Multidimensional Item Response Theory Workshop in R (Day 2)

Phil Chalmers

York University

February 11, 2015

1. Latent trait densities and customization
2. Mixed Effects
3. Latent trait prediction
4. Computerized Adaptive Tests
5. mirtCAT package
6. Closing

Organization

Workshop 2

Focus on this Workshop is more about two additional purpose of using IRT: **explaining variation** and **scoring the test**. Also demonstrates some flexible customizations approaches with the **mirt** package.

- Non-traditional latent densities and customizations with **mirt**
- Modeling explanatory fixed and random item- and person-covariates to explain test variation
- Discuss various algorithms for scoring the test, and how they relate to various test properties
- Demonstrate how test scoring methods integrate with computerized adaptive testing (CAT) applications
- Hands-on creation of real-time (and off-line) CAT and multidimensional CATs with the **mirtCAT** package

Latent trait densities and customization

Latent trait densities and customization

Latent trait densities and customization

In Workshop 1,

- We focused on using the default `itemtype` arguments to create standard relatively standard IRT models that **mirt** uses natively
- We also used the default latent density for estimating the models (the multivariate normal density)

However, it is possible to customize both of these inputs using user-defined `R` functions. These can offer a very flexible way to use **mirt**'s estimation engine over and above what I've had time to code internally.

Latent density

The `technical` list argument to `mirt()` accepts several arguments for modifying the integration grid and functional density. These include:

- `customTheta` – a custom θ grid, in matrix form, used for integration. If not defined, the grid is determined internally based on the number of `quadpts`
- `customPriorFun` – a custom function used to determine the normalized density for integration in the EM algorithm. Must be of the form `function(Theta, Etable){...}`, and return a numeric vector with the same length as number of rows in Theta. `Etable` is the pseudo-complete data-table in the E-step of the EM algorithm

The **sirt** package, written by Alexander Robitzsch, has some wonderful examples of how to use these functions for creating specialized Rasch models.

Latent density example

Create a latent density function to be uniform between -6 and 6 with 100 equally spaced integration nodes.

```
den <- function(Theta, Etable) dunif(Theta, min = -6, max = 6)
Theta <- matrix(seq(-6, 6, length.out = 100))
custom_mod <- mirt(Science, 1, technical =
    list(customPriorFun = den, customTheta = Theta))
```

```
coef(custom_mod, simplify=TRUE)$items
```

```
##              a1    d1    d2     d3
## Comfort 0.257 4.721 2.578 -1.399
## Work    0.330 2.888 0.927 -2.204
## Future  0.739 5.449 2.678 -2.306
## Benefit 0.275 3.259 0.997 -1.614
```

Empirical histogram

A special type of customized density is included based on information in the `Etable` argument, called the *empirical histogram*. This approximates the latent density at each step of the EM by summing across the E-step data-table to create a non-parametric smoothing of the density.

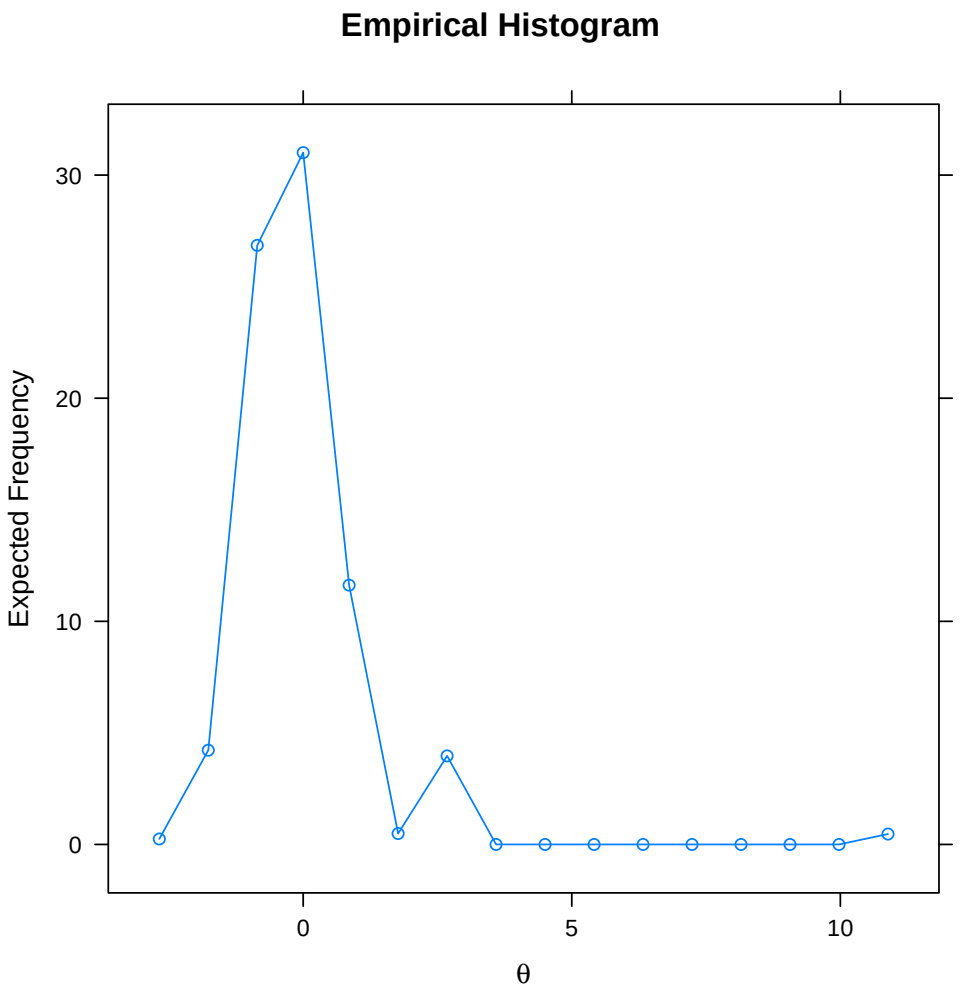
$$h_d(\theta) = \sum_{i=1}^{C} E_i$$

$$g_d(\theta) = \frac{h_d(\theta)}{\sum_{d=1}^{D} h_d(\theta)}$$

Latent density is approximated at the same time as estimating the item parameters; this generally causes instability in the estimates, and the model may have difficulty converging (however, given that it is an exploratory technique, that is not really an issue).

Empirical histogram

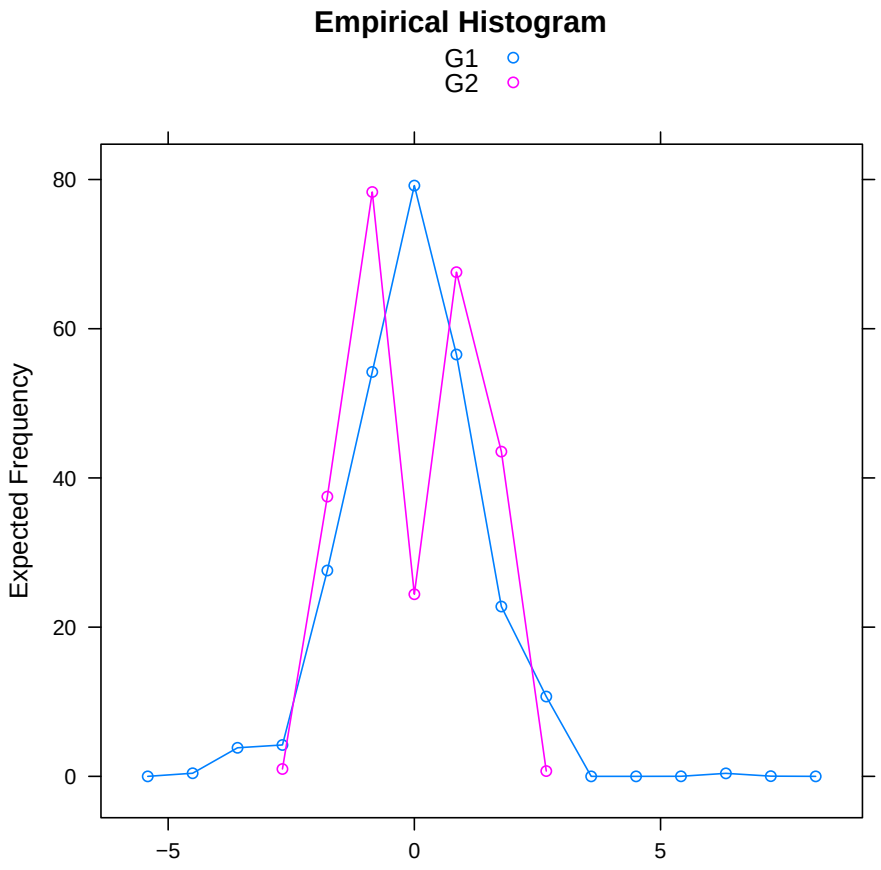
```
eh <- mirt(SAT12_scored, 1, empiricalhist = TRUE)
plot(eh, type = 'empiricalhist')
```



Empirical histogram

Also works in multiple group estimation (and DIF).

```
EH <- multipleGroup(dat, 1, group=group, empiricalhist = TRUE,
                    invariance = colnames(dat))
plot(EH, type = 'empiricalhist')
```



createItem()

IRT models not included in the **mirt** package can be build and passed to the estimation engine through `mirt()`, `multipleGroup()`, or `bfactor()`.

```
P.old2PL <- function(par, Theta, ncat){
     a <- par[1]; b <- par[2]
     P1 <- 1 / (1 + exp(-1*a*(Theta - b)))
     cbind(1-P1, P1)
}
x <- createItem('old2PL', par = c(a = .5, b = -2), est =
                    c(TRUE, TRUE), P = P.old2PL)
dat <- expand.table(LSAT7)
custom_mod <- mirt(dat, 1, itemtype = 'old2PL', verbose = FALSE,
                   customItems = list(old2PL = x))
coef(custom_mod, simplify=TRUE)$items
```

```
##              a      b
## Item.1 0.989 -1.877
## Item.2 1.081 -0.747
## Item.3 1.708 -1.057
## Item.4 0.766 -0.634
## Item.5 0.737 -2.516
```

Further customizations: starting values

Starting values can be found by passing `pars = 'values'` to return a `data.frame` object. This object is customizable and can be passed back into the function. If the model has already been estimated `mod2values()` can be used (useful for providing better starting values).

```
values <- mirt(lsat, 1, pars = 'values')
values[1:2,]
```

```
##   group   item class name parnum    value lbound ubound  est prior.type
## 1   all Item.1  dich   a1      1 0.851000   -Inf    Inf TRUE       none
## 2   all Item.1  dich    d      2 1.859747   -Inf    Inf TRUE       none
##   prior_1 prior_2
## 1     NaN     NaN
## 2     NaN     NaN
```

```
values[1,'value'] <- 1  #change start value for parameter 1...
values[1,'est'] <- FALSE #...and do not estimate it
mod <- mirt(lsat, 1, pars = values)
```

Helper function have also been constructed in the `sirt` package (see `?sirt::mirt.specify.partable`) to modify this `data.frame`.

Mixed Effects

Mixed Effects IRT

Mixed effects (i.e., explanatory) models

Often in IRT we want to know the effect of including additional information not present in the test (i.e., external covariates), and if these could help explain the observed response patterns. A large literature with Rasch modeling has been focused on this area, however extending the framework to include models other than Rasch has been slow.

- Latent regression models are possible in **mirt** with two functions: `mirt()` given a `formula` input and regression data, and `mixedmirt()`, which is a much more general function for explanatory IRT models with fixed and random effects

Latent regression models

The latent regression model attempts to decompose the θ effects into fixed-effect components to *explain* differences between individuals in different populations.

$$\Theta = \mathbf{X}\beta + \epsilon,$$

where $\mathbf{X}$ is a design matrix containing person-level covariate data (e.g., gender indicated by 0s and 1s, or continuous variables such as age), β is a vector of fixed effect regression parameters, and ϵ is a residual with the distribution $MVN(\mathbf{0}, \mathbf{\Sigma})$

- If the IRT models are not from the Rasch family, various item-specific slopes must be fixed to a constant in order to identify the $\mathbf{\Sigma}$ elements

Latent regression models

Latent regression simulation example with $\beta = [0.5, -1]$, and $\epsilon \sim N(0, 0.25)$.

```
beta <- c(0.5, -1); N <- 500
X1 <- rnorm(N); X2 <- rnorm(N)
covdata <- data.frame(X1 = X1, X2 = X2)
Theta <- matrix(beta[1] * X1 + beta[2] * X2 +
                    rnorm(N, sd = sqrt(0.25)))
a <- matrix(1, 10); d <- matrix(rnorm(10))
dat <- simdata(a, d, N, itemtype = 'dich', Theta = Theta)

#latent regression with X1 and X2 as predictors of Theta
mod1 <- mirt(dat, 1, 'Rasch', verbose = FALSE,
             covdata = covdata, formula = ~ X1 + X2)
coef(mod1, simplify = TRUE)
```

Latent regression models

```
##         a1      d g u
## Item_1   1  0.157 0 1
## Item_2   1 -1.178 0 1
## Item_3   1  1.379 0 1

...

## $groupPars
## $groupPars$means
## MEAN_1
##      0
##
## $groupPars$cov
##       F1
## F1 0.203
##
##
## $lr.betas
##                   F1
## (Intercept)  0.000
## X1           0.565
## X2          -0.995
```

Mixed-effects modeling

Purpose of mixed-effects modeling is to include continuous or categorical item and person predictors into the model directly by way of the intercept parameters. An example of including a fixed effect predictor into the model at the person level would be the inclusion of 'Gender', where an indicator coding is used to change the expected probability to:

$$P(x = 1; \boldsymbol{\theta}, \Psi, \beta_{male}) = g + \frac{(u - g)}{1 + \exp\left[-(\mathbf{a}'\boldsymbol{\theta} + d + \beta_{male}X)\right]}.$$

Notice here that θ is not directly decomposed and instead additional *intercepts* are modeled. This model is distinct from the latent regression model, which has the form

$$P(x = 1; \boldsymbol{\theta}, \Psi, \beta_{male}) = g + \frac{(u - g)}{1 + \exp\left[-(\mathbf{a}'\boldsymbol{\theta} + d\right]}.$$

where $\boldsymbol{\theta} = \beta_{male}X + \boldsymbol{\epsilon}$.

`mixedmirt()`

`mixedmirt()` in the **mirt** package was design to include fixed and random intercepts coefficients into the modeling framework directly.

- Effects organized to explain person effects (i.e., latent regression models), but also can explain variability in the test itself (i.e., explain why some items are more difficult than others, account for speeded effects, etc.)

For the M4PL model,

$$P = g + \frac{(u - g)}{1 + \exp\left(-(\boldsymbol{\Theta}\mathbf{a} + \mathbf{X}\beta + \mathbf{Z}\boldsymbol{\delta})\right)}$$

where similar terms are the same as in the latent regression model, and $\mathbf{Z}\boldsymbol{\delta}$ controls the random intercept terms.

Mixed-effects modeling

The equation

$$P = g + \frac{(u - g)}{1 + \exp\left(-(\boldsymbol{\Theta}\mathbf{a} + \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\delta})\right)}$$

can be even further broken down to form the latent regression model.

- $\boldsymbol{\Theta}$ can be decomposed into fixed and random components just like the intercepts: $\boldsymbol{\Theta} = \mathbf{V}\boldsymbol{\gamma} + \mathbf{W}\boldsymbol{\zeta} + \boldsymbol{\epsilon}$
- Hence, the above model can simultaneously capture item and person effects
- Generalizations of this equation are fairly simple to polytomous responses, however intercept designs require slightly more care

`mixedmirt()`

Conceptually, `mixedmirt()` will

- breakdown intercepts into more complicated crossed item by person designs (which, for Rasch models would result in the latent regression model as a special case)
- model random components in the intercepts (LLTM, multi-level IRT for Rasch model) rather than treating all intercepts as fixed effects. This allows intercept variability to be *explained*
- allow the θ values to be decomposed into fixed effects (random effects coming soon for more complete multi-level IRT!)

Estimation exclusively uses the MH-RM algorithm to obtain item and regression parameter estimates, therefore the observed information matrix can always be computed if standard errors/Wald tests are desired.

`mixedmirt()` syntax

Has the usual `data` and `model` arguments as before, however it also supports

- `covdata` – a `data.frame` of person-level covariate information, and
- `itemdesign` – a `data.frame` of item design based effects

The fixed effects are controlled with

- `fixed` – a standard R formula to decompose the intercept effects (e.g., `~ items + gender*IQ`). Corresponds to the $\mathbf{X}\beta$ effects
- `lr.fixed` – an R formula to decompose the latent trait(s). Corresponds to the $\mathbf{V}\gamma$ effects

`mixedmirt()`

Random effect terms have a syntax similar to the **nlme** package:

- `random` – an | separated R formula indicating random intercepts and slope effects for the intercepts (e.g., `~ 1 | group`) ($\mathbf{Z}\boldsymbol{\delta}$)
- `lr.random` – (*not yet supported*) an | separated R formula indicating random intercepts and slope effects for the trait ($\mathbf{W}\boldsymbol{\zeta}$)

By modeling variability with the random effects, fixed effects predictors can be used to 'explain' different sources of variation (e.g., why schools may be different, or why some items are more difficult than others). Hence, random effects are generally interpreted as 'residual variation'.

Syntax examples

Some syntax examples (notice that an `items` keyword is used to estimate intercepts for each item)

```
# Rasch latent regression model
mod1 <- mixedmirt(data, covdata, model,
                  fixed = ~ 0 + group + items)
# equivalently in this case
mod2 <- mixedmirt(data, covdata, model,
                  lr.fixed = ~ group, fixed = ~ 0 + items)

# 2PL latent regression model
mod3 <- mixedmirt(data, covdata, model, itemtype = '2PL',
                  lr.fixed = ~ group, fixed = ~ 0 + items)
# similar to above, but using intercepts instead
mod4 <- mixedmirt(data, covdata, model, itemtype = '2PL',
                  fixed = ~ 0 + group + items)
```

Syntax examples

```
# multilevel Rasch model (random group intercepts)
rmod <- mixedmirt(data, covdata, model = 1, fixed = ~ 0 + items,
                  random = ~ 1|group)
rmod1 <- mixedmirt(data, covdata, model = 1,
                  fixed = ~ 0 + group_means + items,
                  random = ~ 1|group)

# crossed random effects
rmod2 <- mixedmirt(data, covdata, model = 1,
                   random = list(~ 1|group, ~ 1|items))

# linear latent trait model (with residual)
itemdesign <- data.frame(itemorder = factor(c(rep('easier', 16),
                                              rep('harder', 16))))
LLTM <- mixedmirt(data, model = model, fixed = ~ 0 + itemorder,
                  random = ~ 1|items, itemdesign = itemdesign)
```

Latent trait prediction and related methods

Latent trait prediction and related methods

Latent trait prediction

Ability/latent trait prediction ($\hat{\theta}$) is often the focus of IRT analysis since we may wish to compare individual abilities along the continuum given their response patterns, or potentially compare scores between groups of individuals.

- Precision of $\hat{\theta}$ estimate depends on how much information the test provides (easy tests provide more accurate predictions for lower ability individuals)
- Bayesian methods, such as expected and maximum a posteriori estimation, are also available, and the strength of the prior will also contribute to the location and precision estimates (informative priors will shrink the estimates and standard errors accordingly)

ML ability example

Say that we wanted to estimate the ability of a subject who responded with the pattern $(1, 1, 0)$ to three items with parameters $a_1 = a_2 = a_3 = 1$, and $d_1 = 1$, $d_2 = 0$, and $d_3 = -1$. The likelihood/posterior function is

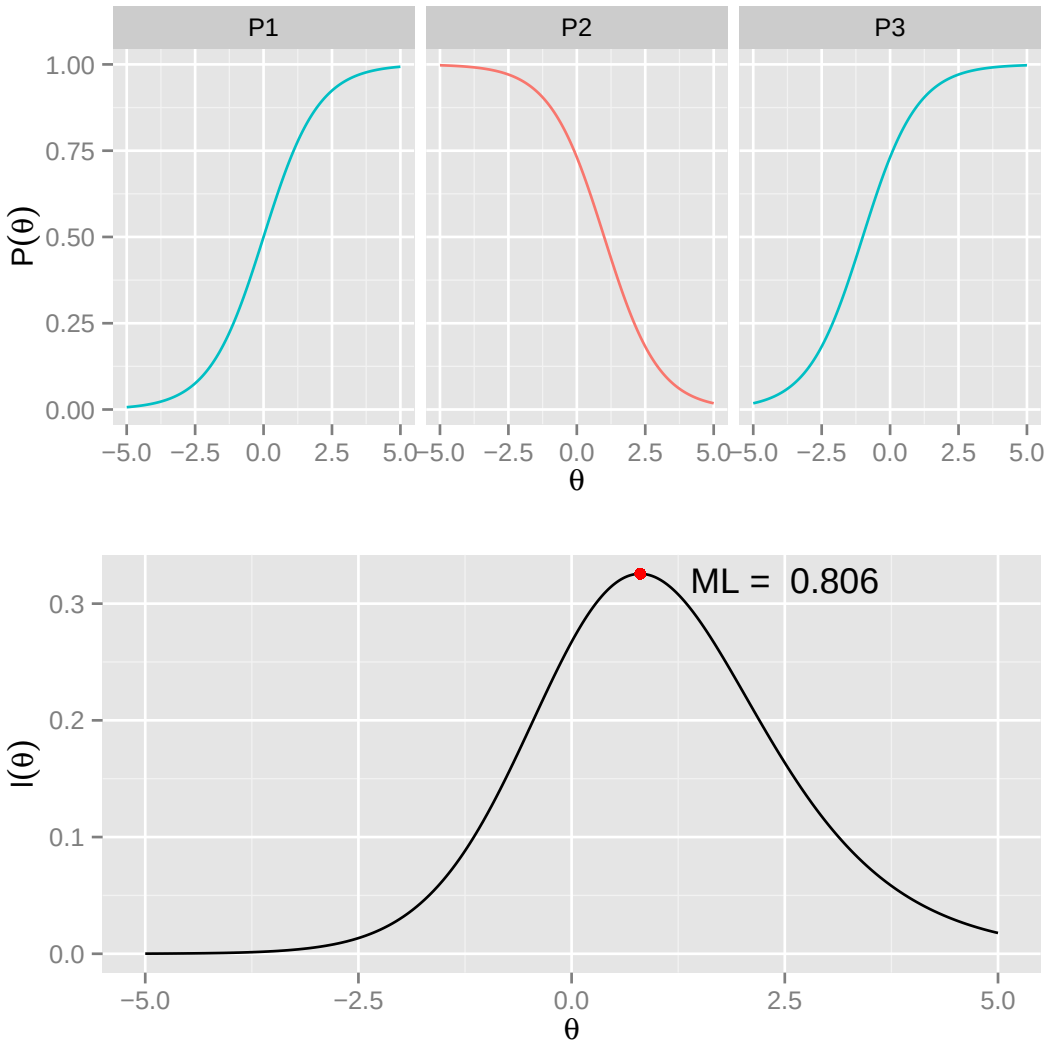
$$L(\theta) = \prod_i \left[P_i(\theta)^{\chi_i} \cdot (1 - P_i(\theta))^{1-\chi_i} \right] \cdot g(\theta),$$

where χ_i is either 0 or 1 for incorrect and correct endorsement, and $g(\theta)$ is a prior distribution.

- ML for this response is just the argmax from the product of 3 item trace-lines, where $g(\theta) \equiv 1$
- Bayesian estimates typically use a normal prior in $g(\theta)$ in computing the posterior mean (EAP) or mode (MAP)

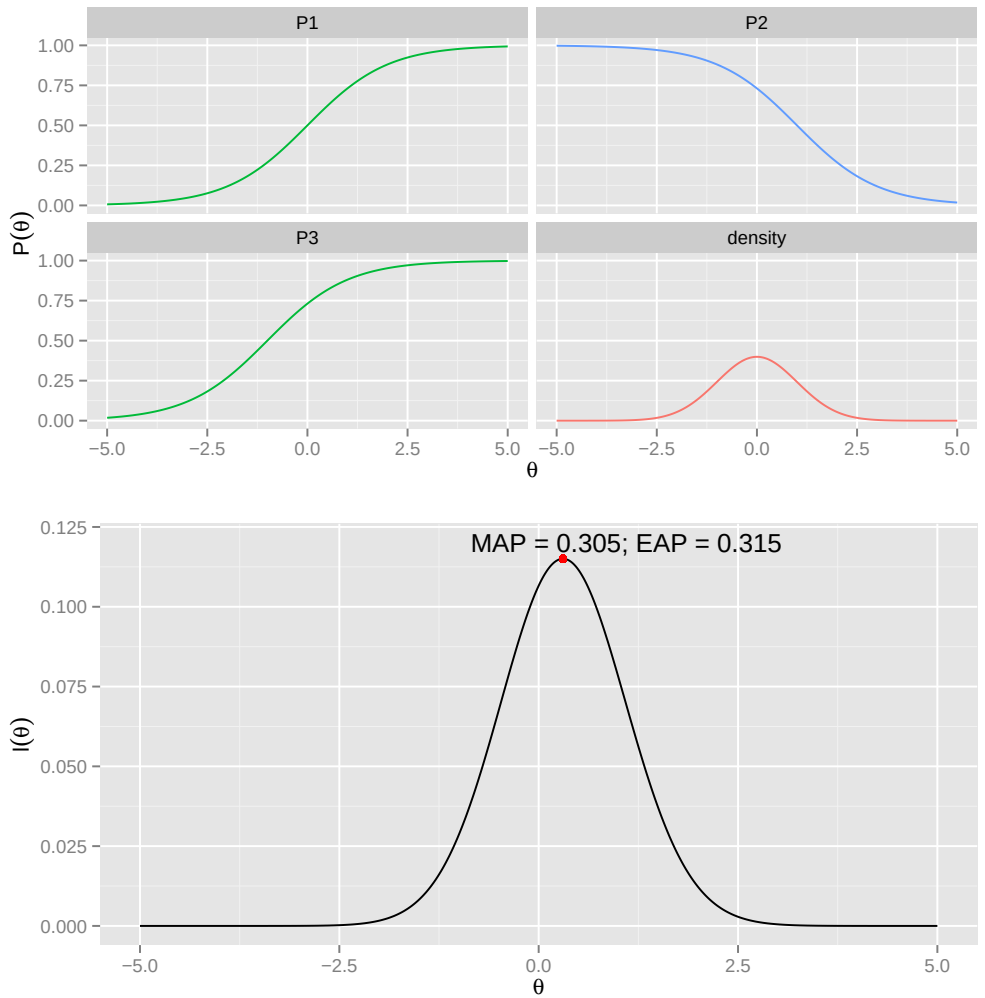
Maximum-likelihood estimation

Say that we had some response pattern [1, 0, 1], and some predefined IRT item parameters ($a = [1, 1, 1]$, $d = [0, -1, 1]$, $g = 0$).



MAP and EAP estimation

Using the same parameters as the previous graphic, including additional information about the distribution of θ will lead to Bayesian estimates. Here an $N(0, 1)$ prior is included.



Uncertainty in predictions

- It is often prudent to estimate the precision of $\hat{\theta}$ to approximate the accuracy of the estimate
- One suitable approach is to use information about the second-order derivatives of the (weighted) log-likelihood/log-posterior distribution (a.k.a., observed information)

$$-(\frac{\partial^2 L}{\partial\hat{\theta}\partial\hat{\theta}'})^{-1} = I(\hat{\theta})^{-1} = ACOV(\hat{\theta})$$

The square-root of the diagonal elements of $ACOV(\hat{\theta})$ give suitable standard errors for each $\hat{\theta}$ value. More information implies smaller standard errors.

Ability prediction methods available in **mirt**

Six algorithms are available in the package to obtain values of latent trait(s) and their standard errors, and are implemented in the `fscores()` function.

1. Maximum likelihood (ML) – Maximize likelihood vector w.r.t. θ directly with iterative methods (returns `Inf` and `-Inf` for all/none patterns)
2. Maximum a posteriori (MAP) – Given a multivariate prior maximize the posterior distribution. Iterative method as well
3. Expected a posteriori (EAP) – Similar to MAP but is not iterative and often a consequence of the MML estimation (mean versus mode estimate)
4. Weighted Likelihood Estimation (WLE) – An iterative estimate of the latent trait that weighs the scores based on how much *test information* is available
5. EAP for summed scores – EAP estimates for θ are derived for using simple sum scores total via a recursive algorithm
6. Plausible value imputations – stochastic imputations of each individual estimate (not used for fixed person estimates)

fscores()

```
fs <- fscores(lsat_mod, method = 'EAP')
```

```
##
## Method:  EAP
##
## Empirical Reliability:
##     F1
## 0.4522
```

```
head(fs)
```

```
##      Item.1 Item.2 Item.3 Item.4 Item.5      F1  SE_F1
## [1,]      0      0      0      0      0 -1.8699 0.6927
## [2,]      0      0      0      0      1 -1.5272 0.6737
## [3,]      0      0      0      1      0 -1.5139 0.6731
## [4,]      0      0      0      1      1 -1.1853 0.6653
## [5,]      0      0      1      0      0 -1.0947 0.6651
## [6,]      0      0      1      0      1 -0.7667 0.6722
```

fscores()

The empirical reliability printout (also returned numerically with `fscores(... returnER = TRUE)`) is a marginalized reliability estimate of the trait scores. This is *highly* related to classical test theory reliability measures:

$$X = T + E$$

$$\sigma^2_X = \sigma^2_T + \sigma^2_E$$

$$r_{XX'} = \frac{\sigma^2_T}{(\sigma^2_T + \sigma^2_E)}$$

- With the IRT scores, $\sigma^2_T = var(\hat{\theta})$ and $\sigma^2_E = mean(SE^2_{\hat{\theta}})$.

fscores() estimates in each row

Because this question comes up so often, here is how to save the scores for each observed response pattern in the data (corresponding to each row).

```
fs_full <- fscores(lsat_mod, method = 'MAP',
                   full.scores = TRUE, full.scores.SE = TRUE)
head(fs_full, 6)
```

```
##              F1     SE_F1
## [1,] -1.816549 0.6750193
## [2,] -1.816549 0.6750193
## [3,] -1.816549 0.6750193
## [4,] -1.816549 0.6750193
## [5,] -1.816549 0.6750193
## [6,] -1.816549 0.6750193
```

Plausible values

Plausible values have a strong history when analyzing tests because they can be used to make group inferences with Rubin's (1987) imputation methodology.

- Each individual estimate is sampled from their respective posterior (rather than obtaining some point estimate)
  - May include latent regression effects for more informative posteriors
- Not useful for making inferences about the individual, but collectively the imprecision is captured in the estimate
- Perform secondary analyses on multiple imputed sets, and average over and collect the between/within variability

This technique is used in many large scale testing programs, such as PISA.

Plausible values

```
pvdat <- simdata(matrix(1, 30), d = matrix(rnorm(30)), N = 500,
                 itemtype = 'dich', sigma = matrix(1.5))
pvmod <- mirt(pvdat, 1, 'Rasch', verbose = FALSE)
pvs <- fscores(pvmod, plausible.draws = 5)
sapply(pvs, var) #variance of imputed estimates
```

```
## [1] 1.516913 1.494971 1.582133 1.610462 1.583809
```

```
mean(sapply(pvs, var)) #average variance
```

```
## [1] 1.557658
```

```
# compare to other point-estimate methods
EAP <- var(fscores(pvmod, method = 'EAP', full.scores = TRUE))
ML <- fscores(pvmod, method = 'ML', full.scores = TRUE)
ML <- var(ML[is.finite(ML)])
WLE <- var(fscores(pvmod, method = 'WLE', full.scores = TRUE))
print(c(EAP=EAP, ML=ML, WLE=WLE))
```

```
##       EAP        ML       WLE
## 1.375588 1.841549 1.763129
```

Customizing `fscores()`

Much like the customization of latent densities in `mirt()`, `fscores()` will accept a customized latent density for predicting $\hat{\theta}$ estimates (only makes sense for Bayesian estimates EAP, MAP, and EAP for sum scores)

```
# EAP estimation with a bimodal prior
fun <- function(Theta, ...) as.numeric(dnorm(Theta, -1.5, 1) +
                                       dnorm(Theta, 1.5, 1))
# Theta <- matrix(seq(-4, 4, length.out = 200))
# ggplot2::qplot(Theta, fun(Theta) / sum(fun(Theta)), geom = 'line')
head(fscores(lsat_mod, custom_den = fun, verbose = FALSE))
```

```
##      Item.1 Item.2 Item.3 Item.4 Item.5      F1  SE_F1
## [1,]      0      0      0      0      0 -2.6491 0.7624
## [2,]      0      0      0      0      1 -2.2379 0.7347
## [3,]      0      0      0      1      0 -2.2221 0.7340
## [4,]      0      0      0      1      1 -1.8318 0.7275
## [5,]      0      0      1      0      0 -1.7228 0.7321
## [6,]      0      0      1      0      1 -1.3045 0.7872
```

Information

Item and test information with respect to θ

Item and test information

Item and test information are very important concepts in IRT, are intimately related to predicting latent trait estimates, and form the building blocks of more advanced applications (such as computerized adaptive testing; CAT)

- The information in a test depends on the items used **as well as the ability of the subject**, and is inversely related to the concept of *reliability*
- *Information*, coined by R. A. Fisher, is directly related to the *steepness of the slopes* in an item. The steeper a slope is, the more information it provides around its inflection point
- For example, easy items and tests tend to tell us very little about individuals in the upper end of the θ distribution ($\theta_{Einstein}$ v.s. $\theta_{Hawking}$) but can tell us something about lower ability subjects (whether $\theta_{Larry} < \theta_{Curly} < \theta_{Moe}$).

Item information

The Fisher (or expected) information function, $\mathcal{F}(\theta)$, is another theoretical way to quantify uncertainty *absent any particular observed response pattern and distribution of* θ.

$$\mathcal{F}(\theta) = -E(\frac{\partial^2 L}{\partial\hat{\theta}\partial\hat{\theta}'})$$

- E.g., for the unidimensional 2PL model (where $g = 0$),

$$\mathcal{F}(\theta) = a^2 P(\theta) \cdot (1 - P(\theta))$$

Fisher's information is important because it contains no reference to the raw data (and therefore can be computed for items that have not been responded to).

IRT 2PL information functions

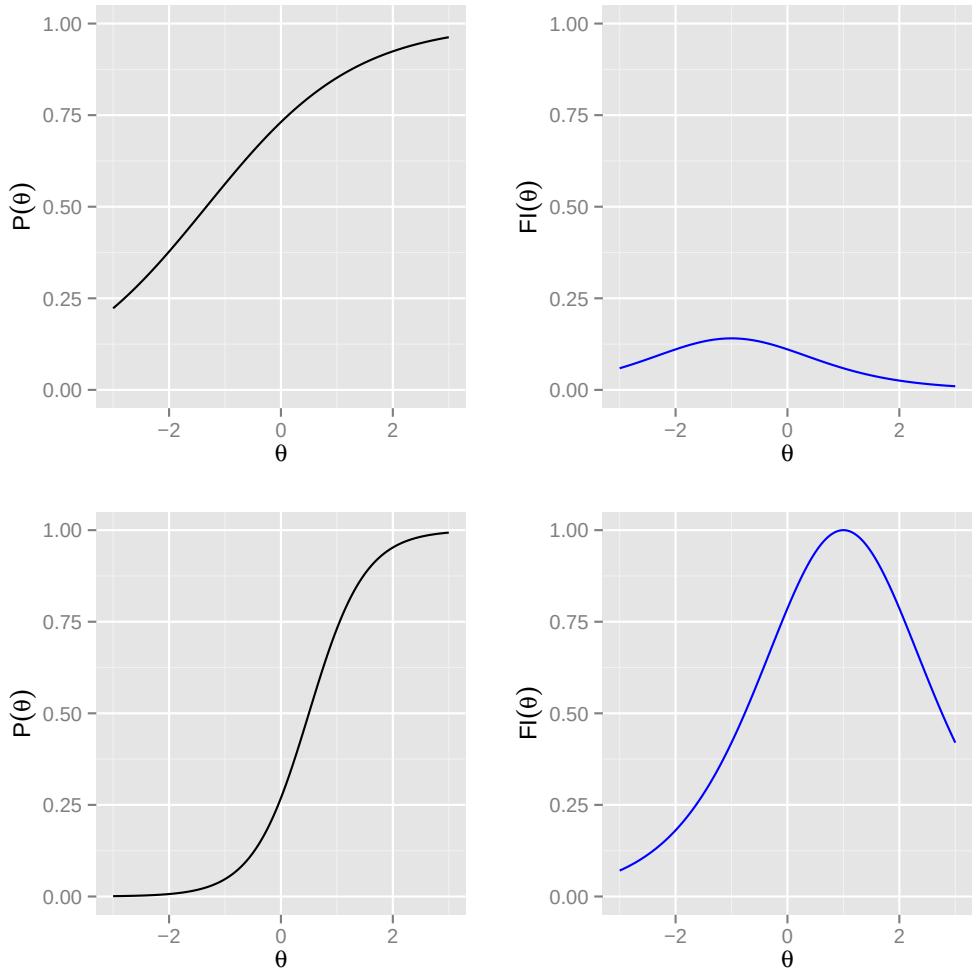


Figure 1: Information functions from 2PL trace-lines with $a = 0.75$, $d = 1$ (top) and $a = 2$, $d = -1$ (bottom).

IRT information functions

In **mirt**, information curves can be found in the `plot()` and `itemplot()` functions.

```
# SE computations only needed for imputation plot
mod1 <- mirt(Science, 1, SE = TRUE)
mod2 <- mirt(Science, 2)

plot(mod1, type = 'infotrace')
itemplot(mod1, 3, type = 'info')
itemplot(mod1, 3, type = 'info', CE = TRUE)
itemplot(mod1, 3, type = 'infoSE')
itemplot(mod1, 3, type = 'infotrace')

itemplot(mod2, 3, type = 'info')
itemplot(mod2, 3, type = 'infocontour')
```

Test information

Item information is additive, so for the entire test $TI(\theta) = \sum_{i=1}^{n} \mathcal{F}_i(\theta)$. This relationship is true for both the observed and expected information functions.

- To find a standard error estimate for $\hat{\theta}$, we can use the relationship $SE(\hat{\theta}) = 1/\sqrt{TI(\hat{\theta})}$
- A test is deemed accurate/reliable if it is able to measure $\hat{\theta}$ with sufficient precision (i.e., small $SE(\hat{\theta})$)
- **Forshadow computerized applications** — Select items that reduce the expected standard error in some optimal way

Test information

Test information is the default `method` argument to `plot()` in **mirt**.

```
plot(mod1)
plot(mod2)
```

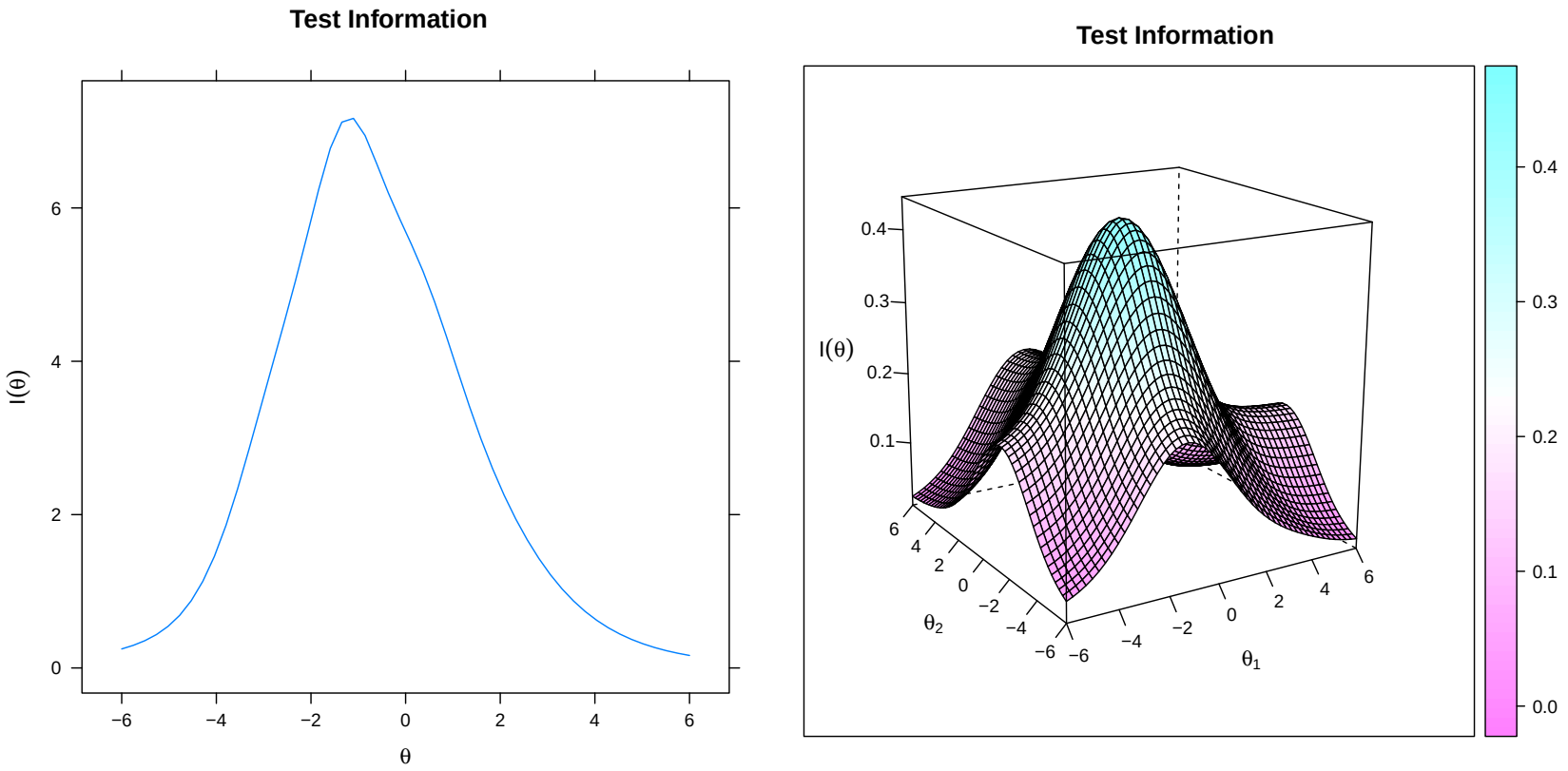


Figure 2: Unidimensionald and multidimensional test information plots in **mirt**.

Exercise

Exercise

You should be able to complete the Exercise found in `Exercise_03.html` now. After that, we'll change gears and look at a different area of applied IRT.

CAT

Computerized adaptive testing

Administer tests by a computer, and dynamically adapt which items are selected given the current estimate(s) of $\hat{\theta}$. This is the core of computerized adaptive testing (CAT).

Computerized adaptive testing

Several benefits to CAT.

- Uninformative items can progressively be avoided (e.g., high-ability participants should not receive easy items)
  - As such, tests can be shorter yet more efficient and accurate
- More precise estimates, especially for those individuals in the extreme ends of the distribution
- Helps to avoid fatigue effects
- Generally better item security (controlled remotely or locally)
- Better control of item exposure (re-tests can be entirely different)

Computerized adaptive testing components

1. Calibrated item bank (a la IRT) for a pool of items
2. Starting rules (initial $\hat{\theta}$'s, pre-CAT data collection, etc)
3. Item selection algorithm
4. Updating $\hat{\theta}$ algorithm
5. Termination criteria
6. (Optional) Include a pre-CAT stage

Points 1) and 2) are pretty straightforward, but the next 3-4 points are largely where the fun occurs.

Item selection

The general idea for item selection is that, given $\hat{\theta}_n$, find the next item such that $SE(\hat{\theta}_{n+1})$ would be the lowest possible value. In other words, pick the next item so that the precision of $\hat{\theta}$ is maximized.

- This reasoning was the primary inspiration for the use of the *maximum expected-information* criteria, or just *maximum-information* (MI) criterion for short
- In unidimensional models, maximum expected information is equivalent to minimum expected error

Additional criteria also exist which do not use information (e.g., Kullback-Leibler information, continuous entropy, etc).

MI item selection

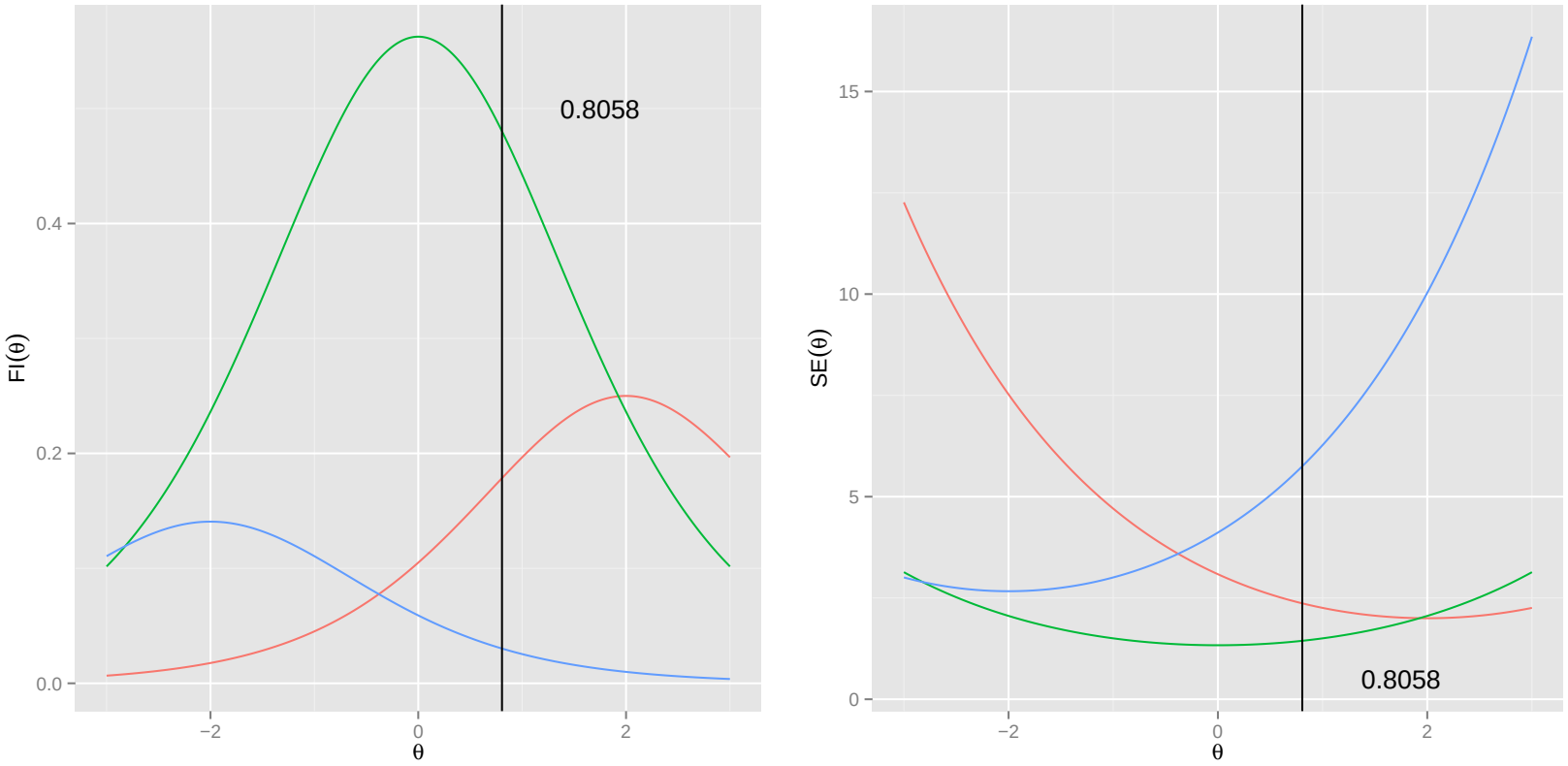


Figure 3: Information and standard errors for three items given $\hat{\theta} = .8058$

Unidimensional classification CATs

Using similar methods, CATs can be used to classify individuals as above or below various threshold values (e.g., determine whether students have 'mastered' the material).

- Easiest method conceptually is to test whether the current estimate confidence intervals include the cutoff value (call it γ). If

$$\frac{|\hat{\theta} - \gamma|}{SE_{\hat{\theta}}} > q_z(1 - \alpha/2)$$

  then terminate the CAT (estimate does not contain the cutoff value as a plausible location).
- Same logic can be applied to multidimensional CAT for classification testing, which we now turn to.

Multidimensional item selection

Multidimensional IRT models, on the other hand, are slightly more complicated in CATs.

- Items may measure more than one trait at once,
- Redundant information may exist if the latent traits are correlated
- Some traits may be more important to measure than others (think bifactor model)

When one or more of these criteria occur it may be more beneficial to perform a multidimensional CAT instead of multi-unidimensional CATs.

Multidimensional item selection

The selection process generally involves evaluating the new $\mathcal{F}(\theta)$ matrices for all remaining items in the pool.

- E.g., information for M2PL with two factors is

$$\mathcal{F}(\theta) = \begin{pmatrix} a_1^2 & a_1 a_2 \\ a_1 a_2 & a_2^2 \end{pmatrix} P(\theta) \cdot (1 - P(\theta))$$

- Similar to the unidimensional information, except now multiple a parameters are included and the result is a matrix, not a scalar

Multidimensional item selection

To compare matrices we should collapse all $\mathcal{F}_p(\theta)$ into a scalars using suitable criteria. One such criteria is the maximum determinant rule (Segall, 1996)

$$\text{D-rule} = |TI(\theta) + \mathcal{F}_p(\theta)|$$

This optimally decreases the expected volume in $ACOV(\theta)$.

If prior information about the latent traits is to be included, then the inverse of their variance may be included in the computations to create a posterior selection rule.

$$\text{DP-rule} = |TI(\theta) + \mathcal{F}_p(\theta) + \Sigma(\theta)^{-1}|$$

Graphical representation of D-rule

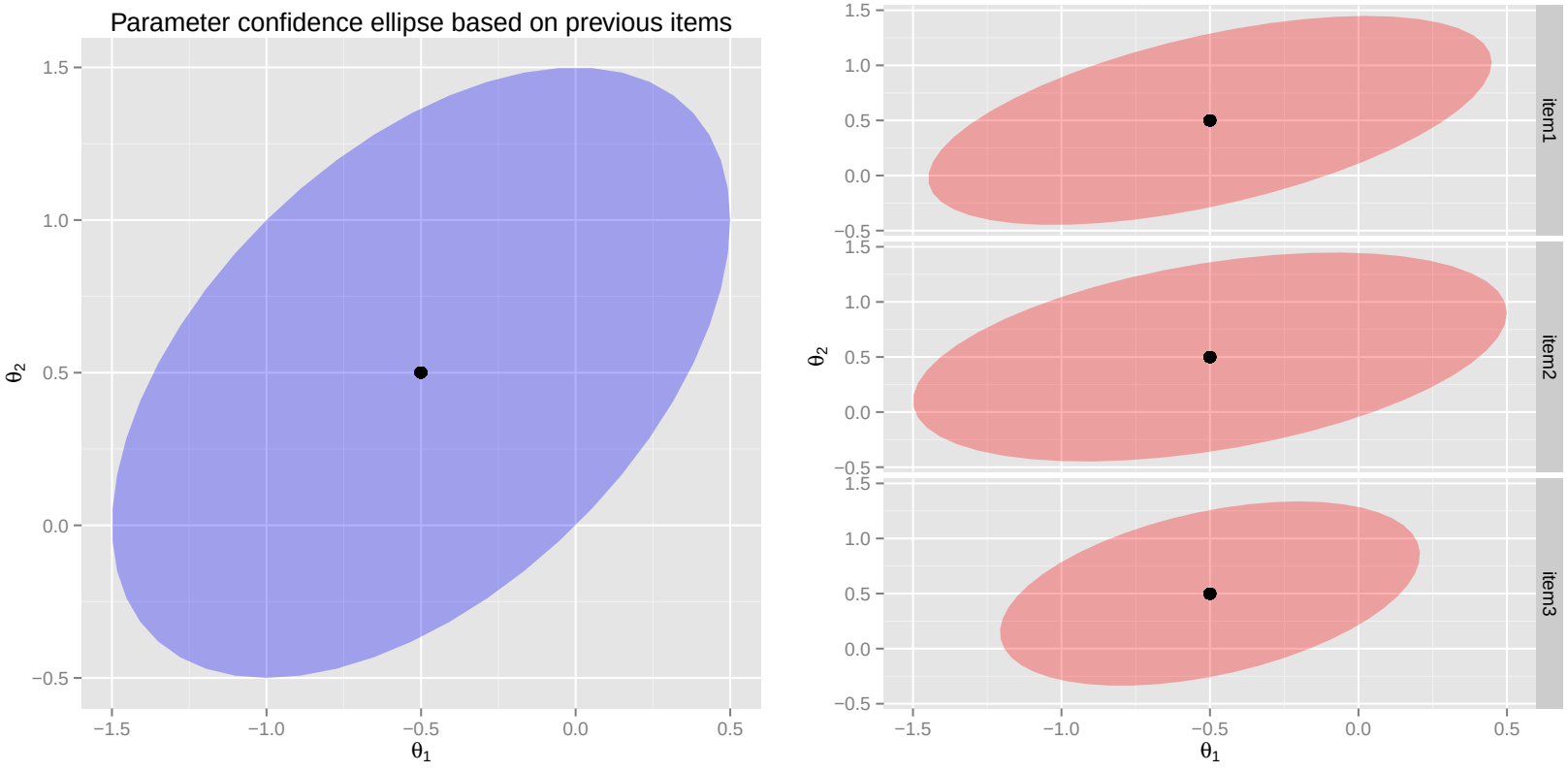


Figure 4: Confidence ellipses for three items given $\hat{\theta} = [-0.5, 0.5]$

Multidimensional item selection

Alternative selection schemes are possible to favor item selection for particular traits, such as the weighted trace of the information

$$\text{T-rule} = \text{Tr}((TI(\hat{\theta}) + \mathcal{F}_p(\hat{\theta}))\mathbf{W})$$

or $ACOV(\hat{\theta})$

$$\text{A-rule} = \text{Tr}((TI(\hat{\theta}) + \mathcal{F}_p(\hat{\theta}))^{-1}\mathbf{W})$$

matrix, where $\mathbf{W}$ is a diagonal weight matrix, and the weighted rule

$$\text{W-rule} = \mathbf{w}'(TI(\hat{\theta}) + \mathcal{F}_p(\hat{\theta}))\mathbf{w},$$

where $\mathbf{w}$ is a weight vector.

Update traits, and terminate test

After the item has been selected, and the response has been collected, $\hat{\theta}$ is updated using one of the methods proposed earlier (alternatively, with the expected a posteriori (EAP) or weighted likelihood estimation (WLE)).

Finally, the test is checked regarding whether it should be terminated.

- $SE(\hat{\theta})$ based on the observed information is less than some predefined tolerance
- a predefined number of items have been administered
- the change in $\hat{\theta}_n$ and $\hat{\theta}_{n+1}$ is less than some δ
- testing time has run out
- individuals can be classified above or below predefined cutoffs
- and so on

Exposure control

One possible issue in CATs is the overuse of items that contain a lot of information (i.e., have large slopes/large factor loading commonalities).

- More informative items get selected more often (naturally), but less informative items rarely get selected
- This causes over exposure of some items, which is counter-productive if the whole item bank should be used
- Several other negative consequences: harder to maintain item security, loss of payout for calibrating items if they are never used, decrease in content coverage, and so on

To help alleviate over exposure of items, several *Exposure Control* methods have been proposed.

Exposure control

One simple exposure control method is to simply sample from the items with the most optimal criteria.

- E.g., instead of simply selecting the criteria with the largest MI, sample from the top n criteria
- Can also change the value of n across the CAT, where n may be larger earlier on in the test (e.g., 5-4-3-2-1-1-1...)

Sampling is crude, but accomplished the goal. Other options, such as the Sympson-Hetter approach, use simulation experiments to determine whether an item gets administered:

- Each item gets a unique SH_i value between 0 and 1
- When each item is selected, a value r is drawn from `runif(1, min=0, max=1)`
- If $SH_i >= r$, administer the item, otherwise remove it from the test bank and try again with the next most optimal item

Content balancing

Yet another area of interest is to ensure that various 'Content Areas' are sampled from during the test (e.g., in a mathematics test, a certain proportion of calculus, algebra, and geometry items items should appear)

- Simple methods include classifying the items according to these groups, declaring a proportion, and then selecting the items throughout the test to minimize the observed versus required proportions (Kingsbury and Zara, 1991)
- Other methods are possible with MCATS, such as setting up bi-factor models where each specific trait is related to a particular 'content' area (Segall, 1996)

mirtCAT package

The **mirtCAT** package

Building and analyzing unidimensional and multidimensional CAT interfaces in R.

mirtCAT package

Why waste time building **mirtCAT** for the R environment?

- R based solution to building CAT GUIs was non-existent
- CAT packages in R were only for simulation designs (e.g., **catR**, **catIrt**, **MAT**) and contained a mix of flexibility and efficiency
- Multidimensional models limited only the multidimensional 3PL model
- None of the packages could handle mixed item types

I wanted an interface to collect survey data that was adaptive OR non-adaptive that integrated directly with R, and could be run locally or remotely. Thankfully, most of the core code work was already written in the **mirt** and **shiny** packages.

mirtCAT package

There really only three main functions in the package:

- `mirtCAT()` – used to run CAT designs off-line, or to generate GUIs for real time CAT applications. Objects returned from this function are of class `'mirtCAT'`, and can be passed to the S3 methods `plot()`, `summary()`, and `print()`
- `generate_pattern()` – generate potential response pattern for CAT application. Useful for Monte Carlo work, and to test the CAT designs
- `generate.mirt_object()` – generate a suitable **mirt** object from population parameters if the MIRT model was not estimated from empirical data

Other functions are available for more specialized work, but are essentially never required (e.g., `findNextItem()`, `updateDesign()`, etc).

mirtCAT

```
mirtCAT(df, mo, ...)
```

- `df` is a `data.frame` object containing question stems, options, answers, and image stem paths
- `mo` is an object defined by **mirt** containing the item/group parameters
- depending on the purpose, one of these inputs can be missing (only `df` supplied, a survey is generated, only `mo` is supplied, an off-line MCAT is run for simulations; supplying both creates the adaptive interface)

Main arguments

The primary arguments to **mirtCAT** (with default in brackets)

- `method` – (`'MAP'`) θ estimation method
- `criteria` – (`'seq'`) item selection criteria
- `start_item` – (`1`) starting item, can be a character similar to `method`
- `local_pattern` – (`NULL`) run a locally defined vector off-line
- `cl` – (`NULL`) a socket-type object for running simulations in parallel
- `design_elements` – (`FALSE`) return an object containing the CAT design elements

For modifying other more specific elements in the GUI and CAT designs, the following lists may be passed: `design`, `shinyGUI`, and `preCAT`.

A simple survey

```
options <- matrix(c("Strongly Disagree", "Disagree",
                    "Neutral", "Agree", "Strongly Agree"),
                  nrow = 3, ncol = 5, byrow = TRUE)
questions <- c("Building CATs with mirtCAT is difficult.",
               "Building tests with mirtCAT requires
                                    a lot of coding.",
               "I would use mirtCAT in my research.")

df <- data.frame(Question = questions, Option = options,
                 Type = "radio")
results <- mirtCAT(df = df)
```

Several features available

- Exposure control (controlled sampling, Sympson-Hetter)
- Content balancing
- Pre-CAT designs to collect data responses prior to beginning the CAT
- Support for all models in **mirt**, which can be mixed in a single session (e.g., 70% 3PL models, 30% GPCM)
- Many item selection criteria (sequential, random, KL, MI, MEPV, . . . , D-rule, DP-rule, W-rule, . . .)
- Implicit parallel computing support for Monte Carlo simulations
- Multiple $\hat{\boldsymbol{\theta}}$ estimators (all estimators from the **mirt** package, including custom priors)
- Customization of GUI elements

Tables of inputs

- The list inputs are fairly verbose, and should be read from the help files.
- However, here is a cursory look at the tables presented in the current version of the **mirtCAT** draft (Chalmers, submitted)

Single case simulated data example

To demonstrate how multidimensional CAT works from a theoretical perspective, we can compare how measurement error changes when administering item sequentially versus using CAT.

- Several ways to do this, but the simplest is to set `min_SEM` to a small value or `min_items` to the length of the test

Single case simulated data example

- Simulated 120 dichotomous items with two factors ($r = 0.5$)
- The items were set up such that the first 30 measured the first dimension only, next 30 contained multidimensional items (measuring dimension one better), next 30 were multidimensional (measuring dimension two better), and finally last 30 measured only dimension two

$$\begin{bmatrix} A & 0 \\ A & a \\ a & A \\ 0 & A \end{bmatrix}$$

Single case simulated data example

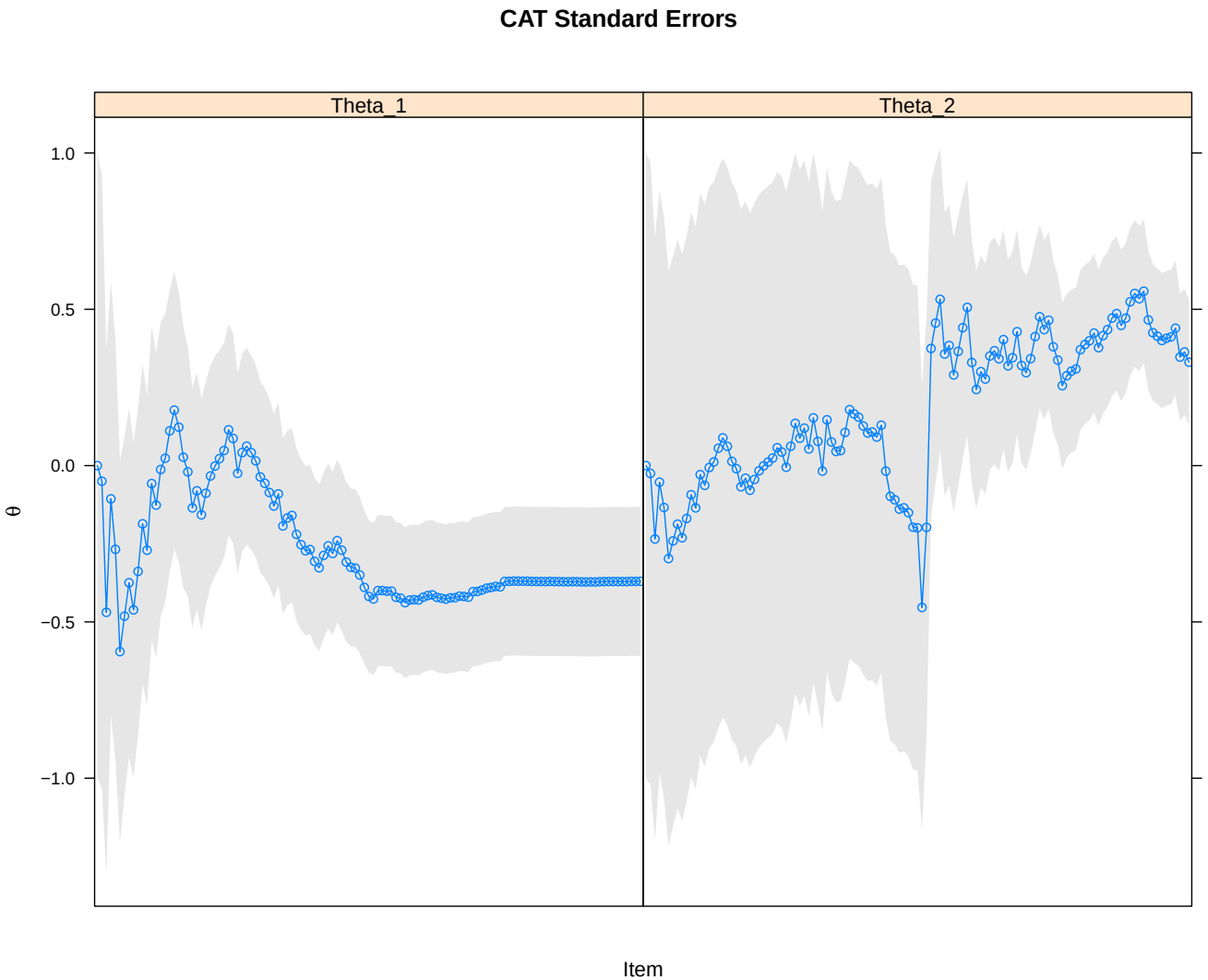


Figure 5: A response vector given $\theta = [-0.5, 0.5]$ was drawn to obtain plausible answers for the entire test.

Single case simulated data example

After the 120 items were administrated, the following information was obtained:

- using MAP estimation: $\hat{\theta}_1 = -0.370$, $SE(\hat{\theta}_1) = 0.238$, and $\hat{\theta}_2 = 0.3307$, $SE(\hat{\theta}_2) = 0.199$
- as well, it took until the **73rd item** before both SE estimates were less than 0.4

Compare this now to a CAT application using the D-rule to select items, terminating when all SE estimates were less than 0.4.

Single case simulated data example

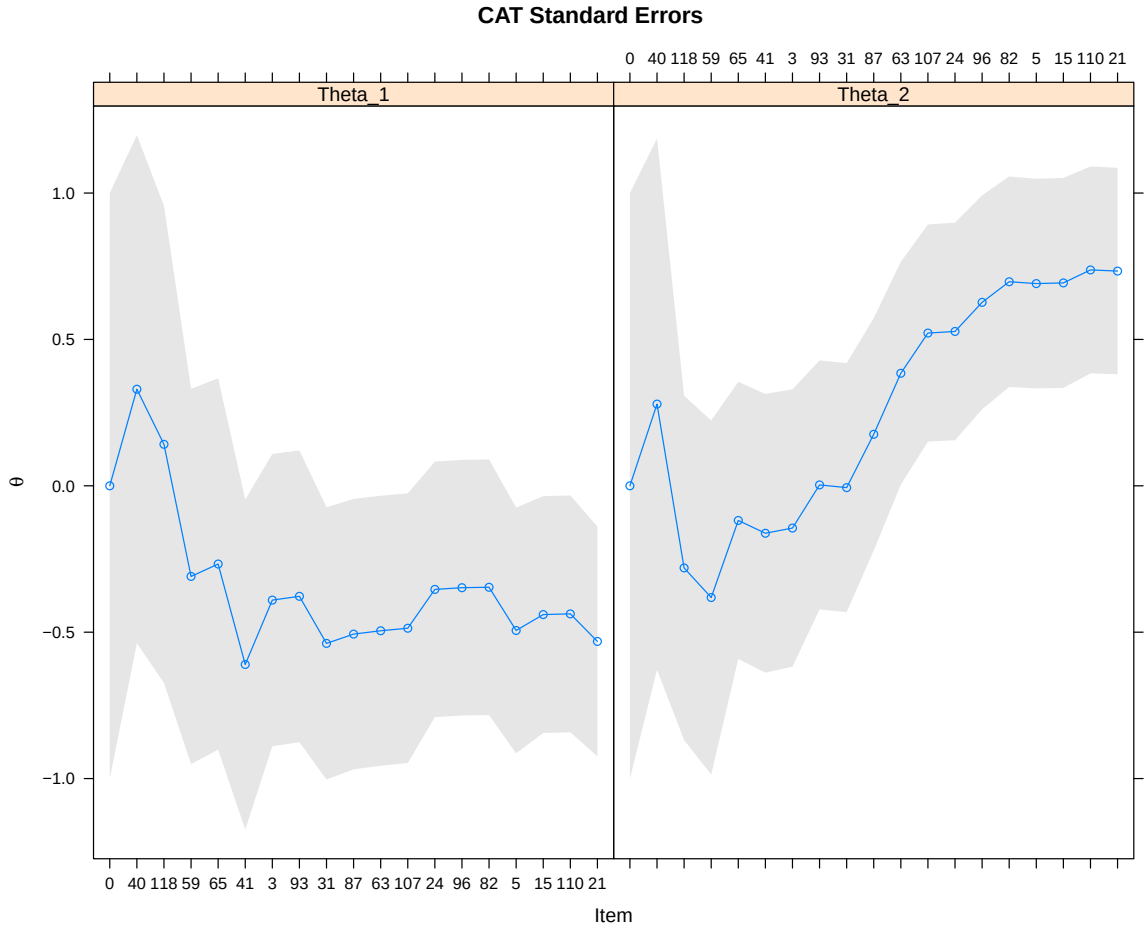


Figure 6: *Test was stopped after 18 items*. Final estimates: $\hat{\theta}_1 = -0.531$, $SE(\hat{\theta}_1) = 0.392$, and $\hat{\theta}_2 = 0.733$, $SE(\hat{\theta}_2) = 0.352$

Example : **mirtCAT** GUI customization and further examples

Example

Table 3 from the mirtCAT draft, and a worked example of a more complex interface.
Also, various examples that have been posted on the Github wiki `https://github.com/philchalmers/mirtCAT/wiki` that can be sourced to better understand how to manipulate the output.

mirtCAT for simulation work

When parameters are known a priori, as they are in simulation work, the `generate.mirt_object()` function can be used to build a suitable `mo` object. The following builds a 10 item bank of 4PL models.

```
a <- rlnorm(10, .2, .2)
d <- rnorm(10)
pars <- data.frame(a1 = a, d = d, g = 0.2, u = .95)
vcov <- matrix(2)
mirt_object <- generate.mirt_object(pars, itemtype = '4PL',
                                    latent_covariance = vcov)
```

mirtCAT for simulation work

```
coef(mirt_object, simplify = TRUE)
```

```
## $items
##              a1      d   g    u
## Item.1   0.925 -0.127 0.2 0.95
## Item.2   1.475 -1.104 0.2 0.95
## Item.3   1.770 -0.109 0.2 0.95
## Item.4   1.007  1.120 0.2 0.95
## Item.5   1.262 -0.662 0.2 0.95
## Item.6   1.686 -0.551 0.2 0.95
## Item.7   0.853 -0.476 0.2 0.95
## Item.8   1.449 -0.597 0.2 0.95
## Item.9   1.362  0.258 0.2 0.95
## Item.10 0.903 -1.004 0.2 0.95
##
## $groupPars
## $groupPars$means
## MEAN_1
##      0
##
## $groupPars$cov
##    F1
## F1  2
```

mirtCAT for simulation work

For simulation work the `generate.pattern()` function will accept a matrix input for `Theta`, returning multiple rows with plausible response pertaining to each row in `Theta`. The response can then easily be passed to `mirtCAT()` to run each row:

```
Theta <- matrix(rnorm(1000))
responses <- generate_pattern(mirt_object, Theta)
res <- mirtCAT(mo = mirt_object, ..., local_pattern = response)
```

- This returns a list of estimated CAT objects to be summarized
- Can include a `cl` object for parallel computing with the **parallel** package

Comparison with other CAT packages

mirtCAT has been compared with 3 other R packages: **catR**, **catIrt**, for unidimensional CATs, and **MAT**, for the multidimensional 3PL. All designs contained 1000 items.

- Across all simulations, results from **mirtCAT** were consistent with the other packages (estimates $r > .9999$)
- Estimation times widely differed though.
  - For unidimensional models with $N = 5000$: **catIrt** (95 minutes), **catR** (28 minutes), **mirtCAT** (11.5 minutes), **mirtCAT** in parallel (using 8 cores, 3 minutes)
  - For multidimensional and $N = 1000$, **MAT** (259 seconds) and **mirtCAT** (435 seconds)

Comparison with other CAT packages

However, **mirtCAT** encapsulates several features of the other CAT packages in R

- contains support for many different estimators (contains all estimators from other packages, and then some), and item selection criteria for unidimensional and multidimensional models (contains most, if not all, item selection criteria available in other R packages)
- various CAT designs properties (exposure control, content balancing, pre-CATs) not present in one or more packages
- mixed item types (including custom items)
- unidimensional and multidimensional IRT modeling support
- GUI capabilities for real-time CATs
- and so on.

Passing responses to `fscores()`

Because **mirtCAT** calls **mirt** to estimate various aspects, re-estimating abilities once the test is complete is also possible.

```
responses <- summary(result, sort = FALSE)$scored_responses
fscores(mirt_object, response.pattern = responses,
        method = "ML")
```

```
##      Item.1 Item.2 Item.3 Item.4 Item.5 Item.6 Item.7 Item.8
## [1,]      1      1      1      0      1      0      0      1
##      Item.10        F1     SE_F1
## [1,]       1 0.5976364 0.8816145
```

Passing responses back to **mirt**

In the same light as with `fscores()`, data can be extracted from **mirtCAT** and added to the original `data.frame` used to calibrate the item parameters. This provides a fluid work-flow between collecting and calibrating item parameters.

```
result <- mirtCAT(df, oldmodel, ...)
responses <- summary(result, sort = FALSE)$scored_responses
new_dat <- rbind(org_dat, responses)
newmodel <- mirt(new_dat, model, ...)

# given new model, run mirtCAT() again....ad nosium
result <- mirtCAT(df, newmodel, ...)
responses <- summary(result, sort = FALSE)$scored_responses
...
```

Customizing GUI elements

- Overview of GUI elements presented in the current version of the **mirtCAT** draft
- Additional details are available in `help(mirtCAT)`
- Include a majority of lower level changes, therefore languages other than English can be used if the test developer wishes

Future work

Designing tests and surveys with **mirtCAT** is, IMHO, fairly easy. Most of the work to set up tests with the **shiny** package has been extracted from the user, and setting up CATs is nearly as simple using information from **mirt**.

In the future I may consider adding:

- testlet based questions (e.g., ETS style)
- support for audio and video files
- interactive item content (**ggvis**, **googleVis**)
- multiple questions per-page (surveys only)
- shadow testing design
- whatever else users want to see (and is possible)

Exercise

Final Exercise

The final exercise file is located in `Exercise_4.html`

End of Workshops

This is the end of the two day Workshops (congratulations!). Just to review what we learned today:

- Customizing density functions for estimation model parameters and person estimates in **mirt**
- Non-parametric estimation of the latent density with empirical histograms
- Latent trait estimation algorithms, and their connection with test and item information
- The fundamentals of (multidimensional) computerized adaptive testing
- How MCATs and standard CATs can be organized and implemented with the **mirtCAT** package

Additional information

mirt and **mirtCAT** are actively being developed by yours truly. The development versions of the packages are obtainable on Github. Feature requests and bugs can be sent there by opening 'Issues', or, if you are so inclined, even code modifications that you wrote which you think would improve the package (called 'pull requests').

- Github: `https://github.com/philchalmers/mirt` and `https://github.com/philchalmers/mirtCAT`
- Github wiki's: `https://github.com/philchalmers/mirt/wiki` and `https://github.com/philchalmers/mirtCAT/wiki`

For general questions about the package, or other IRT related topics on analyzing data, there also is a Google mailing list which is free to join and contribute to.

- Google Forum: `https://groups.google.com/forum/#!forum/mirt-package`

References

Chalmers, R. P. (in review). Generating Adaptive and Non-Adaptive Test Interfaces for Multidimensional Item Response Theory Applications.

Chalmers, R. P. (in review). Extended Mixed Effect Item Response Models.

Fischer, G. H. (1983). Logistic latent trait models with linear constraints. *Psychometrika, 48*, 3-26.

Kingsbury GG, Zara AR (1991). A Comparison of Procedures for Content-Sensitive Item Selection in Computerized Adaptive Tests. *Applied Measurement in Education, 4*, 241–261.

Segall DO (1996). Multidimensional Adaptive Testing. *Psychometrika, 61*, 331-354.