TRI: modelos politómicos

Manuel Torres Acosta

Índice

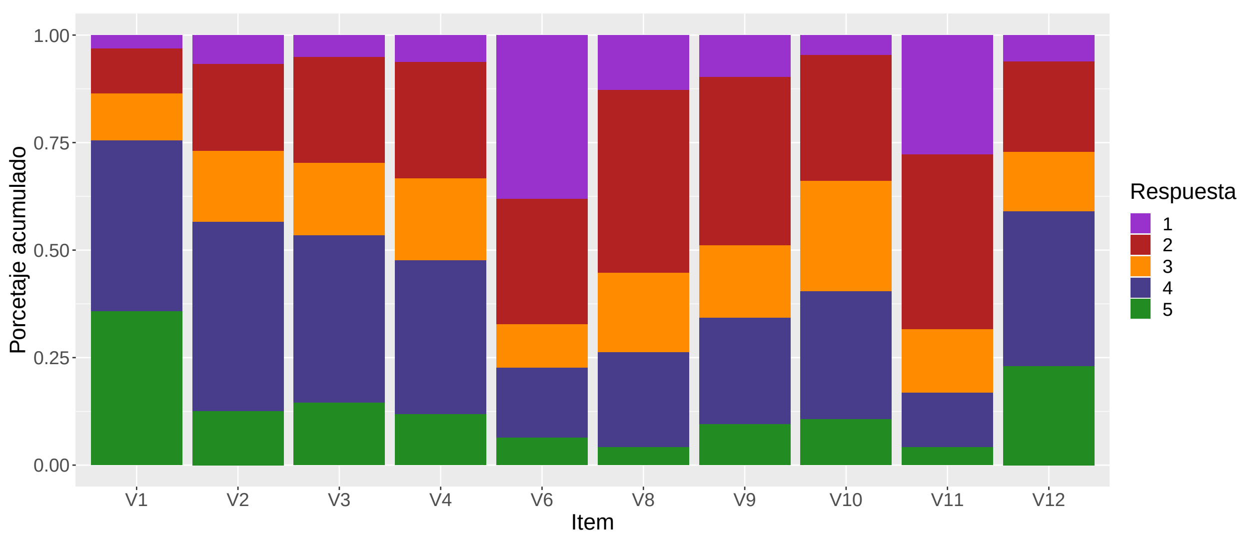
Tarea

La tarea de ésta semana consiste en estudiar las propiedades de un test a través de la TCT y de un modelo politómico de TRI.

Visualización de las respuestas

Ítems: 1, 2, 3, 4, 6, 8, 9, 10, 11, 12

```
Datos <- read.spss(Rutas$Datos,
                   to.data.frame = TRUE)
Datos <- Datos[, idxItems]

#Quitamos los perdidos
Datos <- Datos[rowSums(Datos == 9) == 0, ]

#Obtenemos las frecuencias de respuesta
Frecuencias <- as.data.frame(response.frequencies(Datos))
Frecuencias$miss <- NULL
Frecuencias$Item <- rownames(Frecuencias)

DatosGrafico <- melt(Frecuencias, id.vars = "Item")
DatosGrafico$Item <- factor(DatosGrafico$Item,
                            levels = Frecuencias$Item)

#Visualizamos las frecuencias
ggplot(DatosGrafico, aes(x = Item,
                         y = value,
                         fill = variable)) +
  geom_bar(position = "fill",
           stat = "identity") +
  ylab("Porcetaje acumulado") +
  scale_fill_manual(values = Palette) +
  guides(fill=guide_legend(title = "Respuesta")) +
  theme(text = element_text(size = PlotFont / 1.2))
```

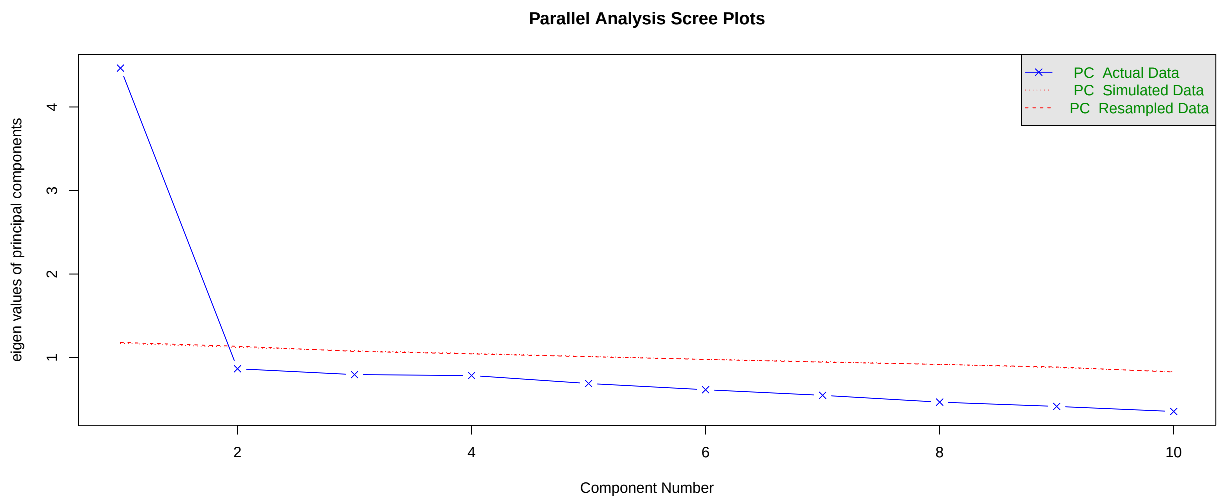


Teoría Clásica de los Tests

Consistencia interna

Utilizamos la función `itemAnalysis()` para obtener el alfa de Cronbach y el efecto de los ítems sobre el mismo. También podemos utilizar el análisis paralelo para determinar si existe una única dimensión y hacer un análisis factorial para comprobarlo.

```
fiabilidad <- itemAnalysis(Datos)

fa.parallel(Datos, fa = "pc")
```




```
Factor1 <- fa(Datos,
              fm = "uls")
Temp <- data.frame(unclass(Factor1$loadings),
                   h2 = Factor1$communalities,
                   u2 = Factor1$uniqueness)
Temp <- Temp[rev(order(Temp$ULS1)), ]
```

En nuestro caso, todos los ítems contribuyen al aumento de alfa, y el indicador adopta el valor 0.859.

El análisis paralelo indica claramente que en nuestros datos solo existe una dimensión.

Además, como podemos ver en la siguiente tabla, todas las comunalidades son suficientemente altas, y las correlaciones adecuadas.

```r
PropItems <- fiabilidad$itemReport
PropItems$PesoFactorial <- Temp$ULS1
kable(PropItems)
```

itemName	itemMean	pBis	bis	alphaIfDeleted	PesoFactorial
V1	3.943971	0.4176849	0.4616404	0.8569674	0.7460441
V2	3.353228	0.5410674	0.5789263	0.8473615	0.7350931
V3	3.331303	0.5185916	0.5514506	0.8492549	0.7310208
V4	3.198538	0.6549648	0.6930227	0.8376457	0.7195646
V6	2.236297	0.6809310	0.7489915	0.8345594	0.5859163
V8	2.623630	0.5298228	0.5650550	0.8482513	0.5855185
V9	2.851401	0.6689980	0.7112441	0.8361836	0.5700183
V10	3.124239	0.6662971	0.7017800	0.8371842	0.5526424
V11	2.247259	0.5407731	0.5850923	0.8473707	0.4762413
V12	3.485993	0.4447970	0.4753808	0.8561758	0.4480420

Con todos éstos datos podemos concluir que existe unidimensionalidad en nuestro test y que todos los ítems contribuyen a dicha dimensión.

Fiabilidad (técnica de las dos mitades)

```r
#Dividimos el test en dos mitades
Temp <- 1:ncol(Datos) %% 2 == 0

#Puntuaciones de las mitades
Mitades <- list(
  Pares = rowSums(Datos[, Temp]),
  Impares = rowSums(Datos[, !Temp])
)
FiabilidadMitad <- cor(Mitades$Pares,
                       Mitades$Impares)
FiabilidadTotal <- spearman.brown(FiabilidadMitad, 2 , "n")$r.new
```

El coeficiente de fiabilidad para el test mitad vale **0.72**. Tras aplicar la corrección de Spearman Brown para obtener la fiabilidad del test con la longitud original el valor asciende a **0.83**.

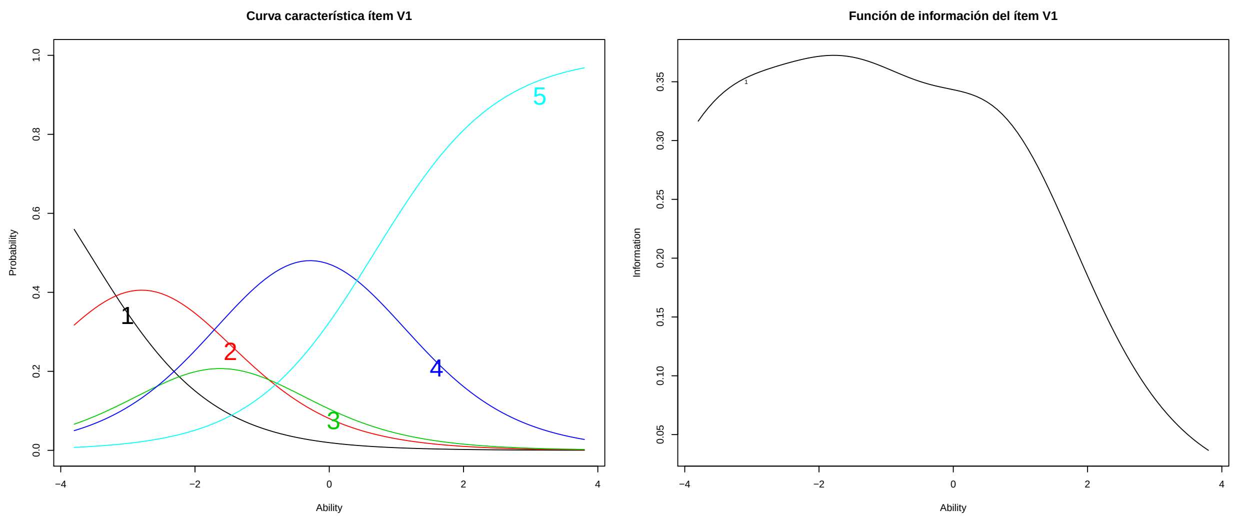
Con éstos datos, si deseamos obtener un coeficiente de fiabilidad de 0.9 necesitamos un total de 18 ítems.

Teoría de Respuesta al Ítem

Dado que se cumple el supuesto de unidimensionalidad, podemos ajustar los modelos politómicos de la TRI.

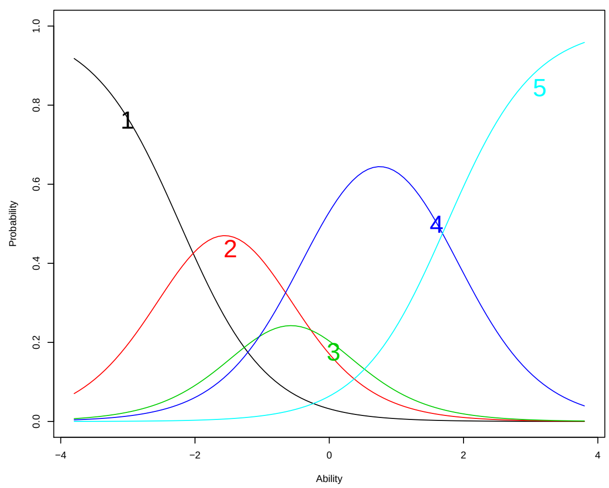
Para ello debemos tener cuidado de que en todos los ítems no queden categorías sin respuestas. Si ocurriese ésto tendríamos que recodificar las categorías siguientes para hacer como si no existiese la que no ha tenido respuesta [1].

```
fit1 <- grm(Datos)

layout(
  matrix(c(1, 1, 1, 1, 2, 2, 2, 2),
  nrow = 2,
  byrow = FALSE)
)

for(i in 1:ncol(Datos)){
  plot(fit1, items = i,
       main = paste("Curva característica ítem",
                    colnames(Datos)[i]),
       cex = 2.5)
  plot(fit1, type = "IIC", items = i,
       main = paste("Función de información del ítem",
                    colnames(Datos)[i]))
}
```

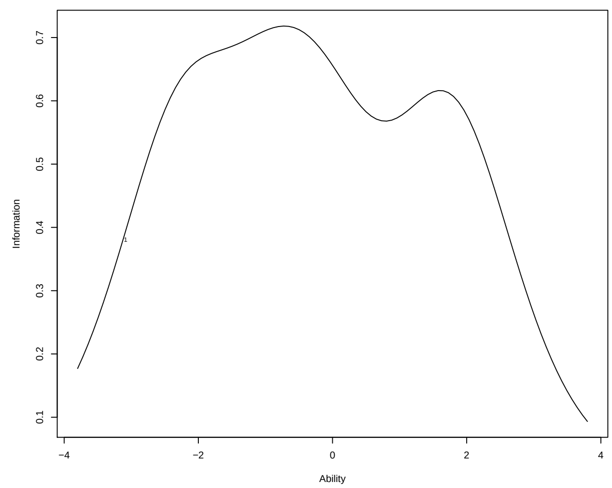


[1]Ejemplo: falta la categoría 4 en una escala de 1-7, las categorías 5, 6, 7 pasarían a ser 4, 5, 6

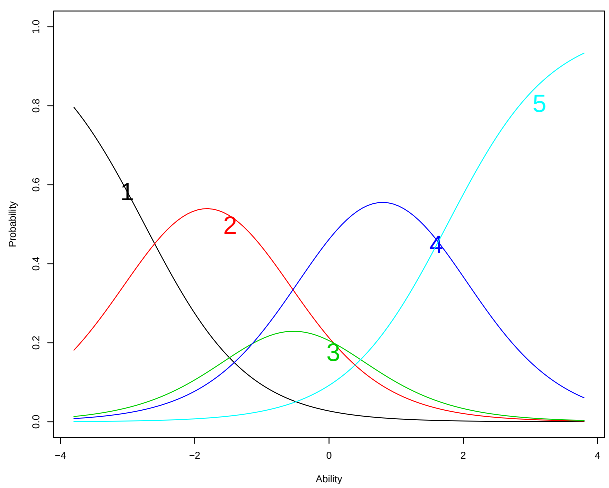
Curva característica ítem V2



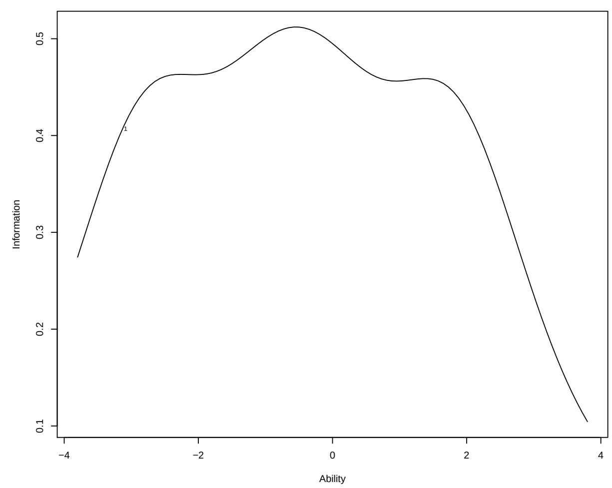
Función de información del ítem V2



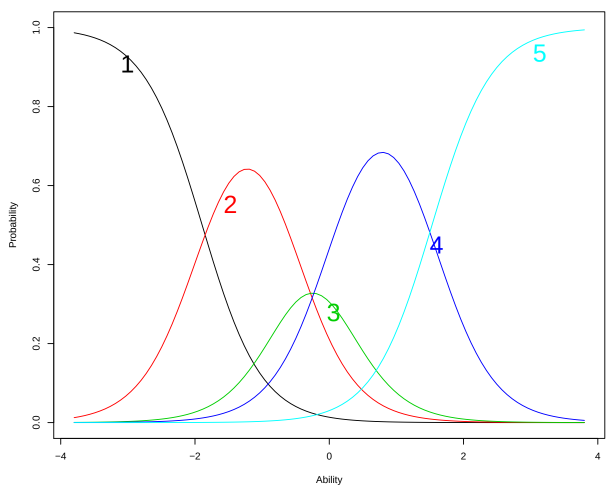
Curva característica ítem V3



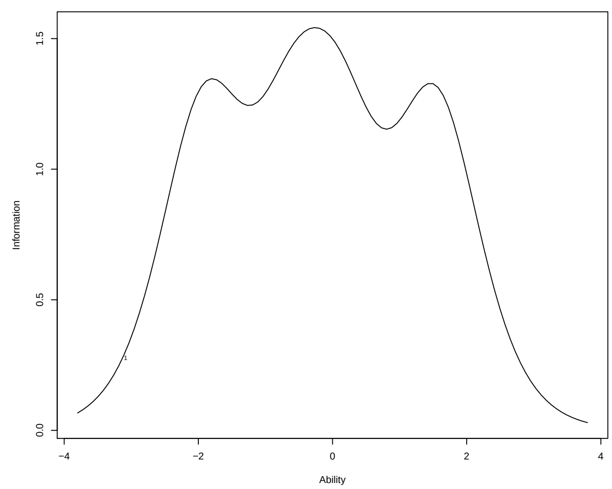
Función de información del ítem V3

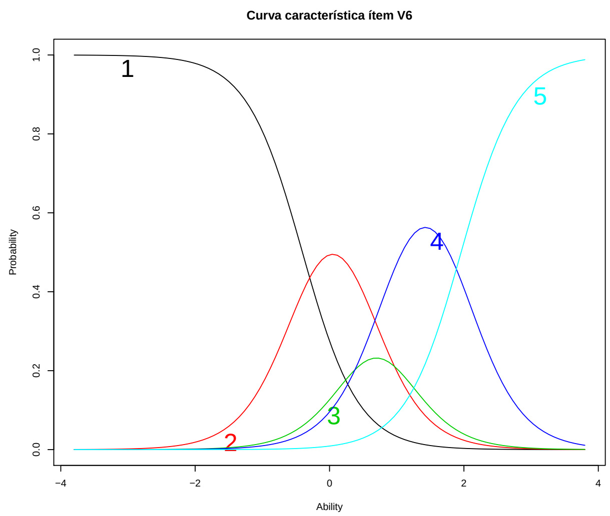


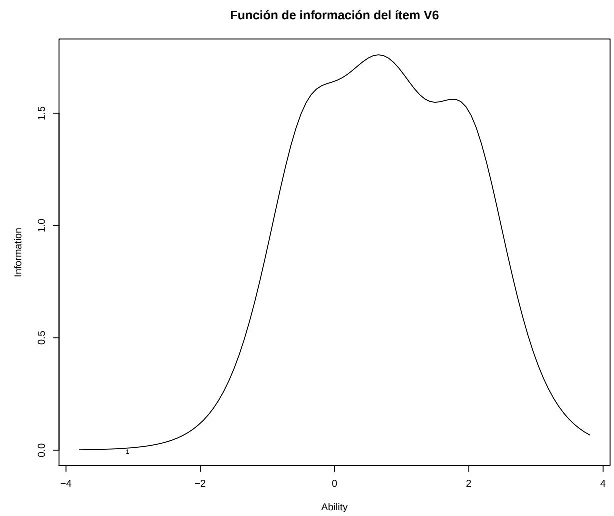
Curva característica ítem V4

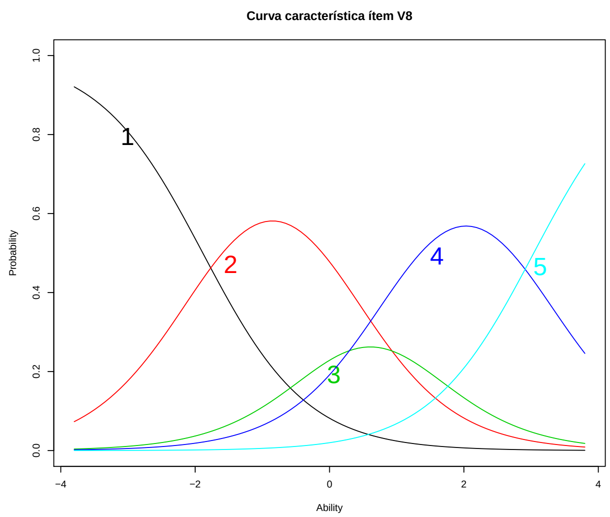


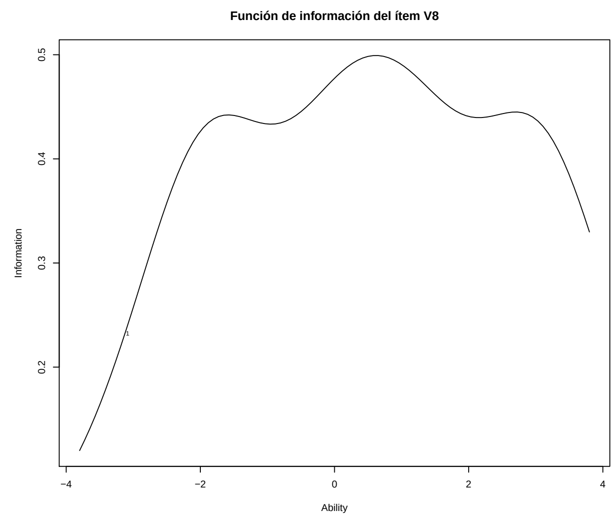
Función de información del ítem V4

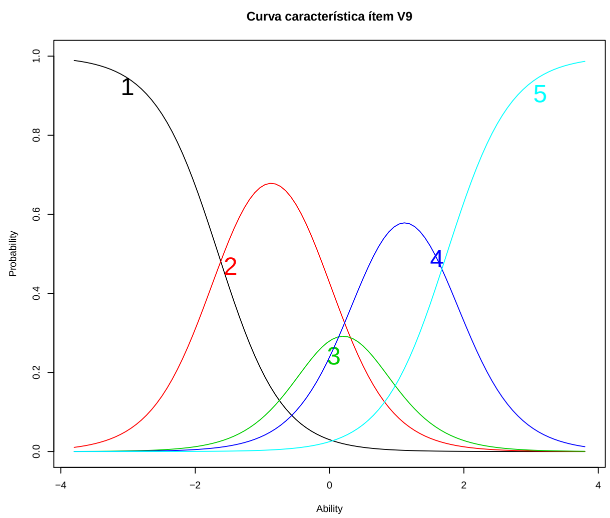


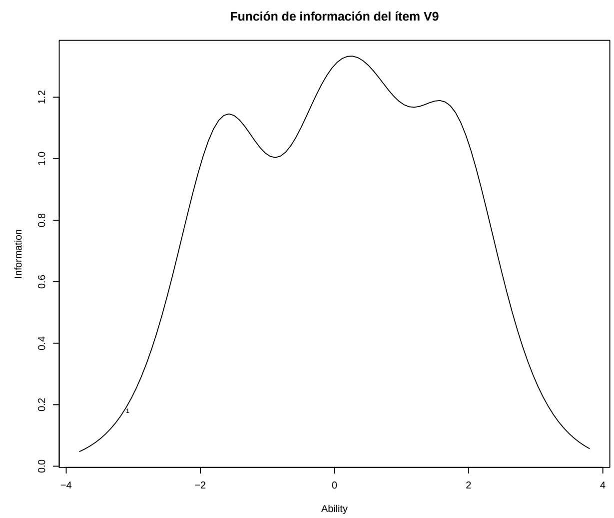

Curva característica ítem V6
Probability
Ability
1
2
3
4
5


Función de información del ítem V6
Information
Ability

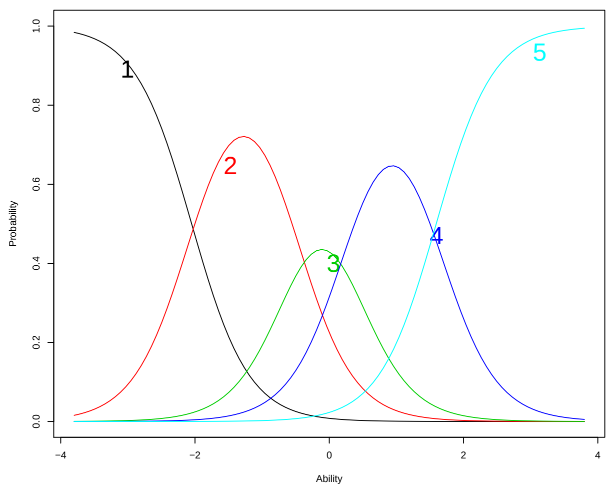

Curva característica ítem V8
Probability
Ability
1
2
3
4
5


Función de información del ítem V8
Information
Ability

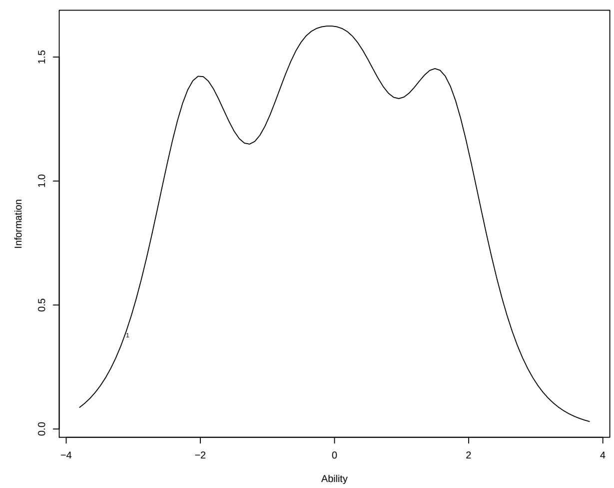

Curva característica ítem V9
Probability
Ability
1
2
3
4
5


Función de información del ítem V9
Information
Ability

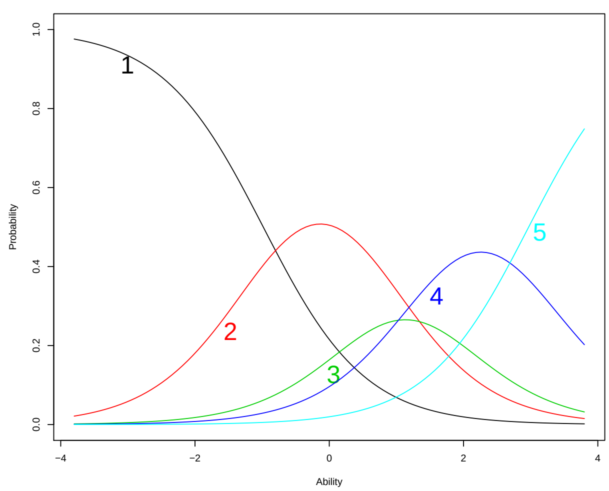
Curva característica ítem V10



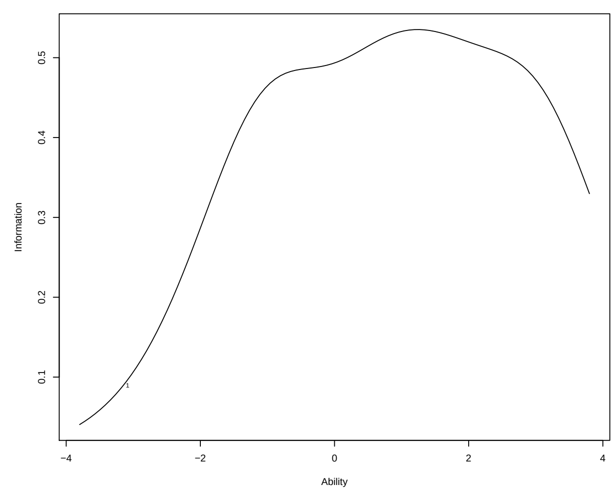
Función de información del ítem V10



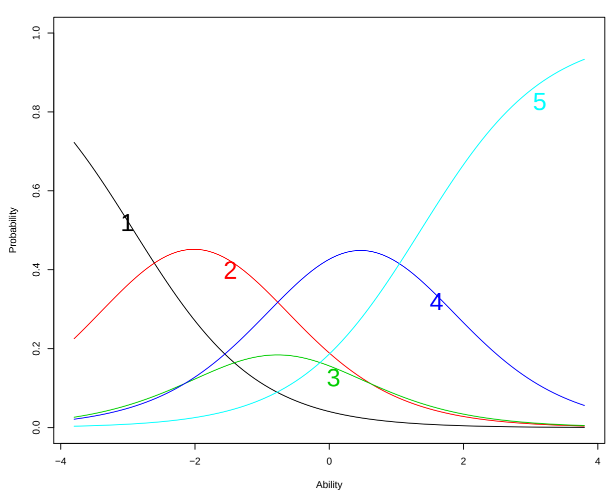
Curva característica ítem V11



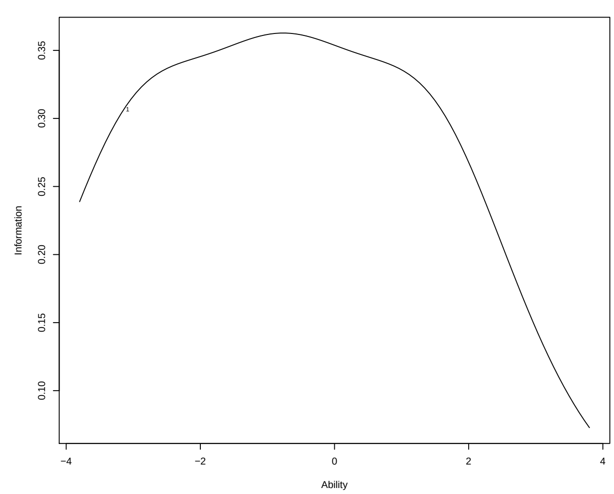
Función de información del ítem V11



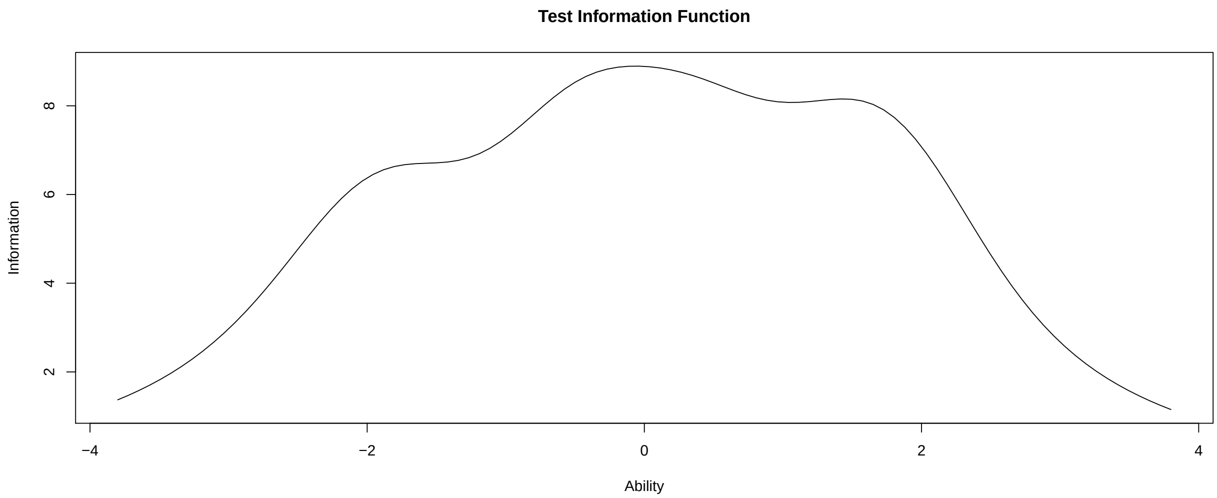
Curva característica ítem V12



Función de información del ítem V12



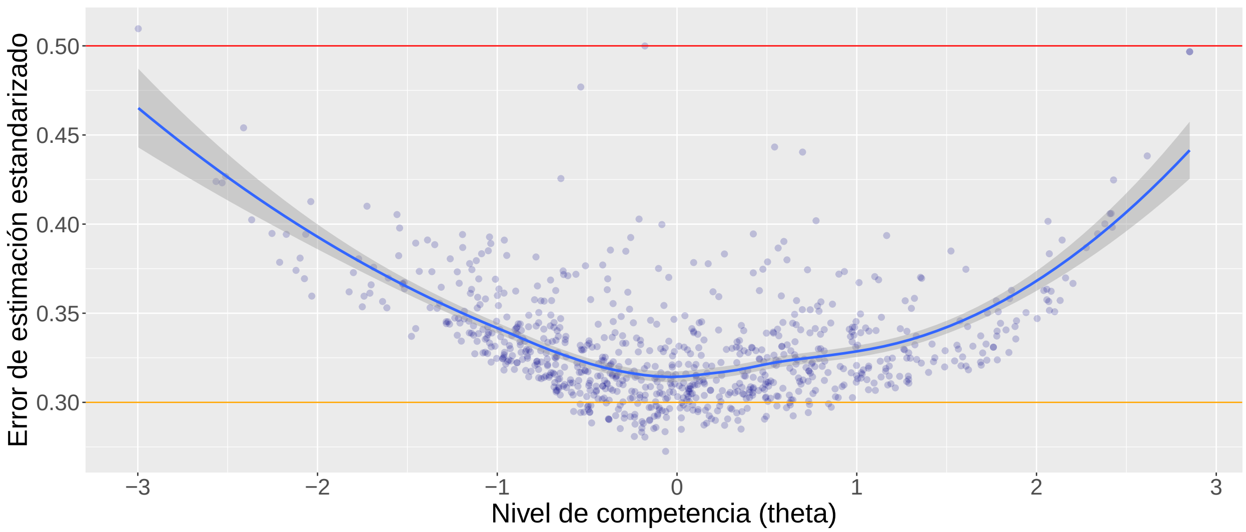
El modelo también nos permite visualizar las curvas características operantes para cada ítem, pasándole el parámetro `type = 'OCCu'` a la función plot, pero no lo haremos por cuestiones de espacio. Podemos representar la función de información del test completo.



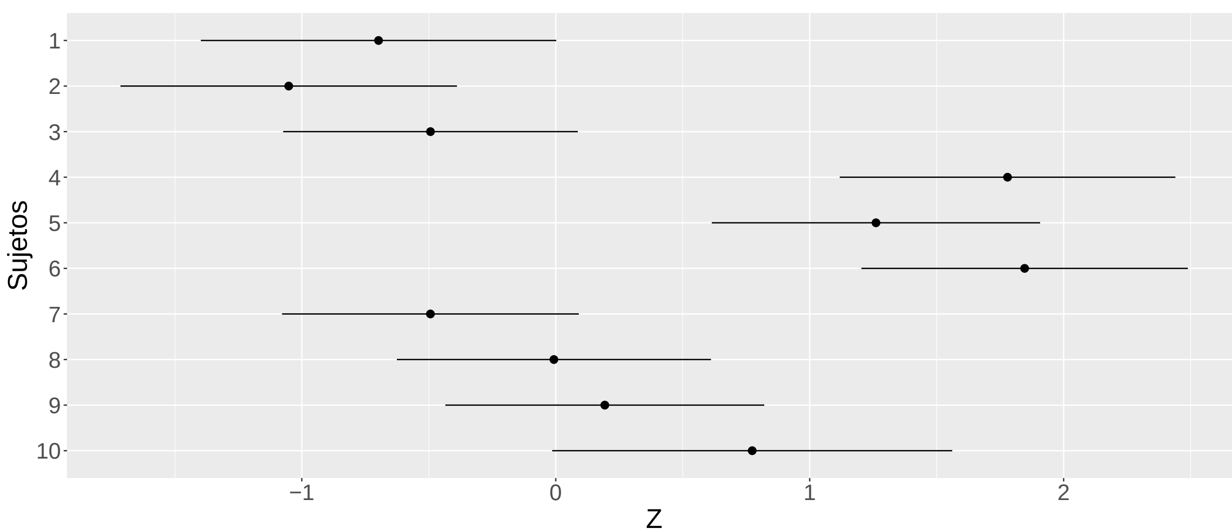
Estimación de las puntuaciones

```
Zest <- factor.scores.grm(object = fit1,
                          resp.patterns = Datos)
Zest <- Zest$score.dat
Zest <- Zest[, c("z1", "se.z1")]
colnames(Zest) <- c("Z", "SE")
```

En el siguiente gráfico podemos ver cómo varía el error de medición en función del nivel de competencia (rasgo) de los sujetos. Podemos ver que la precisión es mayor para valores de competencia medios, pero en todo el rango de niveles obtenemos valores aceptables.

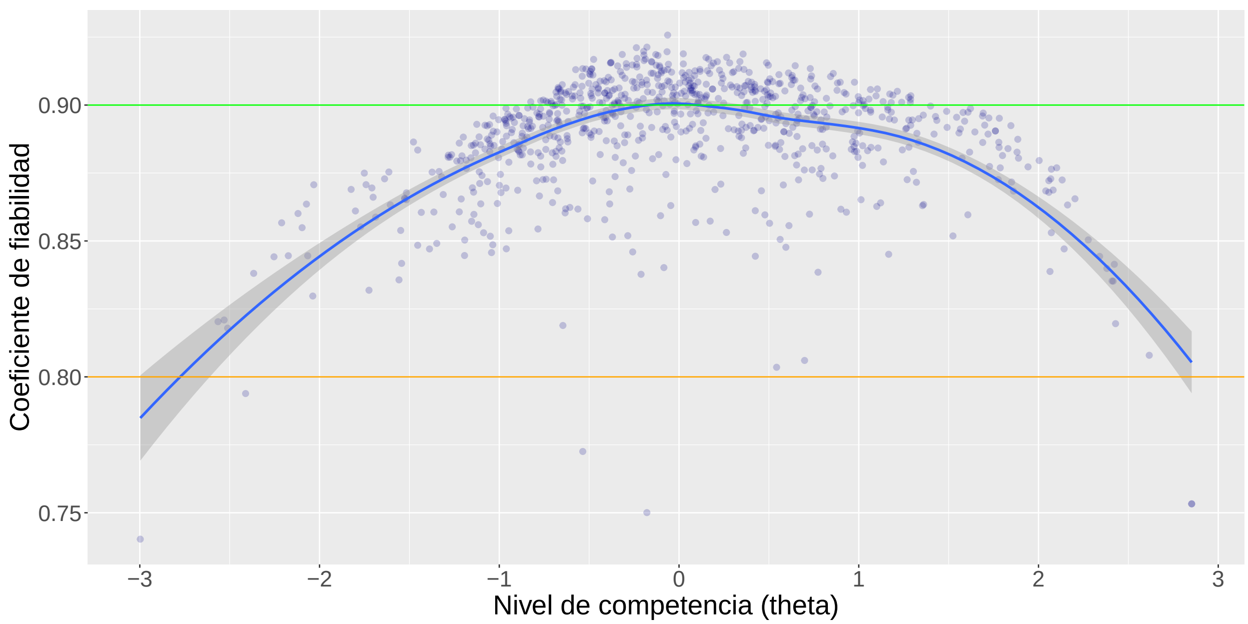


También podemos construir intervalos de confianza al 95 % entorno a las puntuaciones.



Podemos representar la precisión del instrumento para los distintos niveles de θ en términos de fiabilidad de la TCT (línea azul en el gráfico siguiente):

```
ggplot(Zest, aes(x = Z, y = 1 - (SE ^ 2))) +
  geom_point(alpha = 0.2,
             size = 2,
             color = "darkblue") +
  geom_smooth() +
  xlab("Nivel de competencia (theta)") +
  ylab("Coeficiente de fiabilidad") +
  geom_hline(yintercept = 0.9, color = "green") +
  geom_hline(yintercept = 0.8, color = "orange") +
  theme(text = element_text(size = PlotFont))
```



Vemos que tenemos una fiabilidad excelente para valores de θ cercanos a la media, y mantenemos unos niveles de error aceptables en todo el rango de puntuaciones. La media del error estandarizado vale 0.33, por lo que el **coeficiente de fiabilidad** del test valdría **0.89**.

Discusión

Vemos que el coeficiente de fiabilidad estimado por la TCT es algo inferior al que nos proporciona la TRI.

Quizás sea porque en la curva de fiabilidad de la TRI observamos que la mayoría de la muestra se encuentra en la región en la que proporcionamos las mejores mediciones, mientras que la estimación de la TCT no puede identificar éste hecho.

Por tanto, ofrece el mismo error para todos los niveles de competencia, sobreestimando el error de los sujetos que están en torno a la media (donde medimos muy bien y está la mayoría).

No obstante, las mediciones de unidimensionalidad y comunalidad son comunes para ambos modelos.

En la práctica de modelos dicotómicos he realizado una comparación más detallada de los valores de los parámetros entre TCT y TRI.