item with a difficulty of 2.0?

The Rasch model provides the formulas to calculate this probability. $\Pr(x = 1)$ represents answering an item x correctly. Its compliment, $\Pr(x = 0)$, represents answering an item x incorrectly. The two probabilities are

$$\Pr\left(x = 1 \mid \theta, \beta\right) = P\left(\theta, \beta\right) = \frac{\exp\left(\theta - \beta\right)}{1 + \exp\left(\theta - \beta\right)},$$

and

$$\Pr\left(x = 0 \mid \theta, \beta\right) = Q\left(\theta, \beta\right) = \frac{1}{1 + \exp\left(\theta - \beta\right)}.$$

The R syntax shows that there is a 73% probability that the student (with an ability of 1.0) solves item 1 (with an item difficulty of 0.0).

```
> exp(1 - 0) / (1 + exp(1 - 0))
[1] 0.7310586
```

The probability that the student will solve item 2 is 50%. Lastly, the probability that the student will not solve item 3 is 37% (note: the probability of not solving an item is `1 / exp(1 - 0)`). Together, this particular response pattern (correct, correct, incorrect) entails a joint probability of `0.73 * 0.50 * 0.37 = 0.14`.

In practice, the student's ability and the item difficulties are not known in advance and must be estimated from the responses. What if the student's ability level is higher, at 2.0? The probability is no longer .14, it becomes .32. This higher probability suggests that, given the item difficulty parameters, a student with an ability level of 2.0 is actually more likely to produce this response pattern than a student with an ability level of 1.0. What MLE does is analogous to trying many ability levels and item difficulties until it finds the parameters that maximize the probability of the response pattern.

If done by hand, this trial-and-error process can quickly become tedious and error prone. To automate this process efficiently and accurately, one needs to define a general formula to handle different response patterns. Figure 2 uses such a formula to plot the likelihood of a response pattern of `c(1, 1, 0)` against hypothetical ability levels from −2 to 7.

Just from looking at Figure 2, it seems that an ability level near 2.0 is most likely to produce the response pattern, given an assumed set of item difficulties.

Figure 2 is derived from the likelihood function in Equation 3 below. The equation calculates the likelihood of a data pattern given a set of parameters.

$$L\left(x_{s1}, x_{s2}, \ldots, x_{si} \mid \theta_s, \beta_i\right) = \prod_{i=1}^{I} P\left(\theta_s, \beta_i\right)^{x_{si}} Q\left(\theta_s, \beta_i\right)^{\left(1-x_{si}\right)}. \quad (3)$$

The left-hand side indicates that, for a given examinee s, the likelihood of a response pattern $x_{s1}, x_{s2}, \ldots, x_{si}$ depends on the examinee's ability and the difficulty of the items. Assuming that item 1 has a difficulty of 0 (as we did previously), and if examinee s answered item 1 correctly, then $x_{s1} = 1$ and the equation for $i = 1$ becomes

$$P(\theta_s, \beta_1)^1 Q(\theta_s, \beta_1)^0 = P(\theta_s, 0) = \exp(\theta_s - 0)/[1 + \exp(\theta_s - 0)].$$

When repeated for the remaining items, the overall likelihood of the response pattern (correct, correct, incorrect) can be determined. To produce Figure 2, the following R code plots the likelihood of the responses as a function of the examinee's hypothetical ability level, as shown at the bottom of the page.

The vertical line roughly marks a possible optimal ability level estimate for this examinee. A precise estimate of the examinee's ability level can be obtained by applying the `mle()` function in R to the likelihood function, coded here in one simple line:

```
jpr[s] <- prod(pr(th-b)^resp
          * (1-pr(th-b))^(1-resp))
```

The next section will describe a quantitative solution using `mle()`.

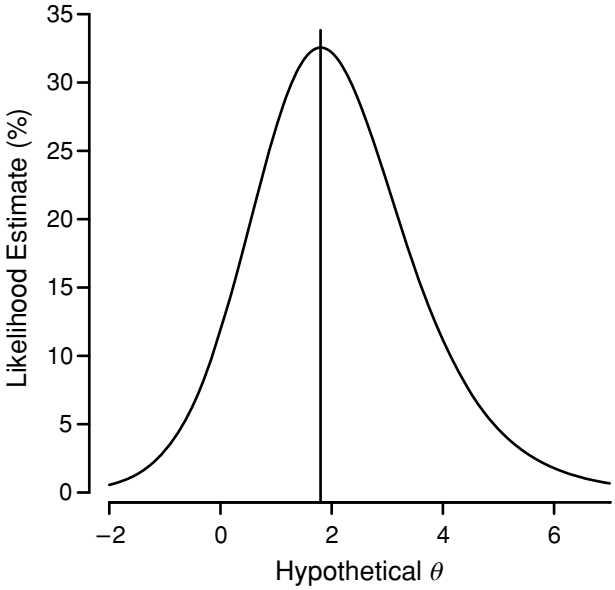

Figure 2. The likelihood function for the hypothetical three-item test.

```
theta <- seq(-2, 7, length=100) # hypothetical ability levels
resp <- c(1, 1, 0)              # responses to the 3 problems
b <- c(0, 1, 2)                 # hypothetical item difficulty
pr <- function(x) { exp(x) / (1 + exp(x)) } # inverse logit(x)
jpr <- numeric(length(theta))   # likelihood to be calculated
for (s in 1:length(jpr)) {
    th <- rep(theta[s], 3)
    jpr[s] <- prod(pr(th - b)^resp * (1- pr(th -b))^(1-resp))
    }
plot(theta, jpr, type='l')      # plots figure 2
abline(v = 1.8)                 # adds a vertical line at 1.8
```

```
prR <-
  function(theta = 2) { - sum(resp * log(exp(theta - b) /
  (1+exp(theta -b))) + (1 - resp) * log(1/(1+exp(theta - b))))
  }
```

5. Maximum Likelihood Estimation by `mle()` in R

Estimating the parameters by visual inspection is imprecise and error prone, especially when there are multiple items and many respondents. This section reevaluates the solution previously obtained by visualization. This time the quantitative `mle()` method designed to carry out MLE estimation by a precise numeric method is used.

The likelihood function described in Equation 3 is not yet suitable for the quantitative analysis performed by the `mle()` function. The `mle()` function works only with the "negative log likelihood" function, which can be derived by taking the natural logarithm of both sides of the likelihood function.

$$-\ln L\left(x_{s1}, x_{s2}, \ldots, x_{si} \mid \theta_s, \beta_i\right) = -\sum_{i=1}^{I} x_{si} P\left(\theta_s, \beta_i\right) + Q\left(\theta_s, \beta_i\right). \quad (4)$$

The negative log likelihood function is first defined in a customized function, then entered into `mle()` for further analysis. Note that `theta = 2` instructs the `mle()` function to begin the iterative search at 2.0, as shown at the top of the page.

The `mle()` function finds the `theta` parameter that minimizes the value of the negative log-likelihood function, assuming the response pattern `(resp <- c(1, 1, 0))` and the item difficulty `(b <- c(0, 1, 2))`, as shown at the bottom of the page.

The last statement, `coef(fit)`, is a generic function that extracts coefficients from model objects. It shows that the most probable ability level for this examinee is approximately 1.80.

```
theta
1.802945
```

This output confirms the earlier observation in Figure 2 that the student's most probable ability level is near 2. These calculations are straightforward. A user only has to identify the negative log-likelihood function and enter it into R.

This example is far simpler than real Rasch models. In the next section, the `mle()` function is used again to analyze 35 children's 18-item dichotomous Knox Cube Test results (Wright & Stone, 1979). The basic procedure remains mostly the same; this time, however, item difficulty and latent trait are estimated jointly in more complicated iterations.

6. Joint Maximum Likelihood Estimation Using `mle()` in R

The previous section explains how to estimate the ability of a single examinee when the item difficulty parameters are already known. The `mle()` function is also capable of determining item difficulty parameters when given the values of the examinee's ability levels. The JML method in many Rasch modeling software packages does this in four iterative steps:

1. A set of item difficulty parameters are assumed. These assumed item difficulty parameters are called *provisional parameters* because they simply provide a starting point. Typically, all items are assumed, at first, to have a difficulty of 0.0.

2. The provisional item difficulty parameters are entered into `mle()` to calculate trait levels of all examinees. These trait levels are also provisional.

3. The examinees' provisional trait levels are entered again into the `mle()` function to update the earlier provisional item difficulty parameters. The updated item difficulty parameters are again used to find a new set of trait levels.

4. Steps 2 and 3 are repeated as many times as needed. After each iteration, the estimated item difficulties and trait levels successively improve. Steps 2 and 3 are repeated until the item difficulty parameters no longer change after new iterations (in technical terms, this is when they "converge").

The raw data must be prepared before they are suitable for the JML method using `mle()` (data reproduced in the Appendix). Data must be entered into a matrix in R. Additionally, uninformative rows and columns of the data matrix are deleted. At the bottom of the data matrix is a row showing the percentages of students who answer each item correctly. Items 1, 2, and 3 are the easiest because all students can solve them. Item 18 is the hardest because no student can solve it. In JML, items like these provide no useful information because they cannot help distinguish between students who possess high ability levels and students who possess low ability levels. Therefore, these questions are typically omitted from the calculation. Other more sophisticated methods, such as the marginal

```
if (! exists("mle")) library(stats4)  # load stats4 if not present
resp <- c(1, 1, 0)
b <- c(0, 1, 2)
fit <- mle(prR)      # runs mle()
coef(fit)            # prints the value of the fitted theta
```

```
# again, define the negative log likelihood function for mle()
prR <- function(theta = 2) { - sum(resp * log(exp(theta - b) /
      (1+exp(theta -b))) + (1 - resp) * log(1/(1+exp(theta -b))))}
rM <- exam1[, 6:19]    # excludes name, sex, and items 1, 2, 3, and 18
rM <- rM[-35, ]        # excludes Helen
```

maximum likelihood estimation (MML) and the conditional maximum likelihood estimation (CML) do not have this limitation (Embretson & Reise, 2000, pp. 210–225). Students who get all or none of the items right are omitted. Helen fails all 18 items. Her scores therefore carry no useful information in distinguishing hard items from easy ones. This process of trimming the raw data is described in detail in Wright and Stone (1979).

The R code at the top of the page carries out the preparations. The `rM <- exam1[, 6:19]` statement means that the data matrix `rM` contains only columns 6–19 of the raw data, excluding respondent name, sex (columns 1 and 2), and uninformative Items 1, 2, 3, and 18 (columns 3, 4, 5, and 20). The `rM <- rM[-35, ]` excludes Helen's responses.

The remainder of this section describes how to implement the four steps in JML Rasch analysis in R. The four steps are separated, and their corresponding segments of R code are explained individually. An experienced researcher can combine the four separate segments of R code in one expedited Rasch analysis. Steps 1 and 2 calculate the provisional parameters. Steps 3 and 4 are included in a long iterative loop. Provisional parameters are updated after each iteration, until the estimates stabilize. Below, Steps 1 and 2 calculate the preliminary person trait from the provisional beta parameters (all set to 0), as shown at the bottom of the page.

The `apply()` function takes the data matrix `rM` and applies a simple `mle()` fit to each examinee's data, across dimension 1 of the data matrix. The `assign()` statement within `apply()` ensures that the interim data created within `apply()` are intact. The `.GlobalEnv` parameter tells R that these interim data are accessible to all other functions within `apply()`.

In Step 3, the examinees' provisional trait levels are entered again into the `mle()` function to update the earlier provisional item difficulty parameters. Step 3 is slightly different from Step 1. The negative log likelihood function, `prRi()`, estimates the item difficulty, assuming that the person trait levels are already known. The call to `mle()` is supplied with the alternate log likelihood function `prRi`, as shown at the top of the next page.

Finally, Steps 2 and 3 are repeated in Step 4 until the estimated parameters converge. Step 4 is an iterative loop controlled by a `while()` statement, which contains two exit points. The first, and the most important, exit point checks for parameter convergence. The convergence check stops the `while()` loop if the differences between the provisional and the updated item parameters are less than five decimal points (`! all((item - b) < 10^-5)`). The second exit point guards against program lock-up due to nonconvergent parameter estimates. Thus, the `while()` loop repeats 55 times or until all item difficulty parameters converge, whichever is reached first, as shown at the bottom of the next page.

The end of the iterative process concludes the main Rasch analysis using R MLE. The item difficulties are in the vector `b` and the trait levels are in the vector `theta`.

The MLE method presented here dates back to 1969 (Wright & Panchapakesan, 1969). It is referred to as the *unconditional* solution (UCON; Wright, 1988; Wright & Douglas, 1977; Wright & Masters, 1982). Wright and Douglas (1977) reported that the UCON method is prone to slightly biased estimates. They suggested a correction factor for item difficulty estimates. To correct the bias, the `b` vector is weighted by $(K - 1)/K$, where K is the number of test items. In R this can be calculated by `b * ( (length(b) - 1) / length(b) )`. The bias becomes very small if the test is long enough and the sample size is adequate. Wright (1988; see also Wright & Douglas, 1977) estimated a minimal test length of 20 items. Jansen, Van den Wollenberg, and Wierda (1988) suggested a length of 10 items. A discussion of this bias correction method is provided by MINISTEP in the help files (entitled "Estimation bias correction—warnings").

7. Rasch Analysis as a Conditional Logit Model

Before discussing the goodness-of-fit statistics, it is worth noting that Rasch analysis can be carried out by popular packages such as SAS and Stata with their conditional logit modeling routines. For example, Ender (2003) used Stata's `clogit` subroutine to illustrate how to carry out dichotomous Rasch analysis with Stata. A similar conditional logit module is also available in R.

```
# 1. Set provisional betas equal to zero
b <- rep(0, dim(rM)[2])         # provisional betas all set to 0
# 2. Find preliminary person trait from provisional beta parameters
theta <- apply(rM, 1, function(x) {
    assign("resp," x, env = .GlobalEnv) # make resp accessible to mle()
    fit <- mle(prR)
    coef(fit)   })
print(theta)   # updated trait levels across individual examinees
```

```
# First define the negative log likelihood function for item
prRi <- function(b = 0) {
     - sum(resp * log(exp(theta - b) /(1+exp(theta - b))) +
     (1 - resp) * log(1/(1+exp(theta - b))) )
}
# 3. Use updated trait levels to estimate item difficulty
item <- apply(rM, 2, function(x) {
    assign("resp," x, env = .GlobalEnv)
    fit <- mle(prRi)
    coef(fit) })
print(item)
```

The `clogit()` function in R only produces estimates of item difficulty (so does the conditional logit subroutine in Stata, as seen in Ender, 2003). The output does not contain estimates of the examinees' ability levels. This limits the application of a conditional logit model in Rasch analysis. A researcher who develops a quality-of-life instrument often wants to find out if some patients report more discomfort than do others. Nevertheless, output from `clogit()` is helpful for understanding the interval scale properties of parameters in a Rasch model. In the next section, the results obtained from the `clogit()` function in R are summarized for comparison.

8. Results Comparison

In this section, the results obtained from R, including the parameter estimates and the goodness-of-fit statistics, are compared with the solutions generated by MINISTEP and the output of a conditional logit model. The purpose of the comparisons is to verify the accuracy of the solutions generated by R.

Verification of parameter estimates. The item difficulty parameters from all three methods (R `mle()`, MINISTEP, and R `clogit()`) are listed in Table 1 for comparison.

Table 1 shows that the numbers derived from the respective methods are not identical. However, these differences are due to scaling and do not necessarily affect how the results are interpreted. Bond and Fox (2001, p. 45) state that many Rasch analysis software programs make adjustments to the estimated parameters—for example, the subtraction of the mean of all estimated item difficulty parameters. Without the source code of MINISTEP, however, this cannot be verified for certain.

The logit scale in Rasch modeling is an interval scale. Generally, interval scales can be transformed into different units with different zeros and still measure the same phenomenon (such as the Fahrenheit-to-Celsius conversion). Another important property of an interval scale is that, although ratios of intervals are meaningful, ratios of scale quantities are not. For example, on all three sets of results, Items 6 and 9 are equally more difficult than Item 4. Items 15, 16, and 17 are also equally difficult relative to Item 14.

Susan and Rick (Examinees 7 and 24) have trait levels of 3.89, the highest of the group. They also have the highest total raw score, since they answered 11 of 14 questions correctly. Susan and Rick both have a probability of about 25% of solving Item 15 (calculated by `exp(3.889 -`

```
# 4. Iterate until the newly calculated item difficulty converges
iter <- 1   # begin with the first iteration
while( !all((item - b) < 10^-5) & iter <= 55)
  {
  # replace previous provisional betas with newly fitted values
  b <- item - mean(item) # recenter b, see Wright & Douglas (1977), p.284
  # 2. Calculate person trait levels again
  theta <- apply(rM, 1, function(x) {
    assign("resp," x, env = .GlobalEnv)
    fit <- mle(prR)
    coef(fit)   })
  print(round(theta, 3)) # print estimates per iteration
  # 3. Estimate item difficulty from updated trait levels
  item <- apply(rM, 2, function(x) {
    assign("resp," x, env = .GlobalEnv)
    fit <- mle(prRi)
    coef(fit) })
  iter <- iter + 1    # iteration increments by 1
  }
```

```
V.0 <- (0 - E)^2 * (1 - E) # when k = 0; M & W, p.113
V.1 <- (1 - E)^2 * E       # when k = 1;
V <- V.0 + V.1
```

Table 1
Results of Rasch Analysis Obtained From R, MINISTEP, and Conditional Logit Model

Item	R `mle()`	MINISTEP	Conditional Logit
4	−4.552	−4.40	0.000
5	−3.969	−3.83	0.514
6	−3.505	−3.38	0.923
7	−3.969	−3.83	0.514
8	−2.439	−2.35	1.875
9	−3.505	−3.38	0.923
10	−1.629	−1.57	2.598
11	0.828	0.79	4.514
12	2.327	2.24	5.792
13	2.028	1.95	5.537
14	3.500	3.37	6.825
15	4.962	4.80	8.095
16	4.962	4.80	8.095
17	4.962	4.80	8.095

`4.962) / (1 + exp(3.889 - 4.962))`). This probability is similar to the one calculated from the output of MINISTEP [`exp(3.73 - 4.80) / (1 + exp(3.73 - 4.80))`]. This is a rudimentary method to verify that R `mle()` generates trustworthy results. The results from conditional logit model cannot be verified this way.

Goodness of fit. Rasch software programs such as MINISTEP and QUEST print out the "infit" and "outfit" goodness-of-fit statistics along with the estimated parameters (Masters & Wright, 1997). These statistics can be used to further examine the results obtained from R. They can also be used for instructional purposes. Below is a brief summary of what these fit statistics are and how they can be derived from the `theta` and `b` vectors. A thorough review of various fit statistics can be found in Karabatsos (2003).

The formulas for calculating these goodness-of-fit statistics involve calculating the standardized residual mean squares between the observed and the expected responses. This, in turn, involves calculating the expected responses (predictions made by the Rasch model) and comparing them with the observed data (in the matrix `rM`). The expected responses are (after Masters & Wright, 1997, p. 113):

$$E_{si} = \sum_{k=0}^{m_i} kP_{sik},$$

where k is between 0 and the maximum possible score on item i, m_i. In a dichotomous model, k can be 0 or 1, which reduces the formula to $E_{si} = P_{si}$. It does not involve summing over multiple response categories k.

Some of R's convenient features can be utilized to simplify the calculation. First, the user-defined `ilog()` function is used again to predict the probability of a correct answer. Next, a built-in function `outer()` is called to apply the Rasch model to every combination of `theta` and `b`. The end product, `E`, is a matrix containing the expected probability of correct answers for all examinees across all items:

```
E <- outer(theta, b, function(x, y) {
          ilog(x - y) })
```

The differences between the observed and the expected responses are standardized by their corresponding variances. The variances are

$$V\left[x_{si}\right] = \sum_{k=0}^{m_i} \left(k - E_{si}\right)^2 P_{sik}.$$

It may be helpful to think of this as adding two layers of matrices together, one for $k = 0$ and the other for $k = 1$, as shown at the top of the page.

Finally, the standardized difference between examinee s's observed and expected response to item i is

$$z_{si} = \frac{\left(x_{si} - E_{si}\right)}{\sqrt{V\left[x_{si}\right]}}.$$

The R statement closely resembles the formula `Z <- (rM - E) / sqrt(V)`. The matrix `Z` can be used to derive the "outfit" and "infit" statistics of all items.

The outfit of item i is

$$u_i = \sum_s^N \frac{z_{si}^2}{N}.$$

A call to the function `apply()` produces the "outfit" statistics of all items, as shown at the bottom of the page.

The calculation of the infit statistics for all items

$$u_i = \frac{\sum_s^N V_{sj} z_{si}^2}{\sum_s^N V_{si}}$$

requires weighting the standardized residuals by their variances, thus lessening the impact of unexpected responses (Bond & Fox, 2001; Masters & Wright, 1997), as shown at the top of the next page.

R produces very similar fit statistics as the ones obtained from MINISTEP (not shown here).

R graphics functions (e.g., `plot()`, `barplot()`, and `hist()`) can be used to produce visual aids such as the item-person map and the item pathway produced by the BLOT package (Bond & Fox, 2001). Space limitations

```
> round(apply(Z, 2, function(x) { mean(x^2) } ), 2) # outfit
  q4   q5   q6   q7   q8   q9  q10  q11   q12  q13  q14  q15  q16  q17
0.35 0.53 1.03 2.48 0.44 0.21 0.86 0.81 1.14 0.37 1.67 0.11 0.11 0.11
```

```
> round(apply(V * Z^2, 2, sum) / apply(V, 2, sum), 2) # infit
  q4   q5   q6   q7   q8   q9  q10  q11  q12  q13  q14  q15  q16  q17
0.92 1.07 1.21 1.38 0.60 0.63 1.10 1.12 1.21 0.71 1.64 0.77 0.77 0.77
```

prevent a more detailed summary of their use in Rasch analysis.

The `mle()` function belongs to an object class called "mle-class" in R. There are many special methods in R that can be used to retrieve useful information from the object `fit`. For example, `vcov(fit)` retrieves the standard errors of the fitted parameters and `summary(fit)` prints a detailed summary of the fitted model, including the log likelihood information of the MLE procedure. Other related methods are listed in `help("mle-class")`.

Conclusion

The application of MLE in dichotomous Rasch modeling is a challenging topic for instructors, in part because it relies on iterative computations often not accessible to student novices. R makes it easier to explain MLE because the R syntax closely resembles the mathematical computations it performs. This simplifies the transformation from mathematical formulas in a Rasch model to the corresponding statistical computations. Pedagogical advantages like this (Embretson & Reise, 2000, chap. 8) makes R a suitable statistical language in teaching the technical details in carrying out Rasch analysis. Moreover, because of its object-oriented programming methods and collaborative software development environment, R continues to incorporate codes and packages from contributors worldwide. This facilitates the accumulation of knowledge and tools that will contribute to the further advancement of open-source IRT.

Many advanced IRT solutions have already been developed in R and S/Splus, such as the Bayesian IRT methods (Fox, 2002; Karabatsos & Batchelder, 2003) and parameter estimation by Markov chain Monte Carlo simulations (Fox, 2002; Karabatsos & Batchelder, 2003; Patz & Junker, 1999a, 1999b). The MCMCIRT package by Patz and Junker (1999b) for Splus is available online (lib.stat.cmu.edu/S/mcmcirt.shar). The LTM package by Rizopoulos (2005) was recently made available on the R Web site. These highly sophisticated and more powerful tools contain computations more complicated than the UCON method. Thus, student novices may use the present tutorial as a springboard to these more powerful IRT solutions in R and S/Splus.

The current R commands have limitations. For example, JML still cannot estimate the parameters of items where everyone passes or fails. Nor can JML be used to estimate the trait levels of examinees who get full marks or fail completely. However, the development of open-source IRT software is an ongoing process and the present tutorial aims to build on these foundations. The present R code can be tailored to incorporate advanced features, perhaps through extending the present R code, to apply more sophisticated methods such as MML or CML to advanced models (e.g., the 2PL and 3PL models in Embretson & Reise, 2000).

REFERENCES

Allbritton, D. W. (2003). Using open-source solutions to teach computing skills for student research. *Behavior Research Methods, Instruments, & Computers*, **35**, 251-254.

Becker, R. A., Chambers, J. M., & Wilks, A. R. (1988). *The new S language: A programming environment for data analysis and graphics*. Pacific Grove, CA: Wadsworth.

Bond, T. G., & Fox, C. M. (2001). *Applying the Rasch model: Fundamental measurement in the human sciences*. Mahwah, NJ: Erlbaum.

Bradstreet, T. (1996). Teaching introductory statistics courses so that nonstatisticians experience statistical reasoning. *American Statistician*, **50**, 69-78.

Dalgaard, P. (2002). *Introductory statistics with R*. New York: Springer.

de la Torre, J., Stark, S., & Chernyshenko, O. S. (2006). Markov chain Monte Carlo estimation of item parameters for the generalized graded unfolding model. *Applied Psychological Measurement*, **30**, 216-232.

Embretson, S. E., & Reise, S. P. (2000). *Item response theory for psychologists*. Mahwah, NJ: Erlbaum.

Ender, P. B. (2003). *Applied categorical & nonnormal data analysis: A Rasch model example*. Retrieved August 2004 from www.gseis.ucla.edu/courses/ed231c/notes2/rasch.html.

Fox, J.-P. (2002). *Multilevel IRT software manual: In Splus 6 for Windows*. Retrieved March 2005 from users.edte.utwente.nl/Fox.

Jansen, P. G., Van den Wollenberg, A. L., & Wierda, F. W. (1988). Correcting unconditional parameter estimates in the Rasch model for inconsistency. *Applied Psychological Measurement*, **12**, 297-306.

Karabatsos, G. (2003). Comparing the aberrant response detection performance of thirty-six person-fit statistics. *Applied Measurement in Education*, **16**, 277-298.

Karabatsos, G., & Batchelder, W. H. (2003). Markov chain estimation for test theory without an answer key. *Psychometrika*, **68**, 373-389.

Linacre, J. M. (2004). *WINSTEPS Rasch measurement computer program*. Chicago: Winsteps.com.

Maindonald, J., & Braun, J. (2003). *Data analysis and graphics using R*. Cambridge: Cambridge University Press.

Marasinghe, M. G., Meeker, W. Q., Cook, D., & Shin, T. S. (1996). Using graphics and simulation to teach statistical concepts. *American Statistician*, **50**, 342-351.

Martin, A. D., & Quinn, K. M. (2006). Package MCMCpack. Retrieved October 18, 2006, from mcmcpack.wustl.edu/wiki.

Masters, G. N., & Wright, B. D. (1997). The partial credit model. In W. J. van der Linden & R. K. Hambleton (Eds.), *Handbook of item response theory* (pp. 101-121). New York: Springer.

Mills, J. D. (2002). Using computer simulation methods to teach statistics: A review of the literature. *Journal of Statistics Education*, **10**(1). Retrieved September 2004 from www.amstat.org/publications/jse/v10n1/mills.html.

Patz, R. J., & Junker, B. W. (1999a). Applications and extensions of MCMC in IRT: Multiple item types, missing data, and rated responses. *Journal of Educational & Behavioral Statistics*, **24**, 342-366.

Patz, R. J., & Junker, B. W. (1999b). A straightforward approach to Markov chain Monte Carlo methods for item response models. *Journal of Educational & Behavioral Statistics*, **24**, 146-178.

Rasch, G. (1980). *Probabilistic models for some intelligence and attainment tests*. Chicago: University of Chicago Press. (Original work published 1960)

R Development Core Team (2004). *R language definition*. Retrieved March 2005 from cran.r-project.org/doc/manuals/R-lang.pdf.

Rizopoulos, D. (2005). *Latent trait models under IRT*. Retrieved March 2005 from cran.us.r-project.org.

Roberts, J. S., & Huang, C. W. (2003). GGUMLINK: A computer program to link parameter estimates of the generalized graded unfolding model from item response theory. *Behavior Research Methods, Instruments, & Computers*, **35**, 525-536.

Sheu, C.-F., Chen, C.-T., Su, Y.-H., & Wang, W.-C. (2005). Using SAS PROC NLMIXED to fit item response theory models. *Behavior Research Methods*, **37**, 202-218.

Venables, W. N., & Ripley, B. D. (2000). *S programming*. New York: Springer.

Waller, N. (1998). *LINKDIF: An S-PLUS routine for linking item parameter and calculating IRT measures of differential functioning of items and tests*. Retrieved March 2005 from freeirt.free.fr/database/detailpgm.php?cond=idprogram=27.

Wright, B. D. (1988). *The efficacy of unconditional maximum likelihood bias correction* (MESA Research Memorandum No. 45). Available at www.rasch.org/memo45.htm.

Wright, B. D., & Douglas, G. A. (1977). Best procedures for sample-free item analysis. *Applied Psychological Measurement*, **1**, 281-295.

Wright, B. D., & Masters, G. (1982). *Rating scale analysis*. Chicago: Mesa.

Wright, B. D., & Panchapakesan, N. A. (1969). A procedure for sample-free item analysis. *Educational & Psychological Measurement*, **29**, 23-48.

Wright, B. D., & Stone, M. H. (1979). *Best test design*. Chicago: Mesa.

APPENDIX

An electronic copy of the raw data can be found in the MINISTEP software package. The "exam1.txt" file contains the raw data and the MINISTEP control statements. The data portion of the file is entered into a text file called "exam1.dat" (see below). The following R statement is used to incorporate the raw data into an R data object.

```
> exam1 <- read.csv(file="exam1.dat," sep="", header=T, row.names=NULL)
```

name	sex	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
Richard	M	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Tracie	F	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Walter	M	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0
Blaise	M	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
Ron	M	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
William	M	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Susan	F	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0
Linda	F	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Kim	F	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Carol	F	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
Pete	M	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Brenda	F	1	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0
Mike	M	1	1	1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0
Zula	F	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
Frank	M	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
Dorothy	F	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0
Rod	M	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0
Britton	F	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0
Janet	F	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
David	M	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0
Thomas	M	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0
Betty	F	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Bert	M	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0
Rick	M	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0
Don	M	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Barbara	F	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Adam	M	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Audrey	F	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0
Anne	F	1	1	1	1	1	1	0	0	1	1	1	0	0	1	0	0	0	0
Lisa	F	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
James	M	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Joe	M	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
Martha	F	1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
Elsie	F	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0
Helen	F	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		1	1	1	.91	.88	.86	.88	.77	.86	.69	.34	.17	.20	.09	.03	.03	.03	0.0

(Manuscript received September 3, 2004;
revision accepted for publication April 27, 2005.)