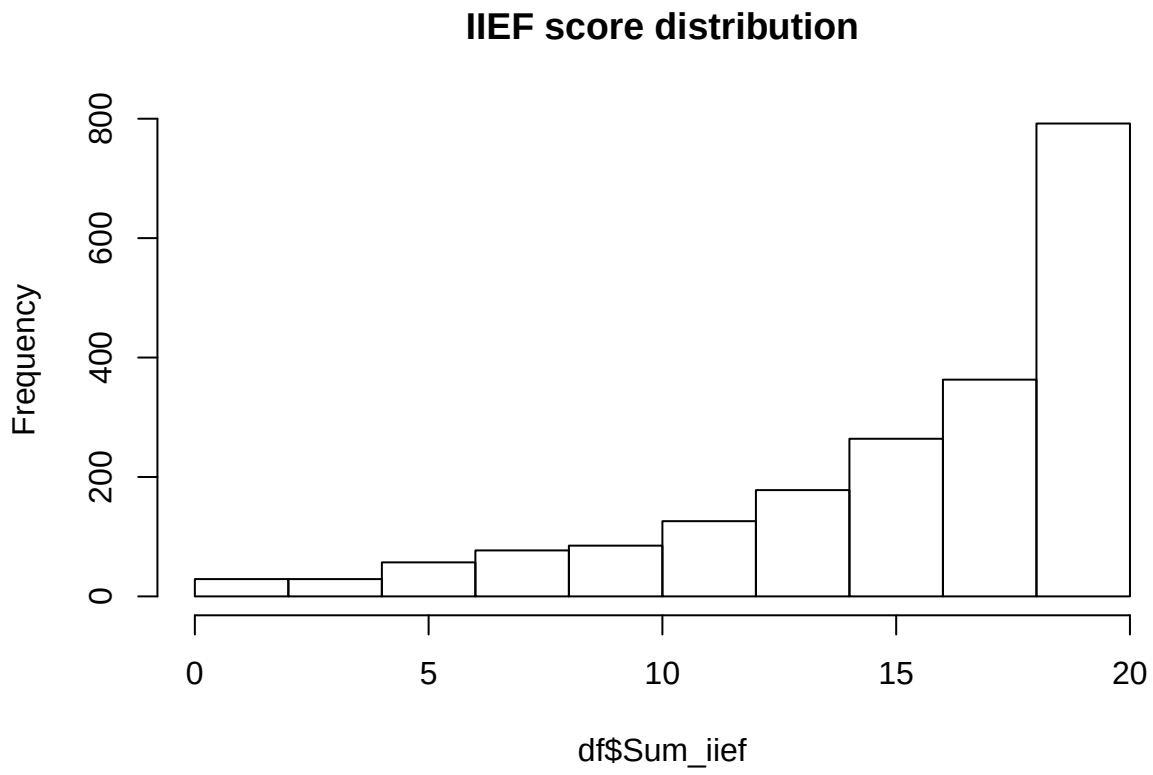
Concurrent Calibration of Psychometric Questionnaires Using Item Response Theory in R

This document illustrates test equating of two short form medical diagnostic questionnaires using item response theory. The purpose here is primarly to illustrate how the equating process of concurrent calibration can be implemented using the mirt package in R, and to predict the scores on one survey from the other.

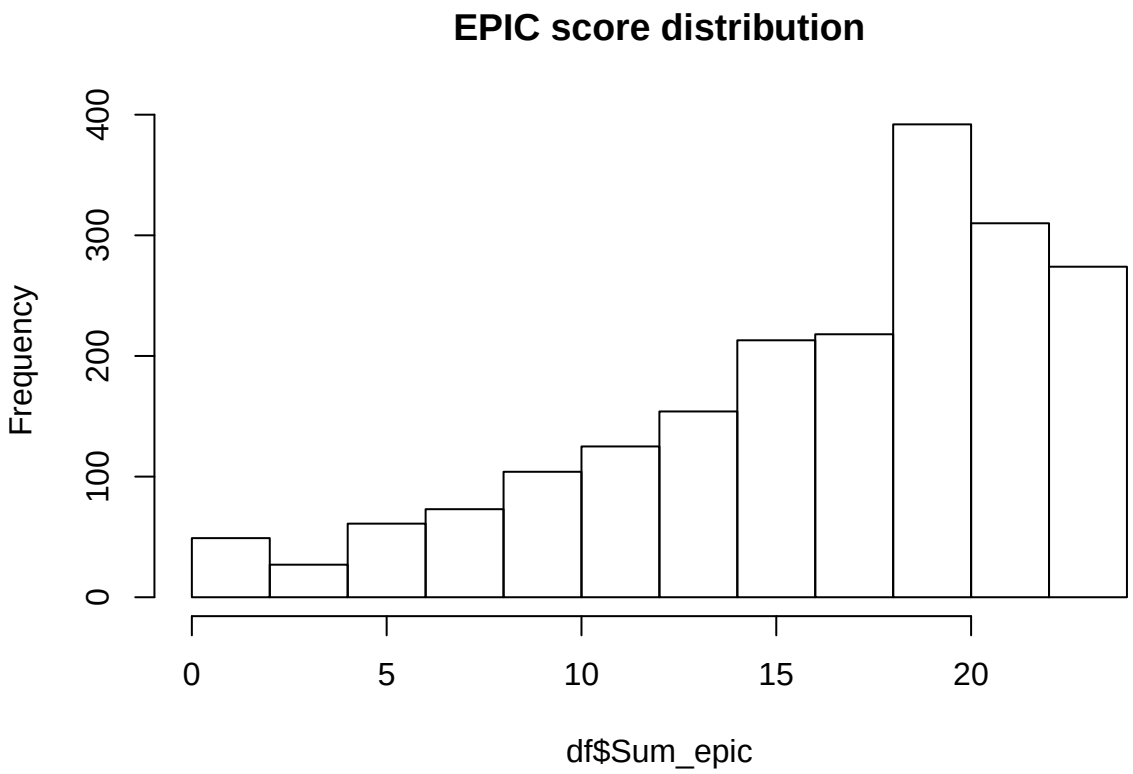
Test equating involves converting two psychometric surveys to a common scale so that scores can be compared between them. Normally test equating with item response theory involves a transformation equation using linking coefficients, which is analogous to slope and intercept parameters as used in linear regression. However, when two forms are designed to measure a common ability distribution, the parameters from both forms can also be estimated jointly from one dataset. A model for each form can then be estimated from those parameters to perform test equating, a process known as concurrent calibration.

The ultimate aim in this example is to predict the score on one form used to measure erectile function, the Expanded Prostate Cancer Index Composite Short Form Sexual Domain, from item responses from an older instrument, the International Index of Erectile Function-5. We will refer to the summation of the scores from all items on the survey as the sum score. Let's take a look at the distribution of sum scores from both surveys:

```
df = read.csv('concurrent_calibration.csv')
hist(df$Sum_iief, main = 'IIEF score distribution')
```




```
hist(df$Sum_epic, main = 'EPIC score distribution')
```



Both score distributions are heavily tailed, which is common for instruments featuring ordinal data. This is because most people surveyed will tend to have little or no dysfunction in a given health area, corresponding to a high average score. Fortunately, normally distributed data is not a requirement for most IRT models.

Since both forms feature Likert-type (polytomous) questions, we'll use a graded response model to estimate person and item parameters. The GRM is essentially a generalization of the two-parameter logistic model in which functions for multiple response categories are aggregated. The category response function for a GRM is the probability of a given response (denoted by k) to a given item (denoted by j) associated with a given trait level (denoted by θ)

$$P_{jk}(\theta) = \begin{cases} 1 - P^*_{j1}(\theta), & \text{k = 1} \\ P^*_{j(K-1)}(\theta), & \text{k = K} \\ P^*_{j(k-1)}(\theta) - P^*_{jk}(\theta) & \text{otherwise} \end{cases}$$

In this formula, K represents the highest response category of item j, whereas $P^*_{jk}(\theta)$ represents the boundary response function

$$P^*_{jk}(\theta) = \{1 + exp[-\alpha_j(\theta - \beta_{jk})]\}^{-1}$$

. Here α_j is the slope (analogous to the discrimination parameter) of item j, and β_{jk} is the category threshold parameter for response k of item j. This parameter represents the point on the theta scale at which respondents have a 0.5 probability of responding above category k.

Let's use the mirt package to estimate a GRM for both forms concurrently, and extract the slope/discrimination (here called $a1$ for each item) and category threshold parameters (denoted as d^k):

```
library(mirt)
base_model <- mirt(df[,1:11], model = 1, itemtype = 'graded')
parameters = mod2values(base_model)
```

```
head(parameters)
```

```
##   group   item  class name parnum     value lbound ubound  est prior.type
## 1   all iief_1 graded   a1      1  2.942408   -Inf    Inf TRUE       none
## 2   all iief_1 graded   d1      2  6.667940   -Inf    Inf TRUE       none
## 3   all iief_1 graded   d2      3  4.371396   -Inf    Inf TRUE       none
## 4   all iief_1 graded   d3      4  1.227523   -Inf    Inf TRUE       none
## 5   all iief_1 graded   d4      5 -1.472179   -Inf    Inf TRUE       none
## 6   all iief_2 graded   a1      6  3.128465   -Inf    Inf TRUE       none
##   prior_1 prior_2
## 1     NaN     NaN
## 2     NaN     NaN
## 3     NaN     NaN
## 4     NaN     NaN
## 5     NaN     NaN
## 6     NaN     NaN
```

Next, we'll create two subsets of the full item parameter list to estimate the individual EPIC and IIEF GRMs.

```
iief_parameters = subset(parameters, !grepl('epic', item))
epic_parameters = subset(parameters, !grepl('iief', item))

# This prevents MIRT from re-estimating the parameters
iief_parameters$est = FALSE

# Re-assign parameter number for the new model
iief_parameters$parnum = 1:nrow(iief_parameters)

epic_parameters$est = FALSE
epic_parameters$parnum = 1:nrow(epic_parameters)
```

Now we will estimate two additional graded response models, using the corresponding parameters estimated from the base model. Then we will use the fscores function to map a theta value to each possible sum score for both models. The score list returned for the IIEF scores will map a theta value to each respondent, but the possible theta values as returned by the "EAPsum" method still only allow for a fixed theta value for each sum score regardless of response pattern.

```
model_iief = mirt(df[,1:5], model = 1, pars = iief_parameters, itemtype = 'graded')
model_epic = mirt(df[,6:11], model = 1, pars = epic_parameters, itemtype ='graded')

epic_scores = as.data.frame(fscores(model_epic, method = "EAPsum", full.scores = FALSE))
iief_scores = as.data.frame(fscores(model_iief, method = "EAPsum", full.scores = FALSE))

#This returns a theta value for each individual participant
iief_scores_full = as.data.frame(fscores(model_iief, method = "EAPsum", full.scores = TRUE))
```

```
print(head(iief_scores_full))
```

```
##        Theta
## 1 -0.2691055
## 2  0.4763703
## 3 -0.4349264
## 4  0.1593186
## 5 -1.5829373
## 6 -0.2691055
```

```
print(head(epic_scores))
```

```
##   Sum.Scores     Theta  SE.Theta observed expected
## 0          0 -2.556683 0.3988734       17 19.38885
## 1          1 -2.194565 0.2714301       13 16.25159
## 2          2 -2.004421 0.2361317       19 16.49482
## 3          3 -1.854222 0.2166051       12 17.81618
## 4          4 -1.724264 0.2065565       15 20.14191
## 5          5 -1.606279 0.2007406       25 23.35828
```

We can use the expected and observed number of participants with each sum score to create goodness of fit plots for both models. This provides one measure of insight into how well our data conforms assumptions made by IRT.

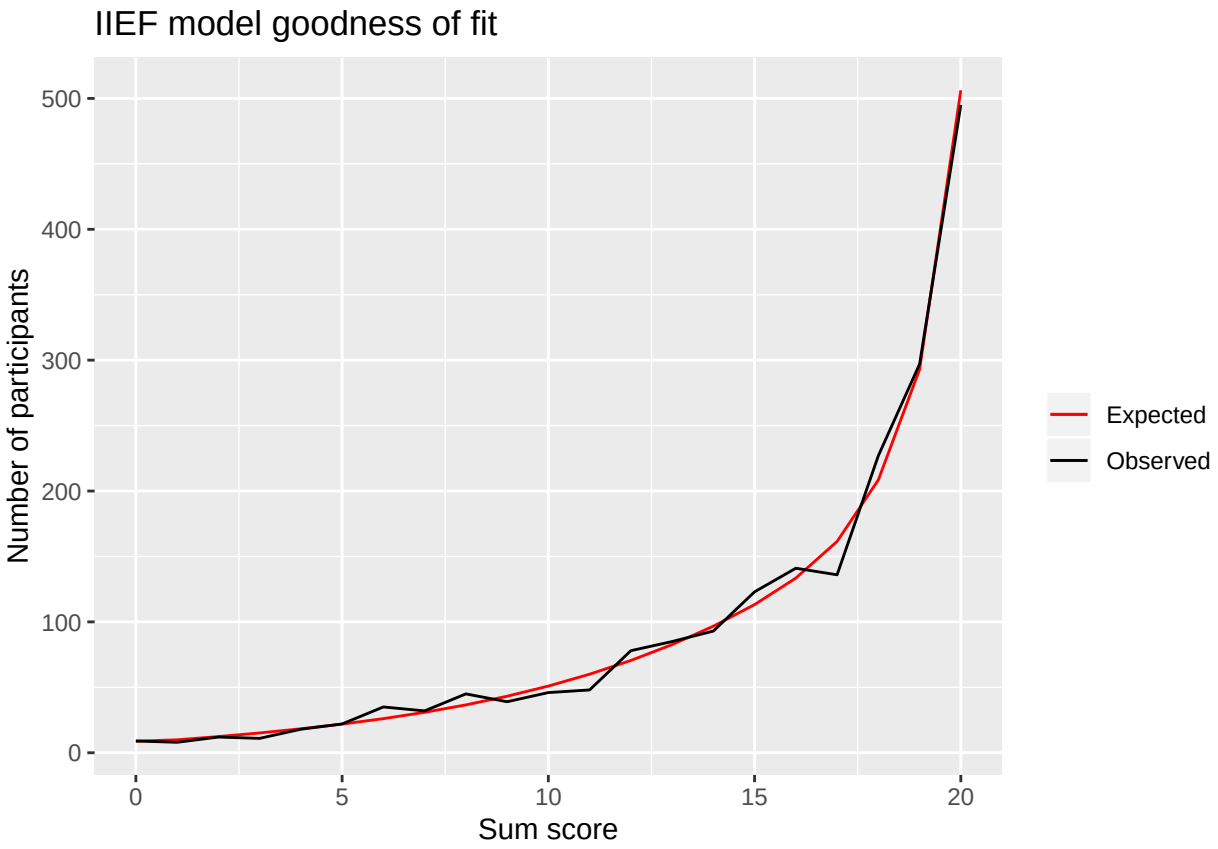
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

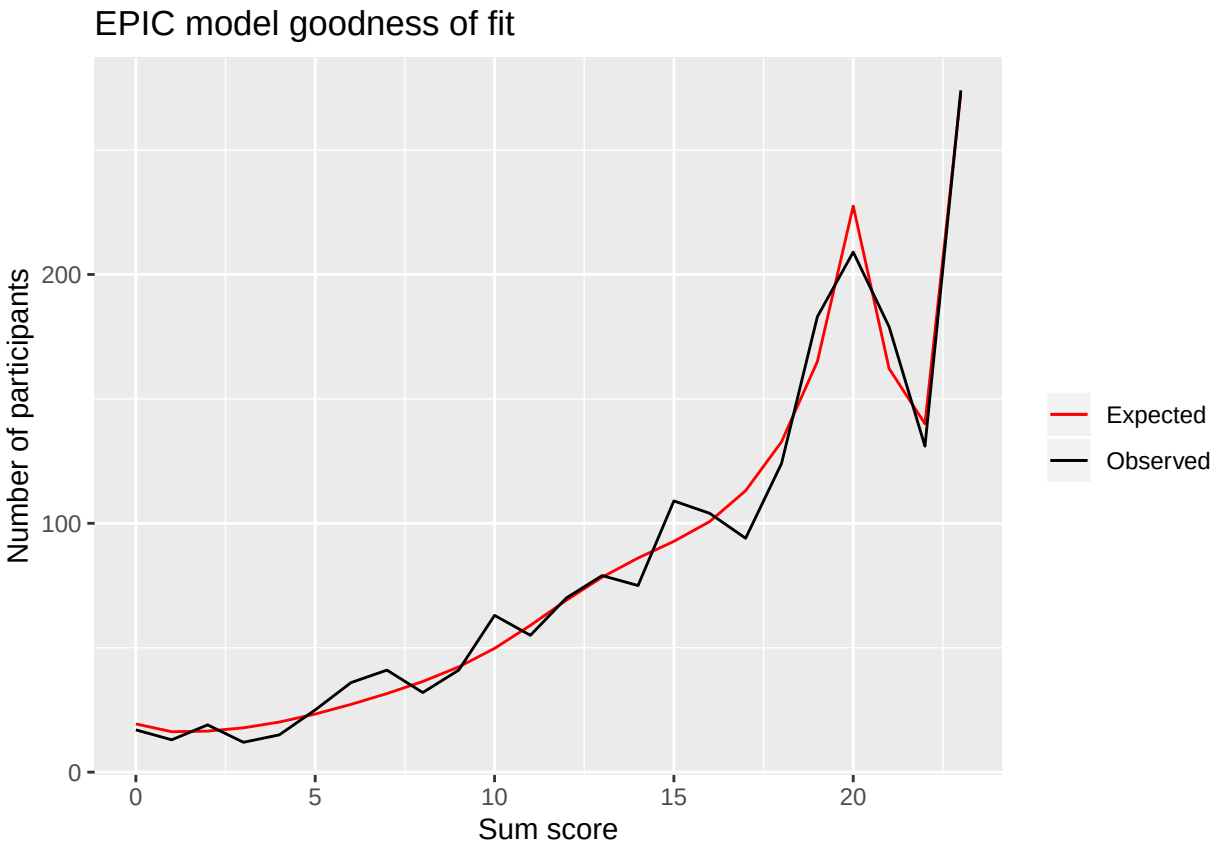
```
iief_gof = ggplot(data = iief_scores, aes(x = Sum.Scores)) +
  geom_line(aes(y=expected, colour='Expected')) +
  geom_line(aes(y=observed, colour='Observed')) +
  xlab('Sum score') + ylab('Number of participants') +
  ggtitle('IIEF model goodness of fit') +
  scale_colour_manual("", breaks = c("Expected", "Observed"), values = c("red", "black"))
```

```
epic_gof = ggplot(data = epic_scores, aes(x = Sum.Scores)) +
  geom_line(aes(y=expected, colour='Expected')) +
  geom_line(aes(y=observed, colour='Observed')) +
  xlab('Sum score') + ylab('Number of participants') +
  ggtitle('EPIC model goodness of fit') +
  scale_colour_manual("", breaks = c("Expected", "Observed"), values = c("red", "black"))

print(iief_gof)
```



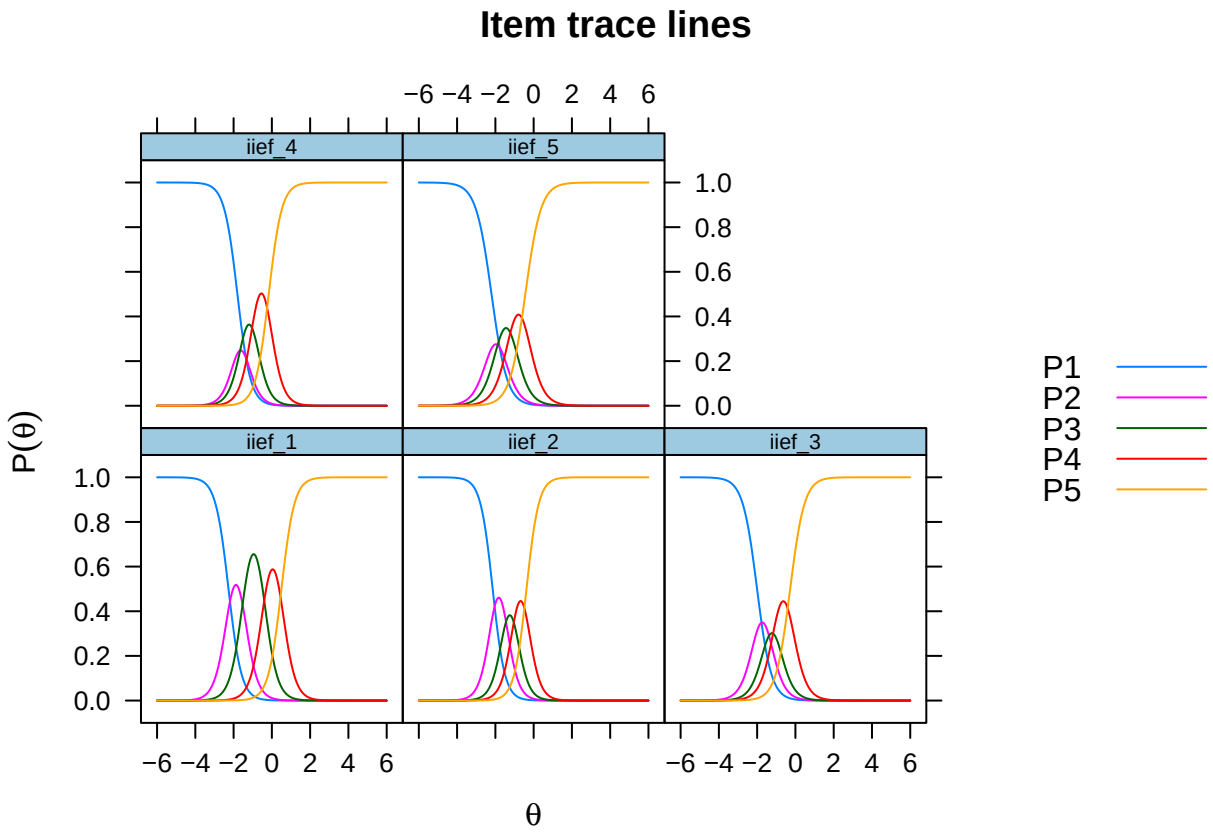

```
print(epic_gof)
```



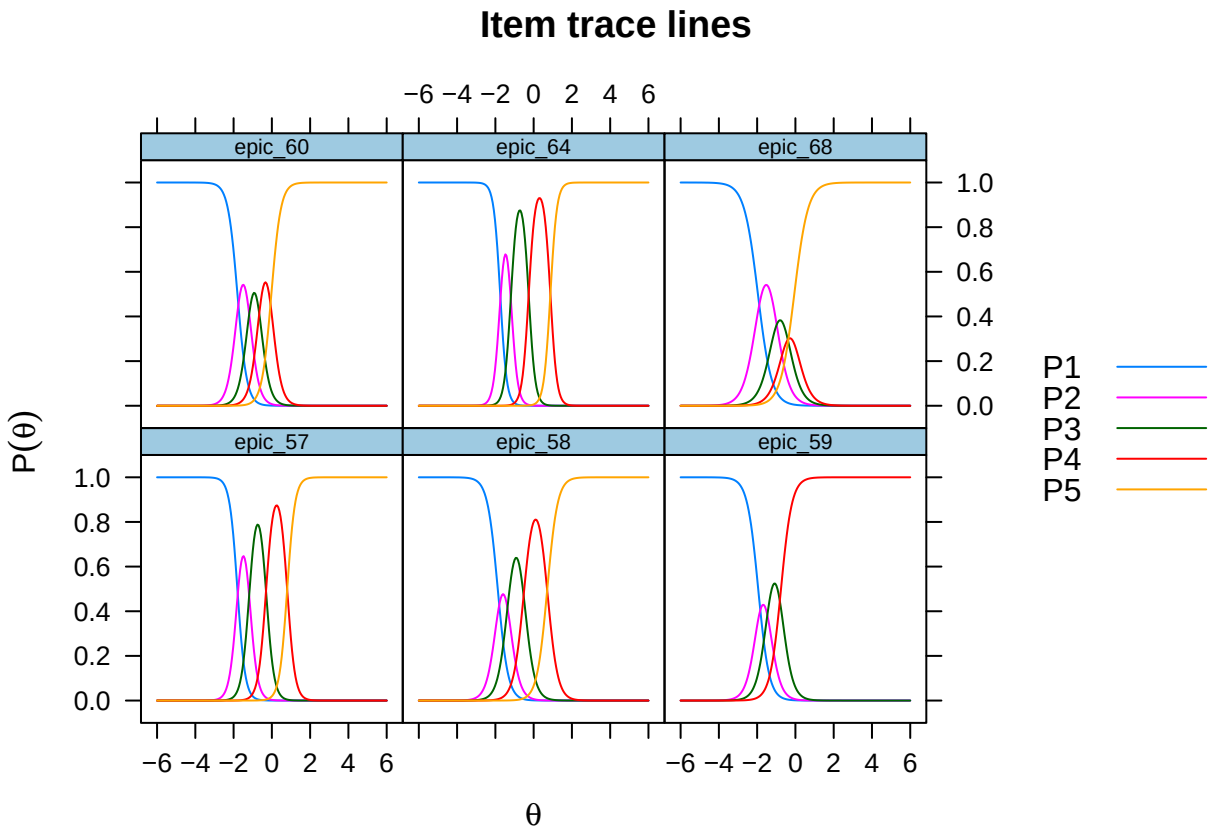
Based on only these plots we can be reasonably confident in the suitability of our data for IRT. Normally testing the two main IRT assumptions of unidimensionality and local dependence involves steps such as confirmatory factor analysis, but for the sake of this example we'll skip this step.

We can examine the reliability of individual items by ploting item trace curves:

```
plot(model_iief, type='trace')
```




```
plot(model_epic, type='trace')
```



These plots represent the probability of responding in each response category for each item for a given level of theta. Each curve in each of the above boxes represents a response category to the corresponding item. In ideally configured items, each of these curves will be steep and separable, corresponding to equally sized segments of the theta range.

We can see from these plots that some items seem to provide more information (reliability in the IRT context) than others. IIEF Q1 and Q2, for example, appear to provide more information than Q4 and Q5, as there is no value of theta for which a respondent would respond in the second category for either of the latter items.

Let's continue on to the prediction step. We can use the expected test function from the mirt package to calculate the expected EPIC sum score for the theta level of each participant as estimated by the IIEF model. We then calculate the mean absolute error between the predicted and osberved EPIC scores for each participant.

```
df$predicted_epic = expected.test(model_epic, as.matrix(iief_scores_full$Theta,,1))
mean(abs(df$predicted_epic-df$Sum_epic))
```

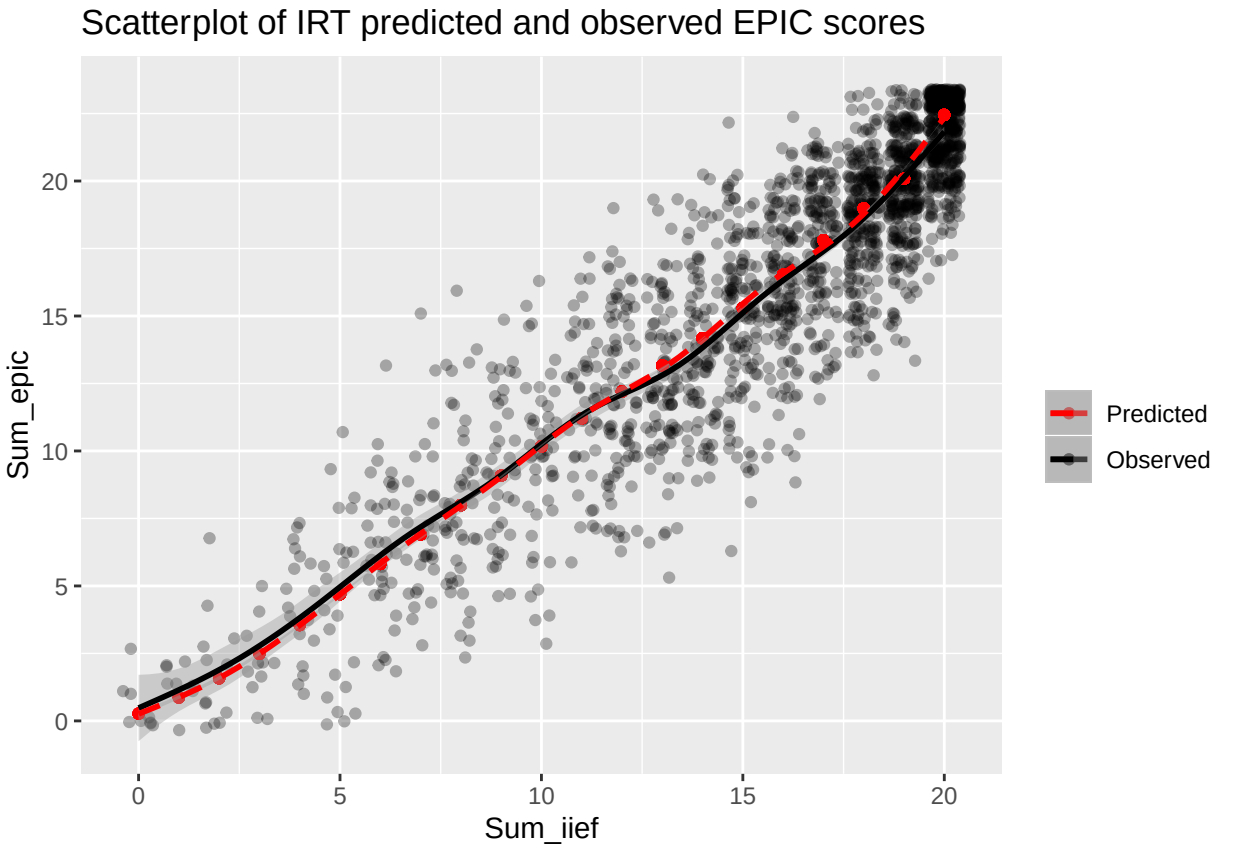
```
## [1] 1.727663
```

When a generalized additive model is fit to the IRT predicted and observed scores, we see that the curves are nearly identical:

```
ggplot(data = df, aes(x=Sum_iief)) + geom_jitter(aes(y = Sum_epic, colour='Observed'), alpha=.3) +
  geom_point(aes(y = predicted_epic, colour='Predicted')) +
  geom_smooth(aes(y = Sum_epic, colour='Observed')) +
  geom_smooth(aes(y = predicted_epic, colour='Predicted'), linetype='dashed') +
  ggtitle('Scatterplot of IRT predicted and observed EPIC scores') +
  scale_color_manual("",
                     breaks = c("Predicted", "Observed"),
```

```
                    values = c('black', 'red'))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



For the above predictions, we implemented the expected a-posteriori sum score method to associate a sum score with each theta value. We also have the option to use the expected a-posteriori method, which estimates a theta value for each response pattern and could perhaps yield better results.

```
iief_scores = as.data.frame(fscores(model_iief, method = "EAP", full.scores = TRUE))
iief_response_patterns = as.data.frame(fscores(model_iief, method = "EAP", full.scores = FALSE))
epic_response_patterns = as.data.frame(fscores(model_epic, method = "EAP", full.scores = FALSE))
```

```
head(iief_response_patterns)
```

```
##   iief_1 iief_2 iief_3 iief_4 iief_5        F1     SE_F1
## 1      0      0      0      0      0 -2.734569 0.4284159
## 2      0      0      0      0      1 -2.425749 0.3425045
## 3      0      0      0      0      2 -2.329982 0.3491519
## 4      0      0      0      1      2 -2.062518 0.3054930
## 5      0      0      0      2      1 -2.089656 0.3144542
## 6      0      0      0      3      2 -1.904575 0.3438868
```

An alternative to visualizing item reliability using item trace curves might be to plot the participant's estimated theta value against the responses in each category for each item. This gives an idea of which items are more influential in the theta estimation, as the response categories for these items should represent adjacent and seperable segments of the theta range, that are ideally also narrow and with smaller standard deviations.

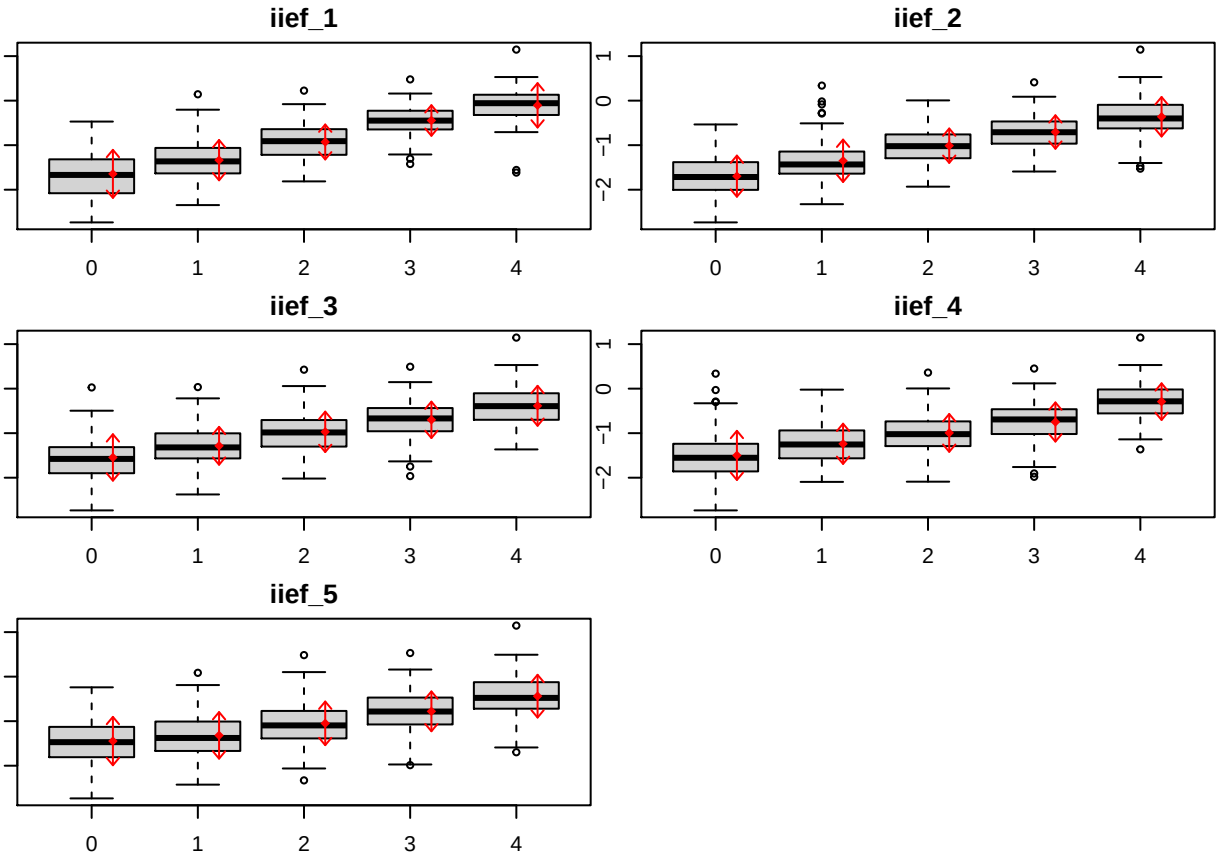


Figure 1: Box plots for IIEF items of EAP-estimated theta level of participants responding in each category. Red arrows indicate standard deviation

```
old.par <- par(mfrow=c(3, 2))
par(mar=c(2,1,2,1))

for (index in 1:(ncol(iief_response_patterns)-2)){
  bp = boxplot(iief_response_patterns$F1 ~ iief_response_patterns[,index], col = "lightgray")
  title(colnames(iief_response_patterns[index]))
  means <- tapply(iief_response_patterns$F1, iief_response_patterns[,index], mean)
  sds <- tapply(iief_response_patterns$F1, iief_response_patterns[,index], sd)
  displace <- 0.2 + seq(bp$n)
  points(displace, means, col = "red", pch = 18)
  arrows(displace, means - sds, displace, means + sds,
         code = 3, col = "red", angle = 40, length = .05)
}
par(old.par)
```

These plots seem to indicate that the IIEF items are relatively consistent, with perhaps some lopsided response categories in Q1 and Q2. The EPIC plots display more inconsistency in items, particularly regarding Q6. The response categorie theta ranges here are all very large, as well as the corresponding standard deviations. EPIC Q6 being a weaker item is also corroborated by the item trace curves discussed above.

How do the predictions using this method compare against EAPsum?

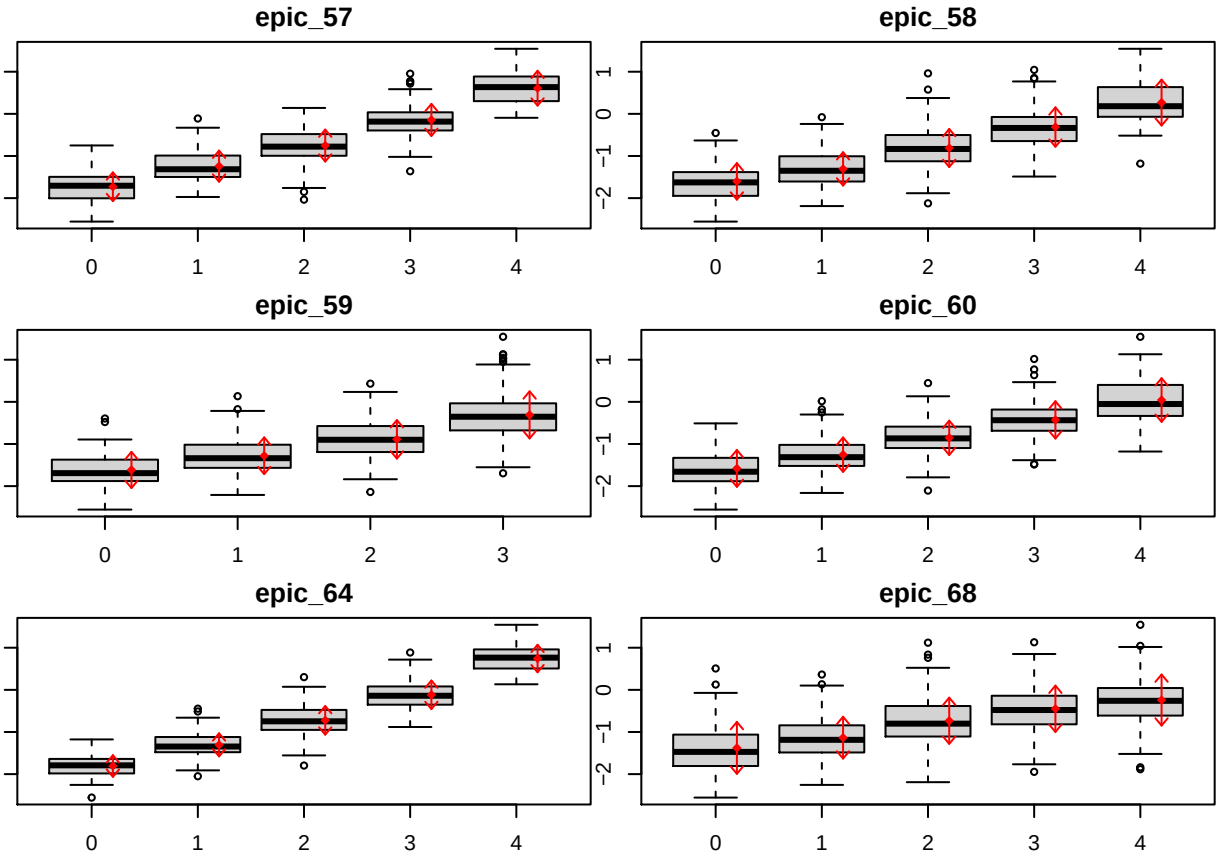


Figure 2: Box plots for EPIC items

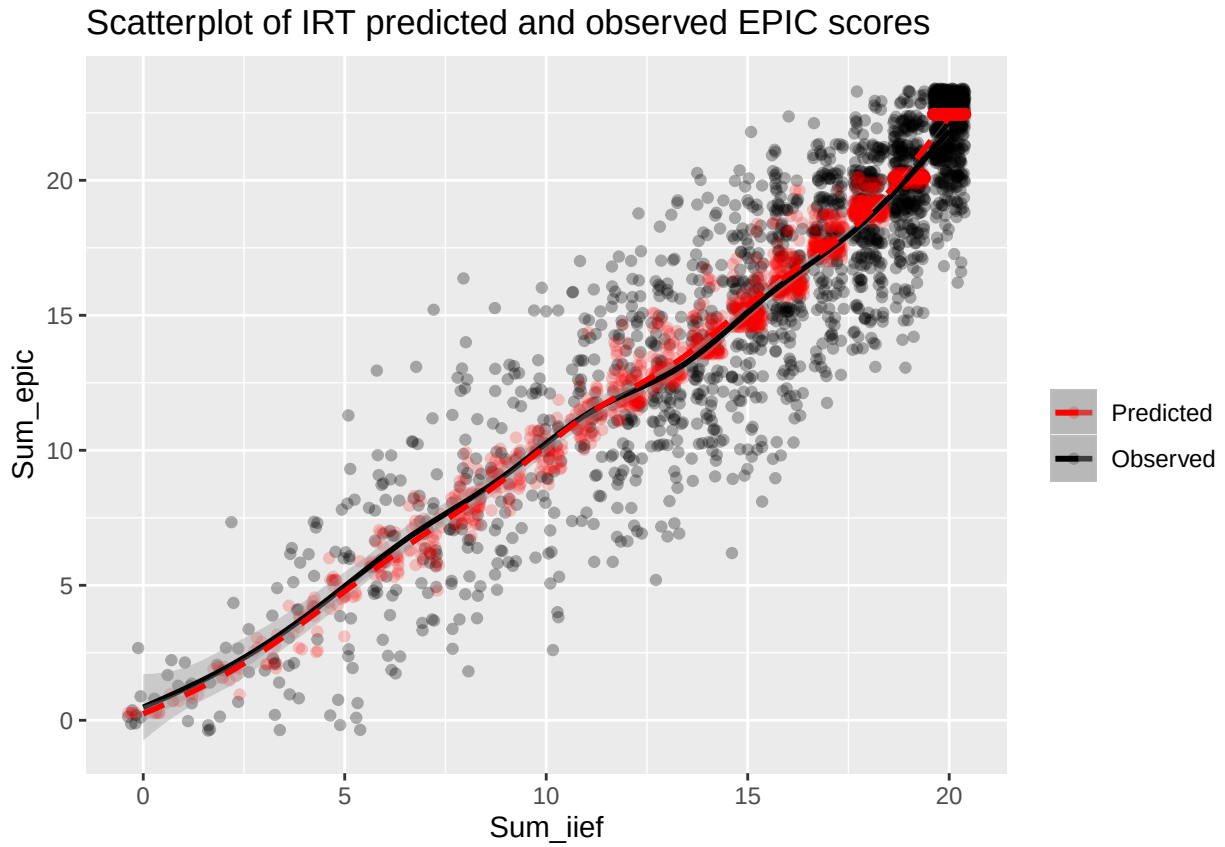
```
df$predicted_epic = expected.test(model_epic, as.matrix(iief_scores$F1,,1))

#Mean absolute error:
mean(abs(df$predicted_epic-df$Sum_epic))
```

```
## [1] 1.703803
```

As we can see, this method results in a slight improvement in prediction error (MAE = 1.70 against 1.73). How do the predictions look when plotted?

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Our plot now reflects that there are multiple possible predictions for each discrete IIEF sum score.

How do the IRT predicitons compare against more traditional regression models? Let's try out a GAM using the IIEF sum score as a single predictor:

```
library(gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.16
```

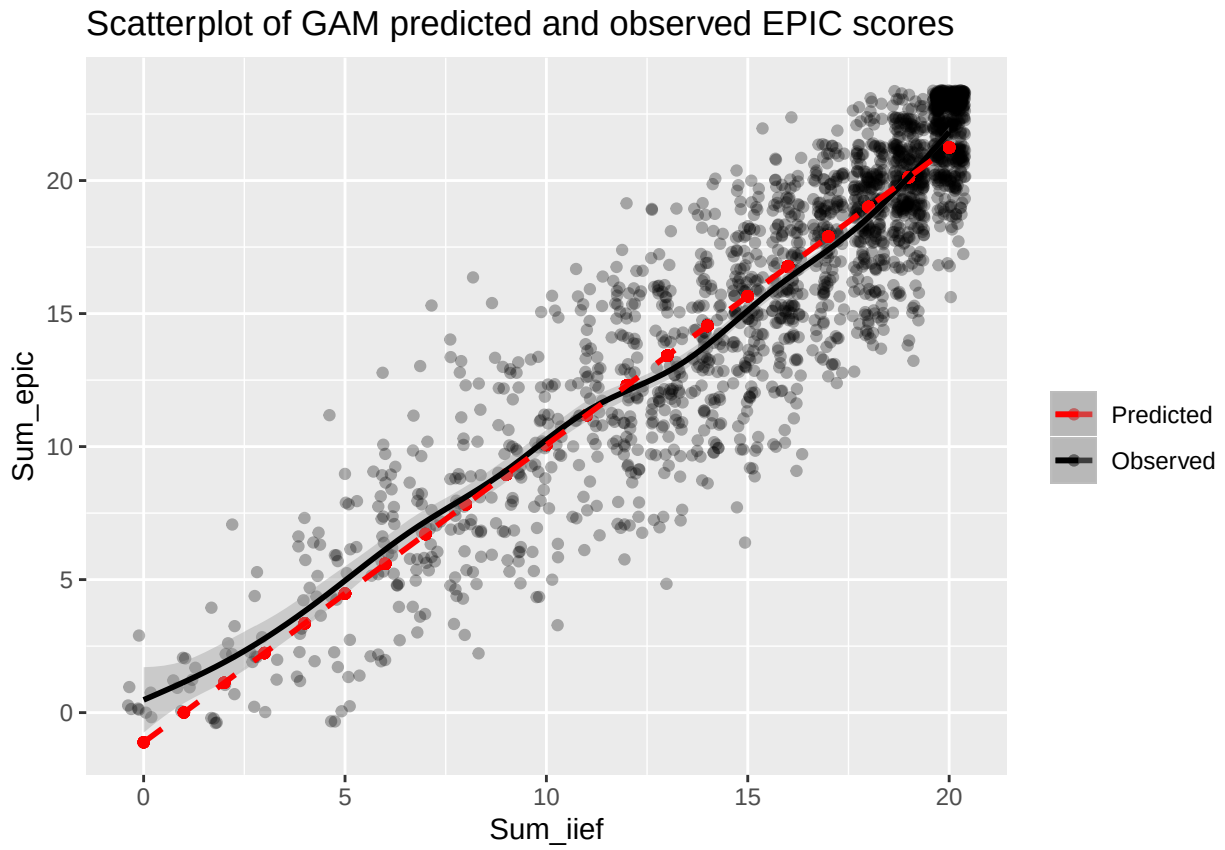
```
mod = gam(Sum_epic~Sum_iief, data=df)
df$predicted_epic = predict(mod)
```

```r
#Mean absolute error:
mean(abs(df$predicted_epic-df$Sum_epic))
```

```
## [1] 1.812583
```

It turns out that our IRT was a bit more precise (MAE of 1.70 against 1.81). Finally, let's see how this looks when plotted:

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



We probably would have better results if we performed a model search across different combinations of items as predictors, but at least we now have a basis for comparison for our IRT model.