Teoría de Respuesta al Ítem

Manuel Torres Acosta

Índice

Tarea

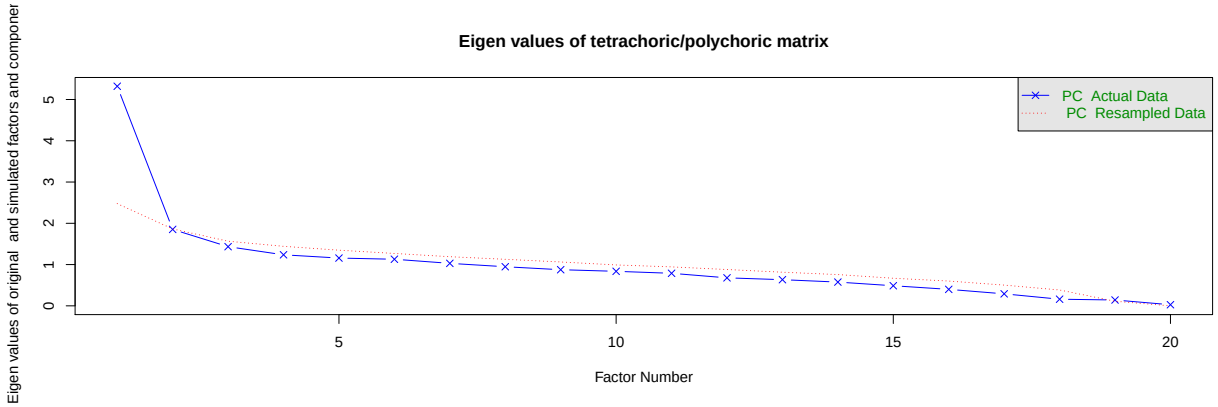
La tarea de ésta semana consiste en analizar las propiedades de los ítems de una prueba de inglés mediante los modelos de la TRI. Tenemos que recuperar los ítems seleccionados en la práctica anterior y aplicar las nuevas técnicas sobre ellos.

Selección de los ítems y unidimensionalidad

Recuperamos los ítems de la práctica anterior, sus índices son los siguientes:

2, 4, 45, 46, 51, 58, 60, 66, 68, 87, 115, 129, 151, 180, 183, 186, 193, 200, 201, 203

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

#Respuestas correctas a los items
Items <- read.csv(Rutas$Items,
                  stringsAsFactors = FALSE)[idxItems, ]
Respuestas <- Items$Respuesta

#Corregimos los items
Corregidos <- score(Datos, Respuestas,
                    output.scored = TRUE)
Temp <- as.data.frame(Corregidos$scored)
Puntuaciones <- unname(Corregidos$score)
rownames(Temp) <- names(Corregidos$score)
Corregidos <- Temp

#Aplicamos el analisis paralelo
Unidimensionalidad <- fa.parallel.poly(Corregidos,
                                       fa = "pc")
```



Vemos que un solo componente basta para explicar la variabilidad de los datos, por lo que podemos asumir unidimensionalidad. De todas formas, podemos evaluar las saturaciones de los items en un modelo unifactorial para comprobar que todos los items están relacionados con el factor.

```
Factor1 <- fa(Corregidos,
              fm = "uls",
              cor = "tet") #corr tetracorica al ser dicotomico
```

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

kable(Temp)
```

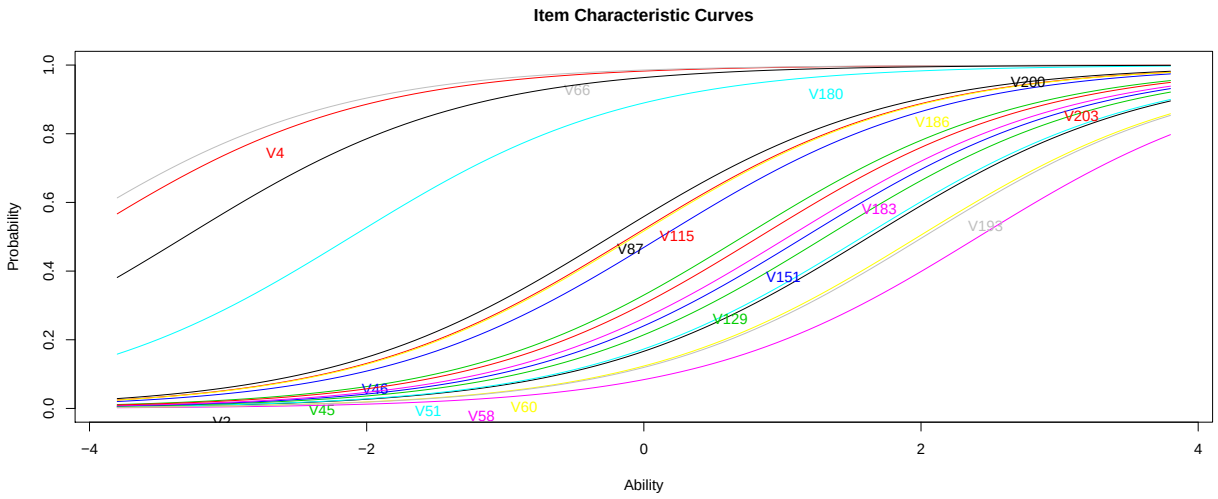
	ULS1	h2	u2
V151	0.7295745	0.5322777	0.4677210
V180	0.6382016	0.4073010	0.5926987
V46	0.6058444	0.3670468	0.6329526
V193	0.6009571	0.3611493	0.6388505
V129	0.5872454	0.3448574	0.6551428
V87	0.5476022	0.2998685	0.7001319
V115	0.5285474	0.2793624	0.7206376
V183	0.5108730	0.2609913	0.7390088
V2	0.5020789	0.2520833	0.7479168
V66	0.4924579	0.2425149	0.7574852
V186	0.4730081	0.2237369	0.7762633
V200	0.4727832	0.2235240	0.7764761
V4	0.3996093	0.1596876	0.8403124
V60	0.3991451	0.1593168	0.8406832
V58	0.3985534	0.1588446	0.8411552
V45	0.3575277	0.1278262	0.8721740
V203	0.3456515	0.1194750	0.8805251
V51	0.2969885	0.0882023	0.9117978
V201	0.2007796	0.0403123	0.9596875
V68	0.0753225	0.0056735	0.9943265

Podemos observar que las saturaciones más bajas las encontramos en los ítems 68 y 201, que son precisamente los mismos que en la práctica anterior hacían que disminuyese el alfa de cronbach. Los retiramos.

```
idxMantener <- !colnames(Datos) %in%
                 rownames(Temp)[Temp$ULS1 < 0.25]
Datos <- Datos[, idxMantener]
Respuestas <- Respuestas[idxMantener]
Items <- Items[idxMantener, ]
Corregidos <- Corregidos[, idxMantener]
```

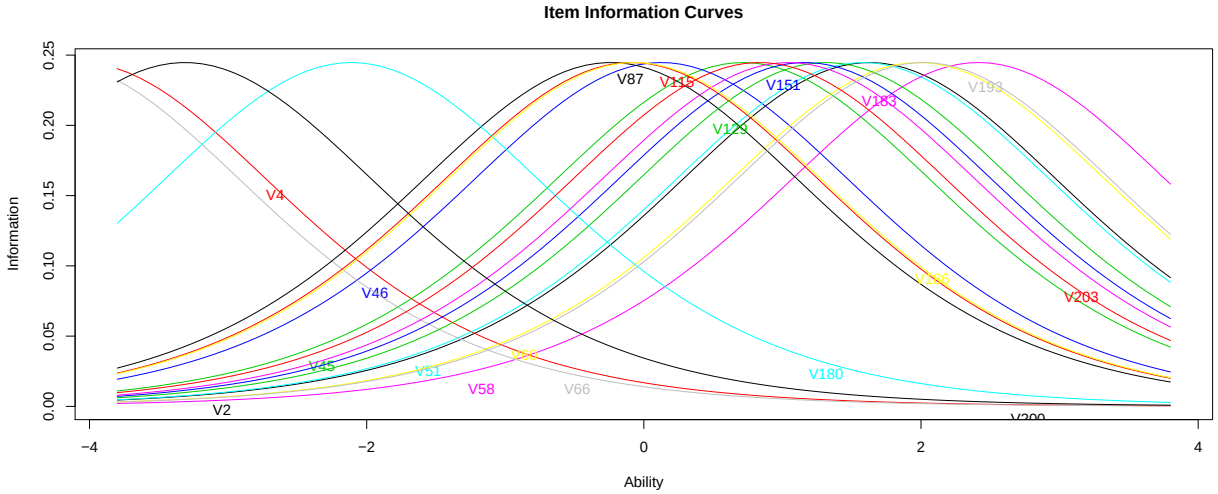
Modelo de un parámetro

```
fit1 <- rasch(Corregidos)
plot(fit1)
```



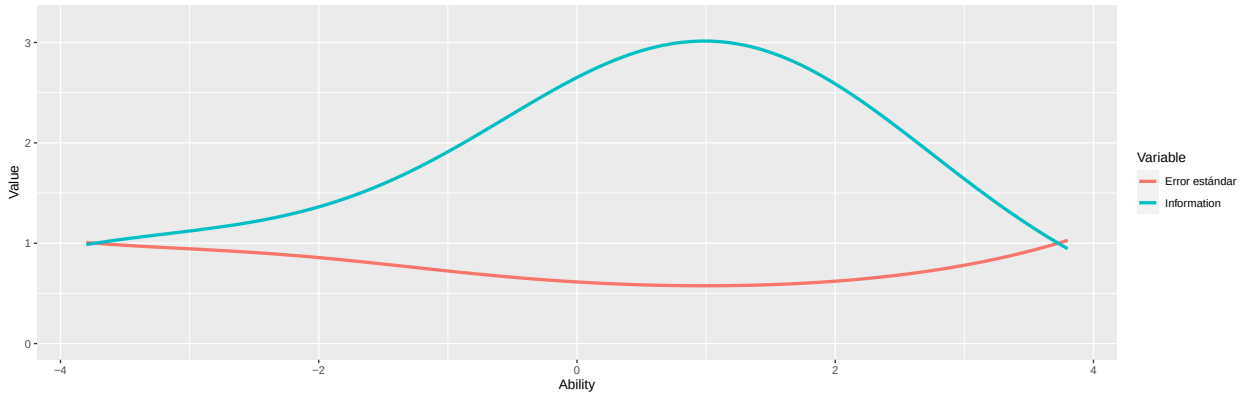
En el gráfico podemos ver las curvas características de los ítems, que al ser un modelo de un parámetro se diferencian por su dificultad. El índice de discriminación es común para todos y vale 0.9893564. La salida además nos ofrece algunas métricas de ajuste, que utilizaremos más adelante para comparar los modelos. Podemos representar también la función de información de los ítems.

```
plot(fit1, type = "IIC")
```



Cada ítem mide mejor a sujetos con una competencia cercana a su dificultad, de modo que ese será el máximo de cada curva. A continuación, vemos la función de información para el test completo.

```
plotInfo(fit1)
```



El test mide mejor a sujetos con un nivel de competencia de en torno a 1, por lo que seguramente tendrá una dificultad moderada.

Dado que hay algunos ítems con un nivel de dificultad muy bajo, los quitamos.

```
idxMantener <- !colnames(Datos) %in%
                  c("V4", "V66", "V180", "V200")
Datos <- Datos[, idxMantener]
Respuestas <- Respuestas[idxMantener]
Items <- Items[idxMantener, ]
Corregidos <- Corregidos[, idxMantener]
fit1 <- rasch(Corregidos) #Volvemos a ajustar el modelo
```

Modelo de dos parámetros

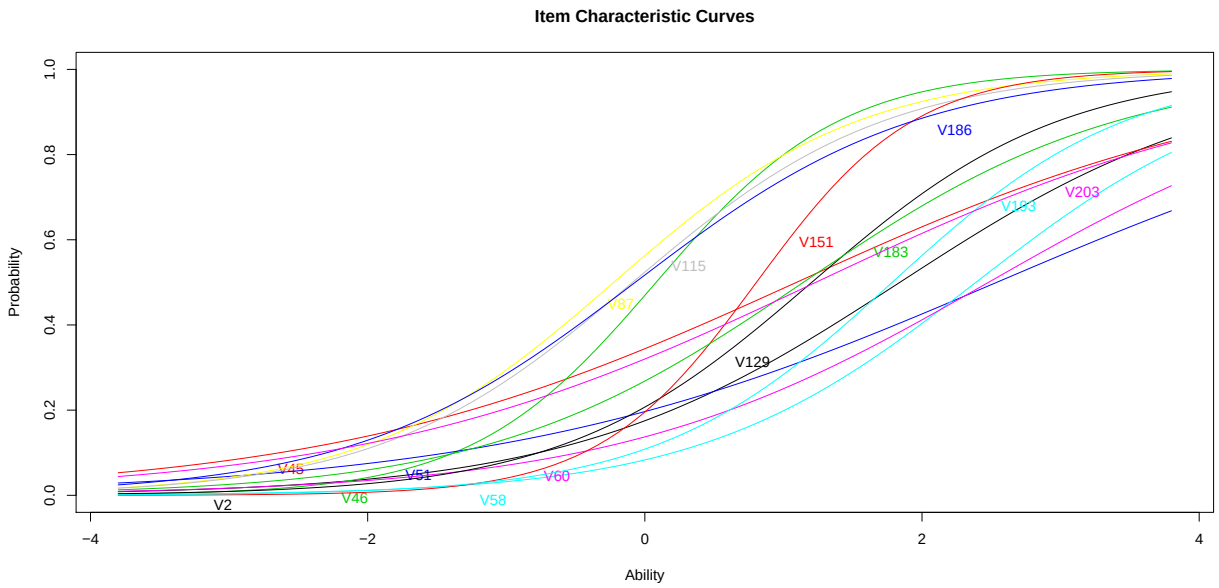
```
#Ajustamos el modelo planteando que solo existe una dimension
fit2 <- ltm(Corregidos ~ z1)

#Comparamos el modelo frente al de 1 parametro
anova(fit1,fit2)
```

```
##
##  Likelihood Ratio Table
##          AIC     BIC  log.Lik   LRT df p.value
## fit1 6904.25 6965.95 -3437.12
## fit2 6887.29 7002.47 -3415.64 42.96 13  <0.001
```

Un modelo más complejo siempre tendrá un mejor ajuste que un modelo simple, por ello, el contraste de las devianzas que hemos aplicado penaliza la complejidad del modelo. Se podría entender como una prueba que contrasta la hipótesis de si al aumentar la complejidad del modelo se produce un incremento notable en el ajuste

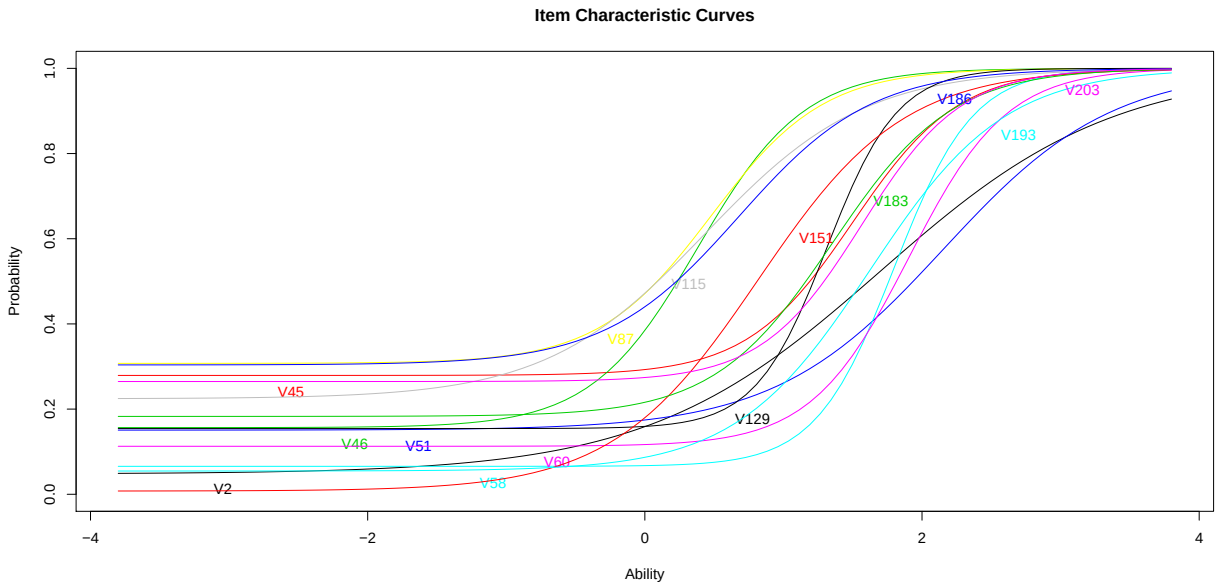
En nuestro caso, el resultado es estadísticamente significativo por lo que elegimos el modelo de 2 parámetros. Podemos ver que efectivamente los ítems tienen discriminaciones muy diferentes, por lo que el parámetro extra añade mucha información:

```
plot(fit2)
```



Modelo de tres parámetros

```
fit3 <- tpm(Corregidos)
plot(fit3)
```




```
anova(fit2, fit3)
```

```
##
##  Likelihood Ratio Table
##          AIC     BIC  log.Lik   LRT df p.value
## fit2 6887.29 7002.47 -3415.64
## fit3 6852.32 7025.10 -3384.16 62.96 14  <0.001
```

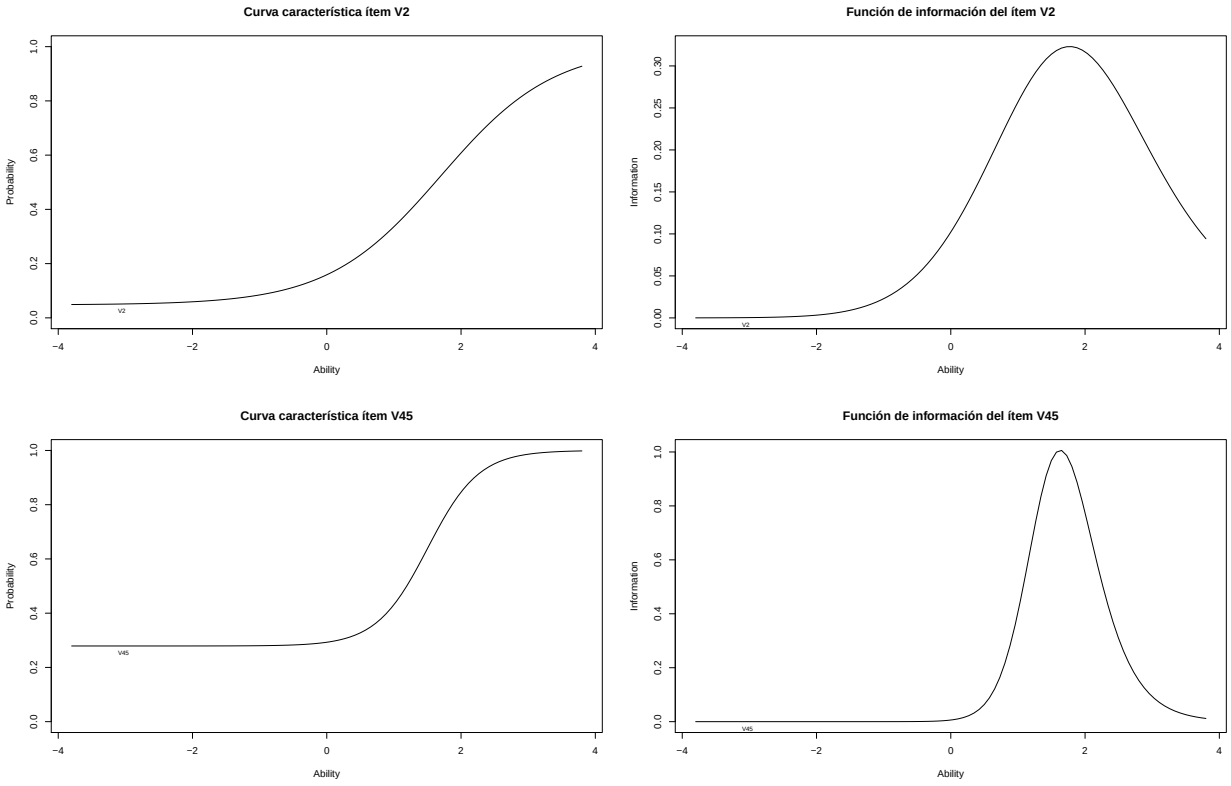
Nuevamente, el ajuste del modelo es superior. Al representar las curvas características de cada ítem vemos que los parámetros c adoptan valores muy diferentes también, por lo que al mantenerlos constantes perdemos mucha información. Por tanto, seleccionamos el modelo de 3 parámetros.

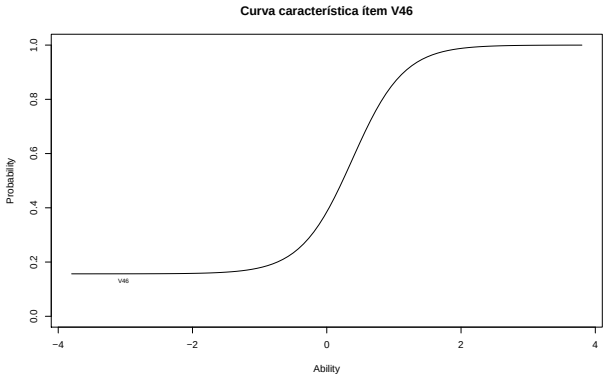
Mejor modelo

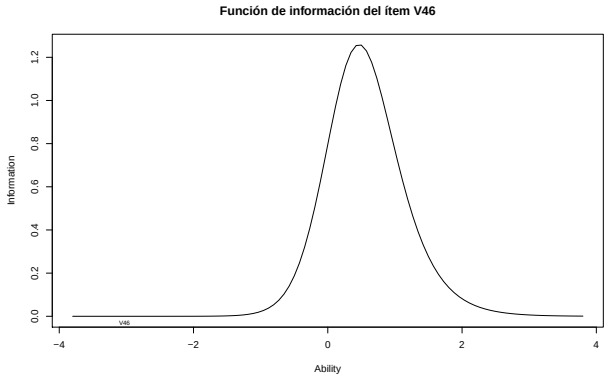
Dadas las pruebas realizadas concluimos que el modelo que mejor ajuste presenta en relación a su complejidad es el de tres parámetros. Por tanto, los ítems difieren en dificultad, discriminación y probabilidad de acierto al azar.

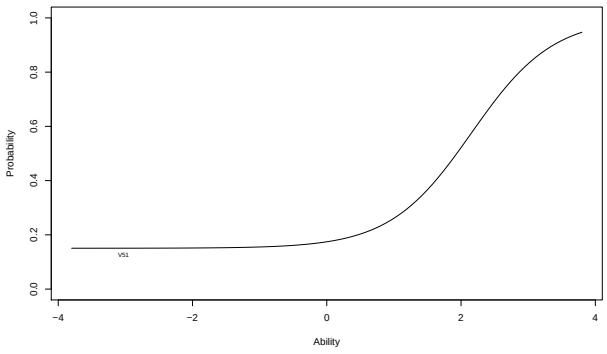
A continuación, podemos realizar un análisis más detallado de los parámetros de cada ítem para el modelo elegido.

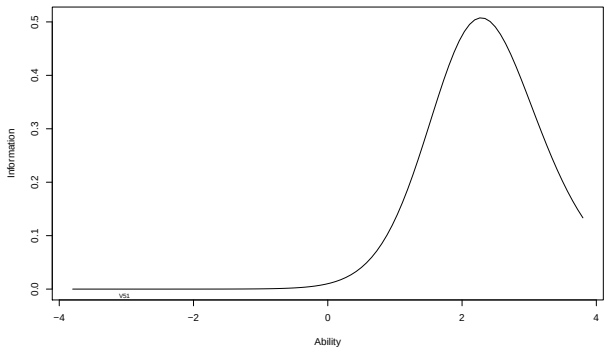
```
layout(
  matrix(c(1, 1, 1, 1, 2, 2, 2, 2),
  nrow = 2,
  byrow = FALSE)
)

for(i in 1:ncol(Datos)){
  plot(fit3, items = i,
       main = paste("Curva característica ítem",
                    colnames(Datos)[i]))
  plot(fit3,type = "IIC", items = i,
       main = paste("Función de información del ítem",
                    colnames(Datos)[i]))
}
```

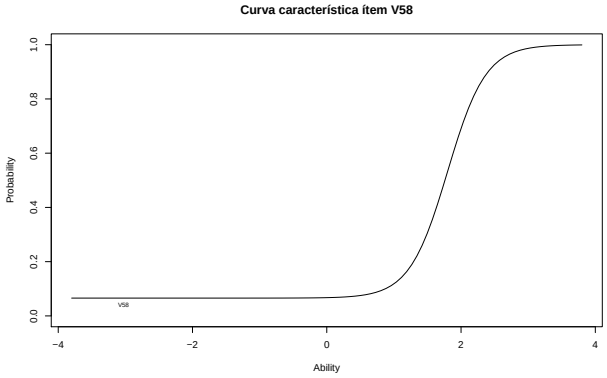


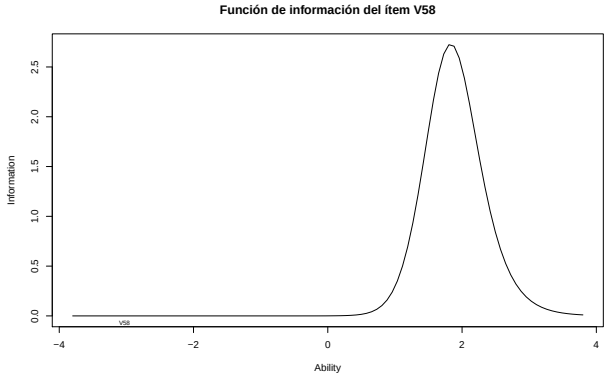

Curva característica ítem V46
Probability
Ability
V46

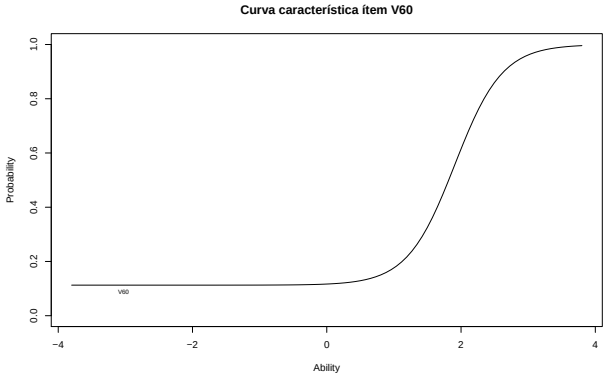

Función de información del ítem V46
Information
Ability
V46

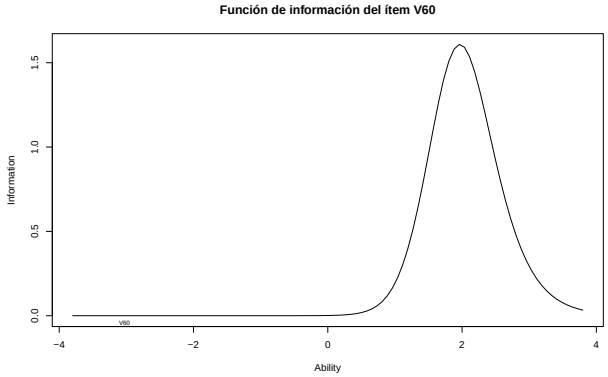

Curva característica ítem V51
Probability
Ability
V51

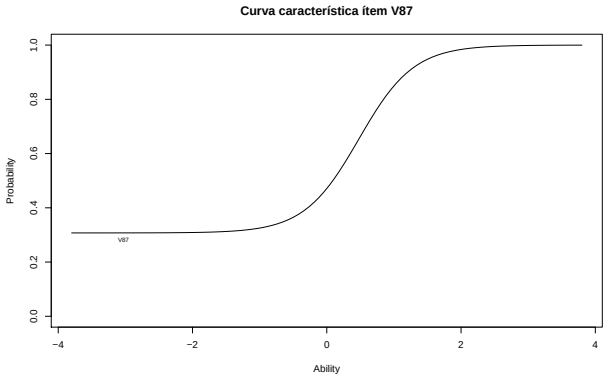

Función de información del ítem V51
Information
Ability
V51

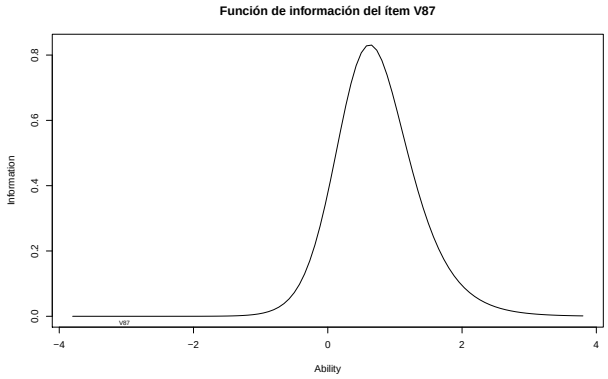

Curva característica ítem V58
Probability
Ability
V58

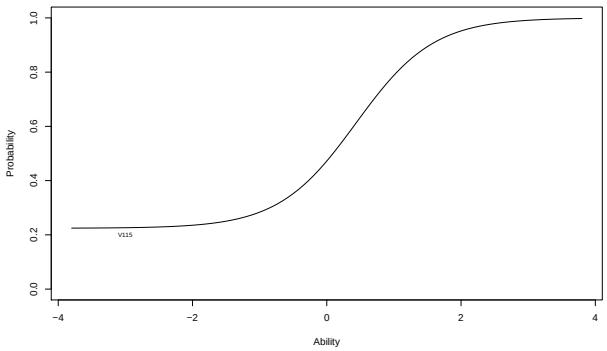

Función de información del ítem V58
Information
Ability
V58

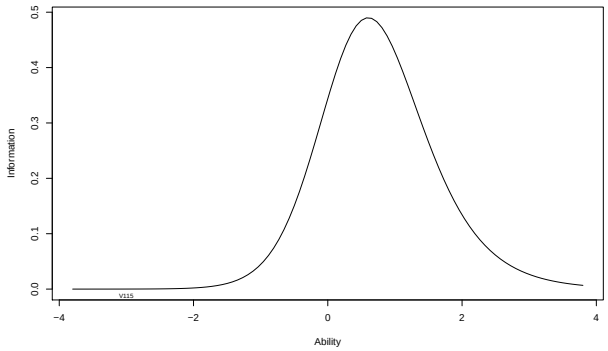

Curva característica ítem V60
Probability
Ability
V60

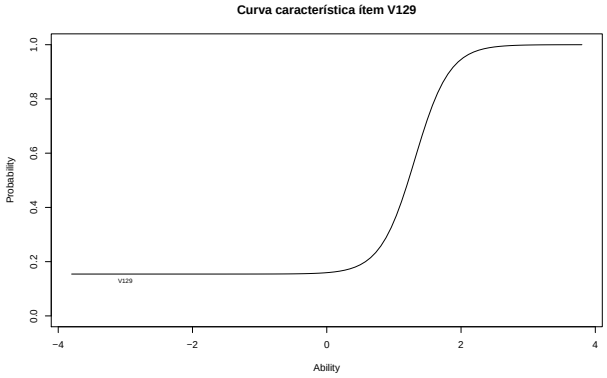

Función de información del ítem V60
Information
Ability
V60

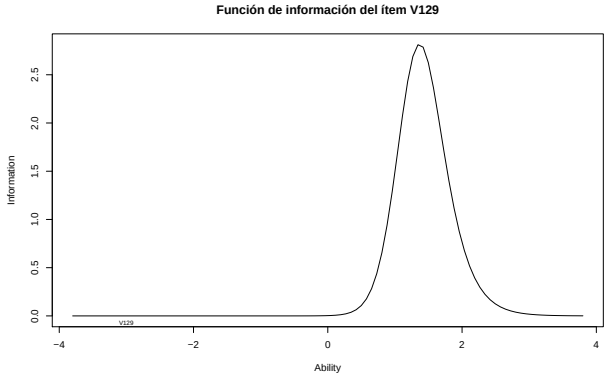

Curva característica ítem V87
Probability
Ability
V87

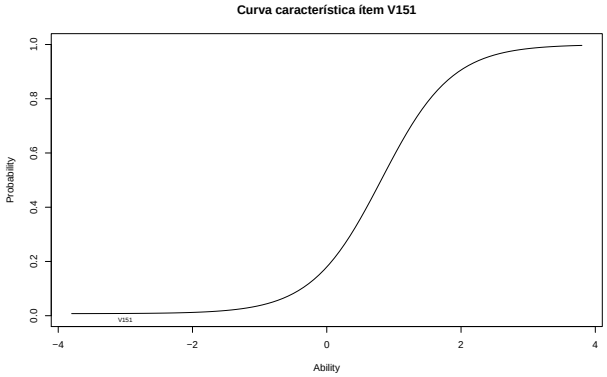

Función de información del ítem V87
Information
Ability
V87

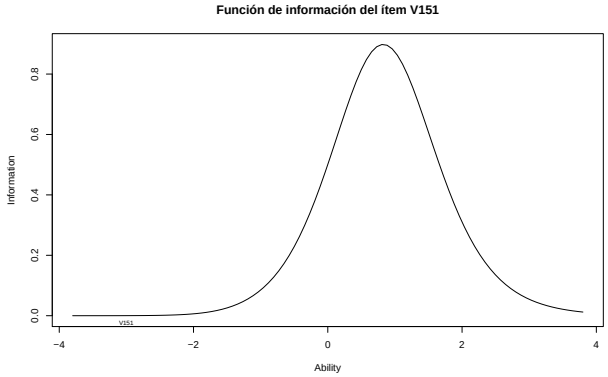

Curva característica ítem V115
Probability
Ability
V115

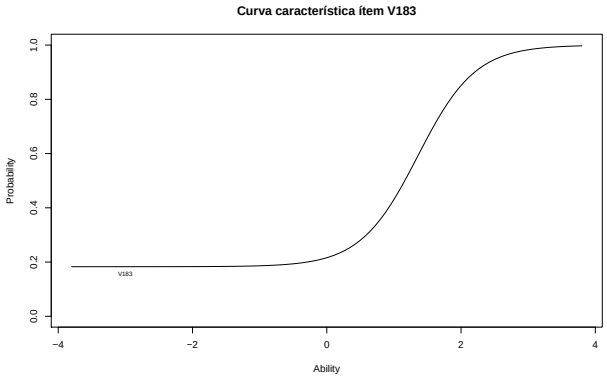

Función de información del ítem V115
Information
Ability
V115

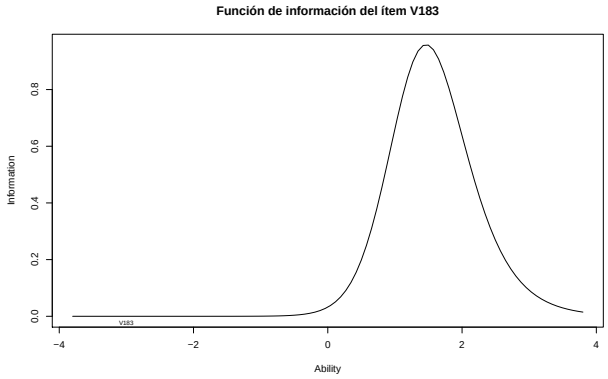

Curva característica ítem V129
Probability
Ability
V129

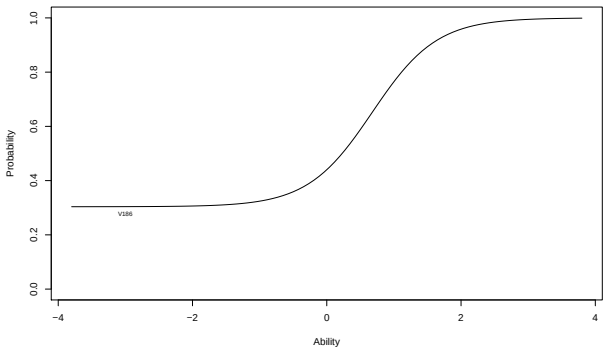

Función de información del ítem V129
Information
Ability
V129

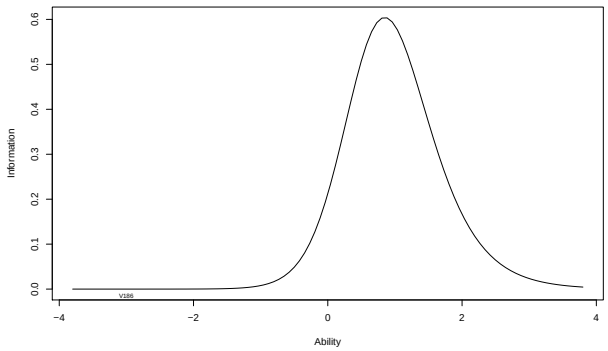

Curva característica ítem V151
Probability
Ability
V151

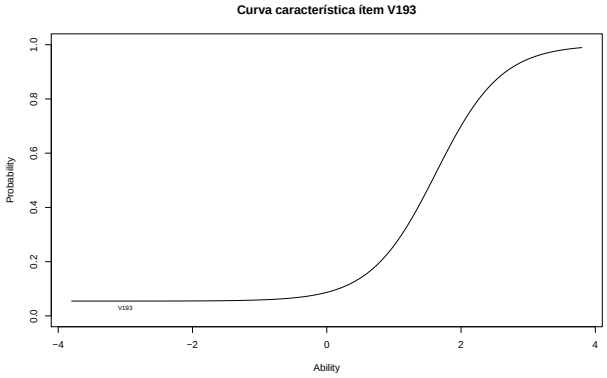

Función de información del ítem V151
Information
Ability
V151

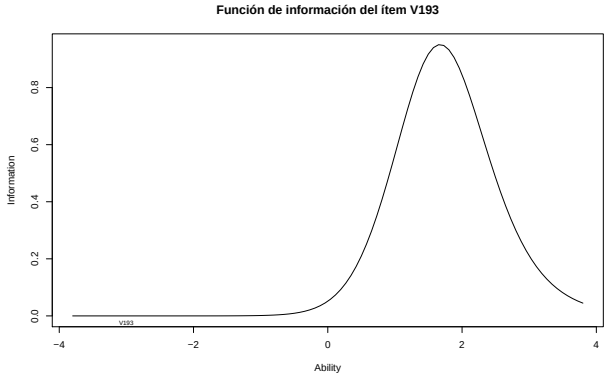

Curva característica ítem V183
Probability
Ability
V183

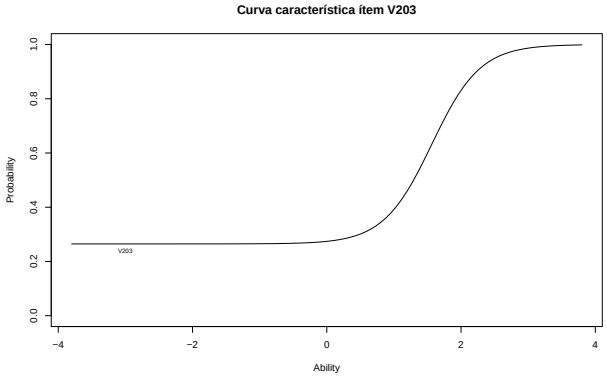

Función de información del ítem V183
Information
Ability
V183

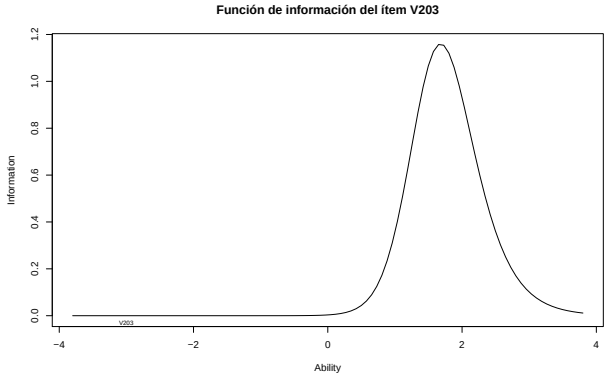

Curva característica ítem V186
Probability
Ability
V186

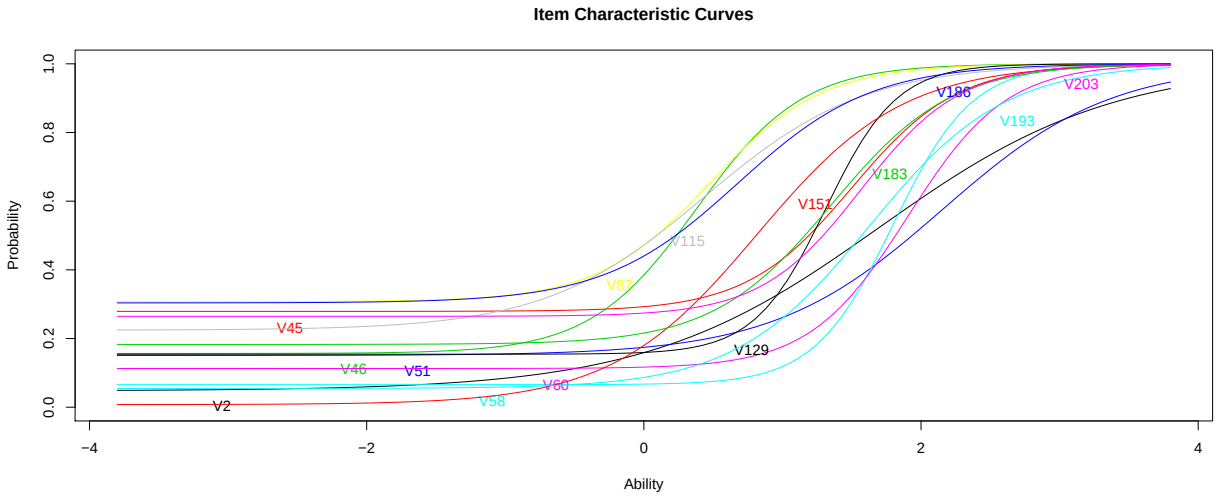

Función de información del ítem V186
Information
Ability
V186

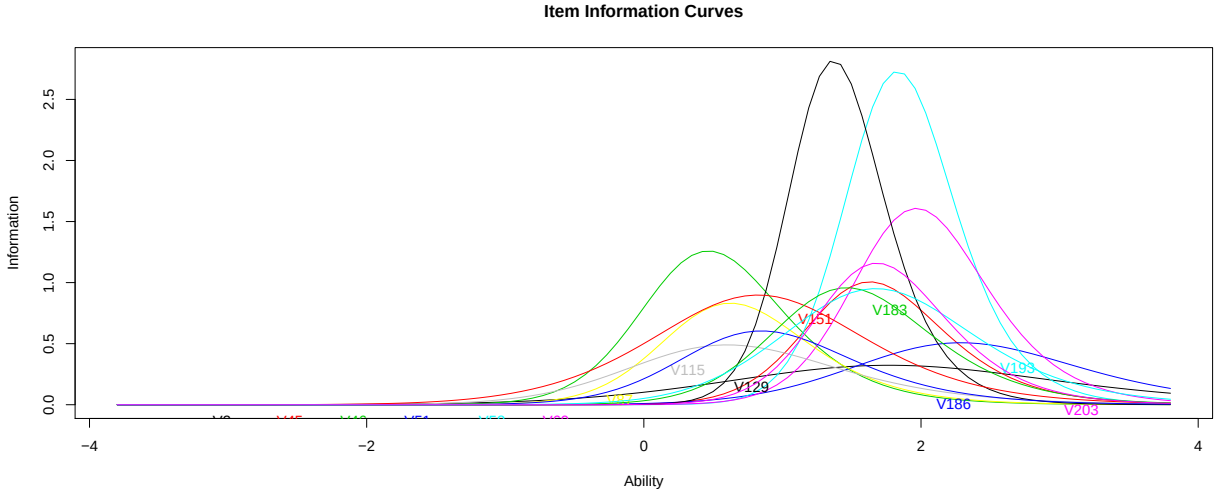

Curva característica ítem V193
Probability
Ability
V193


Función de información del ítem V193
Information
Ability
V193

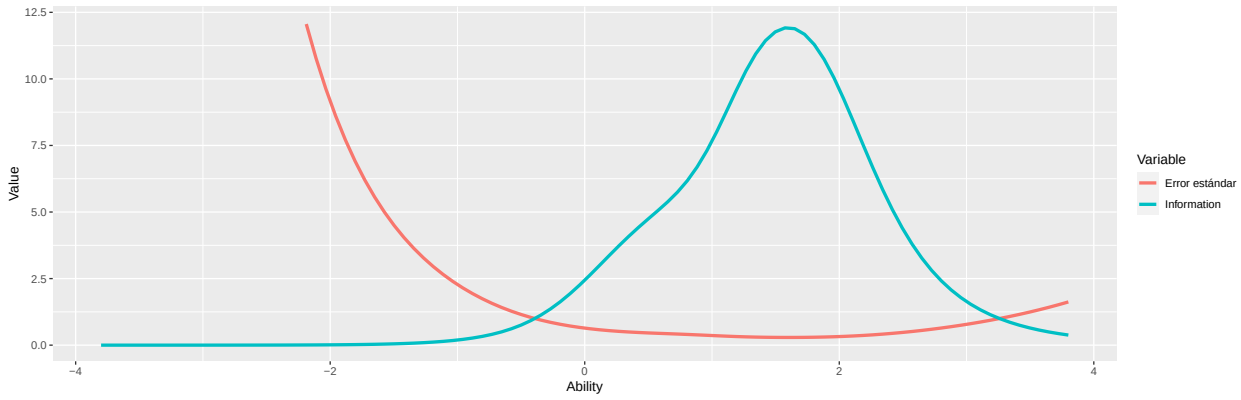

Curva característica ítem V203
Probability
Ability
V203


Función de información del ítem V203
Information
Ability
V203





Representamos la función de información del test completo. El test mide mejor a sujetos con un nivel de competencia de entorno a 1.5-2, dado que los ítems miden mejor a los sujetos con un nivel de competencia parecido a su dificultad, podemos concluir que es algo difícil.



Podemos obtener además una tabla con todos los parámetros del modelo para cada ítem.

```
Temp <- data.frame(summary(fit3)$coefficients)
Temp$coeff <- rownames(Temp)
Temp$coeffCate <- sub("\\..*", "", rownames(Temp))
Temp2 <- as.list(unique(Temp$coeffCate))
Temp2 <- sapply(Temp2, function(x){
  Temp$value[Temp$coeffCate == x]
})
Parametros <- data.frame(Temp2)
rownames(Parametros) <- colnames(Datos)
colnames(Parametros) <- unique(Temp$coeffCate)

kable(round(Parametros, 4))
```

	Gussng	Dffclt	Dscrmn
V2	0.0478	1.7002	1.1914
V45	0.2791	1.5022	2.6247
V46	0.1565	0.3802	2.6085
V51	0.1506	2.1540	1.6470
V58	0.0656	1.8000	3.5235
V60	0.1127	1.9060	2.8280
V87	0.3074	0.4782	2.4546
V115	0.2243	0.4356	1.7360
V129	0.1543	1.3115	3.8948
V151	0.0076	0.8179	1.9098
V183	0.1828	1.3568	2.3334
V186	0.3037	0.6775	2.0845
V193	0.0546	1.6268	2.0573
V203	0.2647	1.5646	2.7780

Ajuste absoluto del modelo

En los apartados anteriores comparabamos el ajuste del modelo de tres parámetros con los de uno y dos para ver si el incremento en la complejidad del modelo produce un incremento suficiente del ajuste, es decir evaluamos su ajuste en relación con los otros modelos.

También podemos evaluar el ajuste absoluto del modelo.

```
ajuste <- item.fit(fit3,
                   simulate.p.value = TRUE,
                   B = 20)

Temp <- round(unname(ajuste$p.values), 3)
Resumen <- cbind(
  data.frame(
    Item = colnames(Datos)[1:((ncol(Datos) / 2))],
    Desajuste = Temp[1:((ncol(Datos) / 2))]
    ),
  data.frame(
    Item = colnames(Datos)[((ncol(Datos) + 1) / 2):ncol(Datos)],
    Desajuste = Temp[((ncol(Datos) + 1) / 2):ncol(Datos)]
  )
)

kable(Resumen)
```

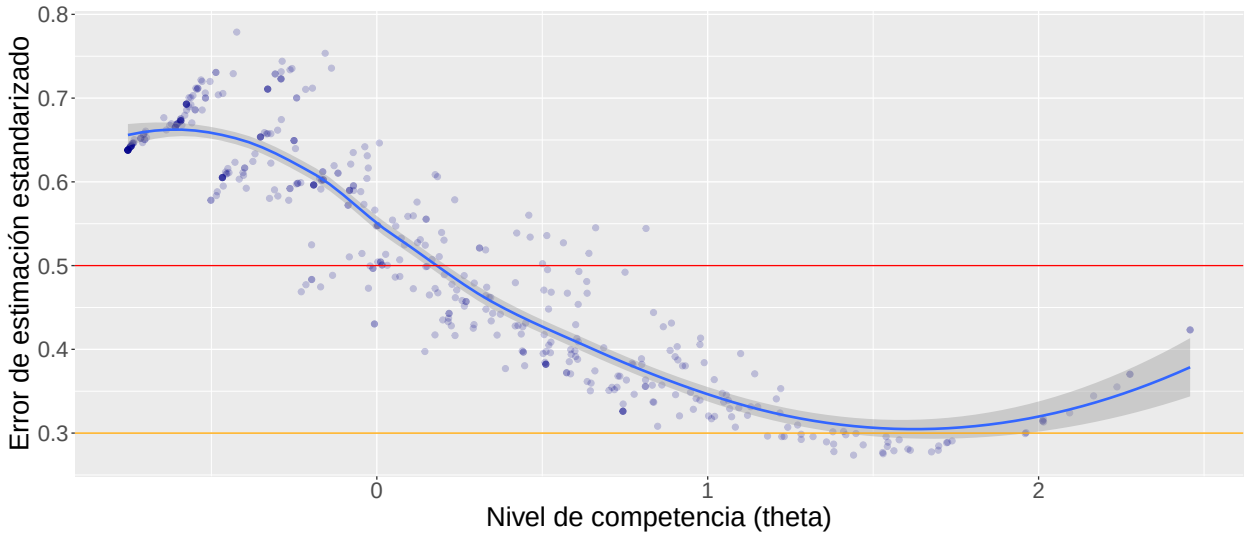
Item	Desajuste	Item	Desajuste
V2	0.286	V87	0.524
V45	1.000	V115	0.524
V46	0.810	V129	0.524
V51	0.571	V151	0.619
V58	0.524	V183	0.476
V60	0.810	V186	0.667
V87	0.524	V193	0.476

En la tabla vemos los p-valores asociados al contraste que determina si el desajuste del ítem es significativo. En todos los casos observamos p-valores altos, por lo que mantenemos la hipótesis nula (**el desajuste no es significativo**).

Estimación de puntuaciones

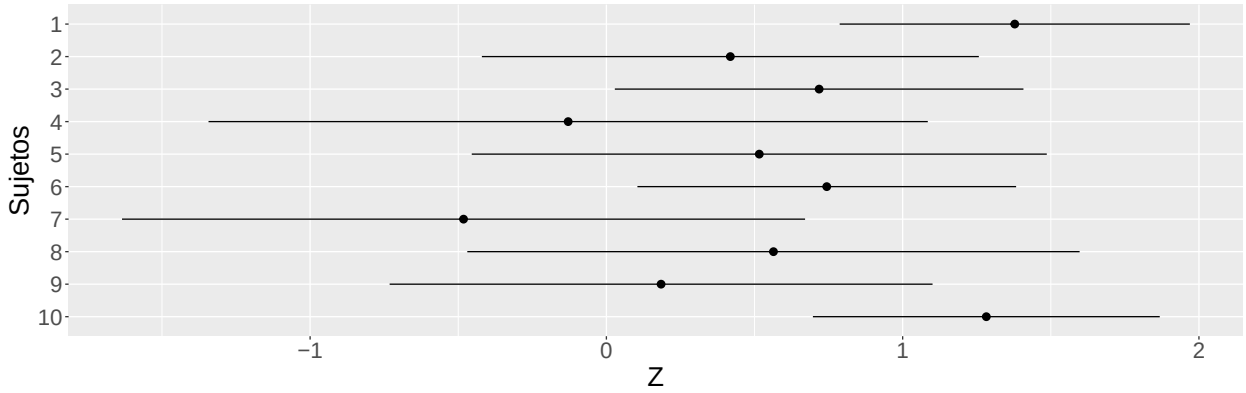
A continuación, estimamos las puntuaciones para los sujetos en el rasgo (θ). Las representamos en un gráfico relacionandolas con su error de estimación

```
Puntuaciones <- ltm::factor.scores(fit3,
                                   Corregidos)
Zest <- matrix(0, nrow(Datos), 2)
Zest[, 1] <- Puntuaciones$score.dat$z1
Zest[, 2] <- Puntuaciones$score.dat$se.z1
Zest <- data.frame(Zest)
colnames(Zest) <- c("Z","SE")

#Representamos los errores de estimacion
ggplot(Zest, aes(x = Z, y = SE)) +
  geom_point(alpha = 0.2,
             size = 2,
             color = "darkblue") +
  geom_smooth() +
  xlab("Nivel de competencia (theta)") +
  ylab("Error de estimación estandarizado") +
  geom_hline(yintercept = 0.3, color = "orange") +
  geom_hline(yintercept = 0.5, color = "red") +
  theme(text = element_text(size = PlotFont))
```



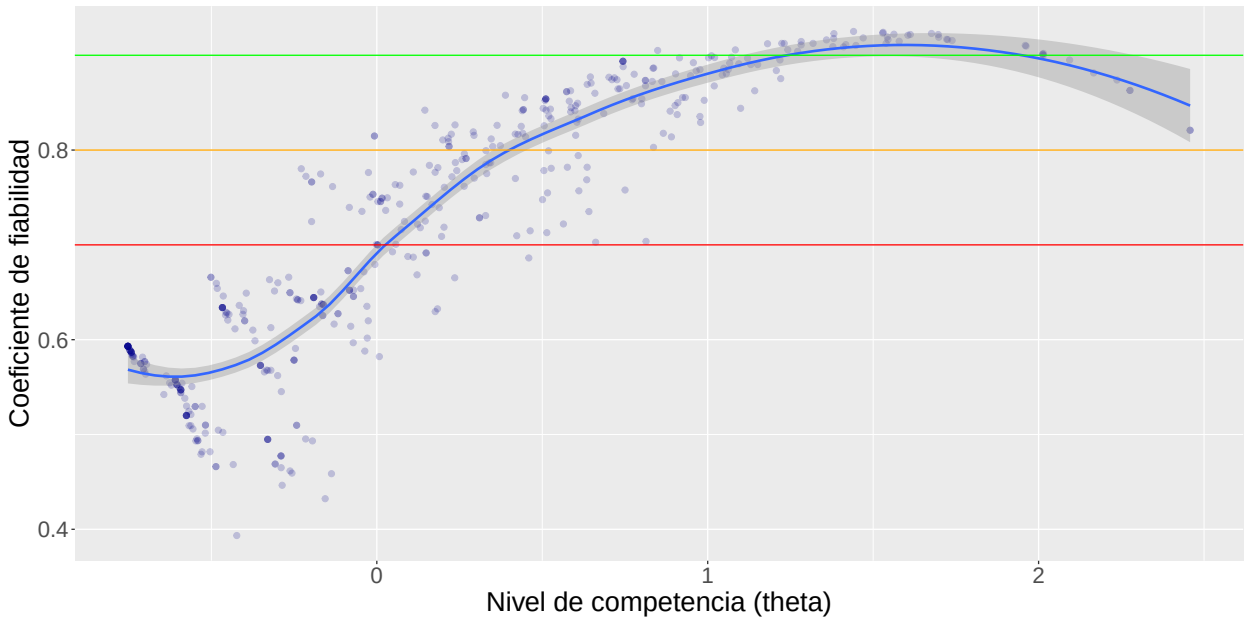
Observamos que en los niveles bajos de competencia cometemos mucho error (debido a que la dificultad del test es más elevada). La línea horizontal naranja representaría un nivel de precisión aceptable. Valores de error por encima de la misma empienzan a ser altos. Cualquier valor que esté por encima de la línea roja implica que esa estimación es bastante imprecisa.

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") +
  geom_hline(yintercept = 0.7, color = "red") +
  theme(text = element_text(size = PlotFont))
```



Comentarios sobre el modelo

En base a éstos datos, son pertinentes las siguientes observaciones:

- El 29 % de los ítems tienen parámetros de adivinación superiores a lo que cabría esperar por azar.
  - Dado el número de alternativas, la probabilidad de acierto por azar es del 20 %, pero éstos ítems tienen unos valores superiores al 25 %, que sería lo esperable para ítems con 4 alternativas de respuesta.
  - Es posible que en éstos casos los distractores no estén funcionando correctamente.
- Todos los índices de dificultad son mayores que cero. Durante el proceso, quitamos algunos ítems por ser demasiado fáciles, ya que no son adecuados para ésta muestra en particular. Sin embargo, viendo la función de información del test quizás no sería conveniente utilizar éste criterio para eliminar los ítems.
  - El test resultante mide muy bien a los sujetos con un nivel de competencia medio-alto, pero la función de información presenta un grado de apuntamiento muy elevado. La cola izquierda en particular incluye niveles de competencia bastante frecuentes, estando el nivel medio ($\theta = 0$) muy próximo a la misma con unos niveles de información relativamente bajos. Por tanto, a medida que desciende el nivel de competencia por debajo de cero, las mediciones van siendo cada vez más imprecisas hasta llegar a unos niveles de error muy altos.
- En vista a éstos resultados queda patente que el criterio para incluir o no un ítem quizás no deba ser simplemente si es demasiado fácil o difícil, ya que conviene que todos los niveles de dificultad estén bien representados en un banco de ítems, así obtendremos un test que ofrece buenas mediciones en rangos más amplios de competencia.
  - Por tanto sería coveniente añadir ítems con dificultades diferentes, especialmente más bajas.
  - Lógicamente, la dificultad de los ítems debe ir siempre en función de la población sobre la que se aplicará. En nuestro caso, sería interesante añadir ítems con dificultades entre -1.5 y 0, y algunos más con dificultades cercanas a 3 para discriminar mejor entre los sujetos más competentes.
- En relación con los índices de discriminación, se observa que la mayoría son adecuados.
  - Lo ideal es contar con ítems muy discriminativos y con niveles de dificultad variados, de ésta forma, podremos mantener un buen nivel de precisión para sujetos con distintos niveles de competencia.
- Nuestro test posee una precisión excelente para sujetos con niveles de competencia en torno a 1.5, y aceptable entre valores de 0.5 y 2.5. No obstante, por debajo de 0 presenta unos niveles de fiabilidad excesivamente bajos.
  - Queda claro que hacen falta bastantes ítems con dificultades que se encuentren por debajo de 0.5 y algunos por encima de 2.5 para corregir los problemas de precisión en éstos niveles de competencia.

Tarea voluntaria

Comparación TCT vs TRI

```
#Obtenemos los indicadores de la TRI
ModelosTRI <- list(fit1, fit2, fit3)

obtenerCoeficientes <- function(model){
  Temp <- data.frame(summary(model)$coefficients)
  Temp$coeff <- rownames(Temp)
  Temp$coeffCate <- sub("\\..*", "", rownames(Temp))
  Temp2 <- as.list(unique(Temp$coeffCate))
  Temp2 <- sapply(Temp2, function(x){
    Temp$value[Temp$coeffCate == x]
  })
  return(Temp)
}

Temp <- lapply(ModelosTRI, obtenerCoeficientes)
for(i in 1:length(Temp)){
  Temp[[i]]$Model = i
}

#Parametros de los tres modelos de TRI
TRI <- (do.call(rbind, Temp))
rownames(TRI) <- NULL

#A continuación obtenemos los parametros de la TCT
TCT <- itemAnalysis(Corregidos)
TCT <- TCT$itemReport

#Los juntamos en un mismo data frame
n <- nrow(Items) - 1
Temp <- data.frame(
  DiffTRI = TRI$value[TRI$coeffCate == "Dffclt"],
  DiscrTRI = c(rep(TRI$value[TRI$coeffCate == "Dscrmn"][1], n),
               TRI$value[TRI$coeffCate == "Dscrmn"]),
  Model = TRI$Model[TRI$coeffCate == "Dffclt"]
)
Temp$DiffTCT <- rep(TCT$itemMean, nrow(Temp) / nrow(TCT))
Temp$DiscrTCT <- rep(TCT$pBis, nrow(Temp) / nrow(TCT))
TCTvsTRI <- Temp
```

En la TCT el índice de dificultad de define como la proporción de sujetos que aciertan el item [1]. En TRI, hace referencia al nivel de θ que hace que la probabilidad de acertar el ítem sea del 50 %.

El indicador de la TCT se hace más pequeño cuanto más difícil es el ítem (menos sujetos lo aciertan), mientras que el de TRI se hace más grande (es necesario un nivel de competencia mayor para tener una probabilidad de acierto del 50 %). Por tanto, cabe esperar una correlación próxima a 1 y con signo negativo.

En la siguiente tabla podemos ver las correlaciones entre los índices de dificultad y discriminación de los tres modelos de TRI y los mismos valores de la TRI.

Model	CorrDiff	CorrDiscr
1 parámetros	-0.9937782	NA
2 parámetros	-0.9419610	0.9691673
3 parámetros	-0.8673085	0.0072091

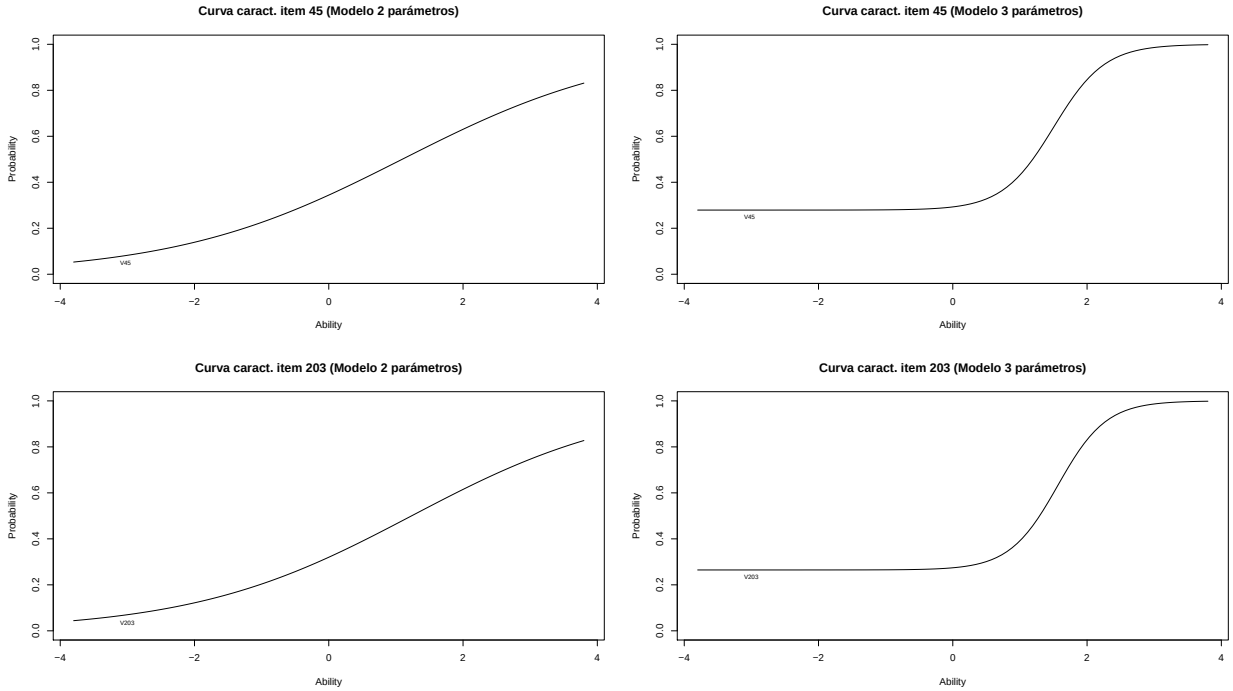
En cada fila tenemos las correlaciones entre uno de los modelos de la TRI y la TCT. La primera columna muestra las correlaciones entre los índices de dificultad y la segunda entre los índices de discriminación. Podemos ver que en la segunda columna no hay valor para el modelo de 1 parámetro ya que el valor del parámetro es el mismo para todos los ítems.

Para el parámetro de dificultad encontramos el resultado esperado: correlaciones altas y con signo negativo. Vemos que se van haciendo algo más pequeñas (en valor absoluto) conforme aumenta el número de parámetros, presumiblemente debido a los cambios en las curvas características de los ítems.

Dado que comprobamos que el ajuste de los datos mejora al añadir parámetros, ésto implica que la TCT no llega a ajustar demasiado bien (los valores más precisos serían los del modelo de 3 parámetros). Aun con todo, la TCT ofrece unas aproximaciones bastante correctas.

[1] Por tanto, quizás sea más adecuado llamarlo índice de facilidad

El caso de los índices de discriminación es más curioso, ya que la correlación es muy alta para el modelo de 1 parámetro y casi nula para el de 3. Examinando las curvas características de los ítems para los dos modelos nos damos cuenta de que el tener en cuenta el parámetro de adininación cambia de forma considerable muchas de las curvas. Podemos poner como ejemplos los ítems 45 y 203.



Sin tener en cuenta la probabilidad de adivinación los ítems tienen una discriminación muy baja. Una vez que lo tenemos en cuenta la discriminación aumenta considerablemente. Como los ítems tienen una probabilidad de acierto por azar muy alta no son capaces de discriminar muy bien entre sujetos competentes e incompetentes (los incompetentes los aciertan por azar).

Al controlar la influencia del azar, nos damos cuenta sí tienen una buena discriminación, pero tienen una probabilidad base de acierto alta, incluso para sujetos con bajo nivel de competencia.

Por tanto, podemos asumir que los valores correctos de discriminación son los del modelo de tres parámetros de la TRI. Para nuestro test, la TCT no ofrece unas buenas estimaciones de la discriminación. Ésto se debe al hecho que comentamos anteriormente. Para un tercio de nuestros ítems los distractores no funcionan muy bien, por lo que la probabiliad de acierto por azar es muy alta.

Si quisieramos utilizar la TCT, deberíamos prestar especial atención a éste hecho mediante el análisis de distractores, ya que el indicador que proporciona no tiene en cuenta la probabilidad de acierto por azar. No obsante, se podría corregir para tenerlo en cuenta.

Invarianza de medida

Podemos comprobar si los valores obtenidos para los parámetros de los ítems se mantienen independientemente de los sujetos. Para ello, podemos dividir la muestra en dos mitades y ver si estimamos los mismos parámetros.

Construimos una simulación que crea las muestras, ajusta los modelos y devuelve las correlaciones entre los parámetros. Para aumentar el rendimiento podemos usar la libería parallel, que nos permite utilizar todos los núcleos del procesador para realizar los cálculos en paralelo.

```
n <- 100 #Veces que ejecutamos la funcion
Semillas <- sample(1:1000, n, replace = FALSE)

AjustaModelos <- function(x){
  set.seed(x) #La generacion de numeros al azar el paralelo
  #da problemas, definimos la semilla de antemano
  #Aplicamos un shuffle sobre los datos
  Temp <- sample(1:nrow(Corregidos), replace = FALSE)
  Temp2 <- Corregidos[Temp, ]

  #Creamos las dos muestras y las metemos en una lista
  DosMuestras <- list(
    g1 = Temp2[1:ceiling(nrow(Temp2) / 2), ],
    g2 = Temp2[(ceiling(nrow(Temp2) / 2) - 1): nrow(Temp2), ]
  )

  Coef <- lapply(DosMuestras, function(df){
    fit <- ltm(df ~ z1)
    Temp <- data.frame(summary(fit)$coefficients)
    Temp$coeff <- rownames(Temp)
    Temp$coeffCate <- sub("\\..*", "", rownames(Temp))
    Temp2 <- as.list(unique(Temp$coeffCate))
    Temp2 <- sapply(Temp2, function(x){
    Temp$value[Temp$coeffCate == x]
    })
    Parametros <- data.frame(Temp2)
    rownames(Parametros) <- colnames(Datos)
    colnames(Parametros) <- unique(Temp$coeffCate)
    return(Parametros)
  })
  Coef <- do.call(cbind, Coef)
  return(c(cor(Coef$g1.Dffclt, Coef$g2.Dffclt),
           cor(Coef$g1.Dscrmn, Coef$g2.Dscrmn)))
}
```

```r
library(parallel)
Ncores <- round(detectCores()/2, 0)
if(Ncores < 1) Ncores = 1

#Creamos un cluster de procesamiento
clust <- makeCluster(Ncores, type = "FORK")

#Aplicamos la funcion en paralelo
Resultados <- t(parSapply(clust, Semillas, AjustaModelos))
stopCluster(clust) #Detenemos el cluster
colnames(Resultados) <- c("CorrDiff", "CorrDiscrim")
```

En las siguientes tablas podemos ver los cuantiles para las correlaciones obtenidas en la simulación. En primer lugar para las correlaciones entre los parámetros de dificultad. La correlación media obtenida en las 100 iteraciones fue de **0.845**

```
round(quantile(Resultados[, "CorrDiff"]), 3)
```

```
##    0%   25%   50%   75%  100%
## 0.201 0.827 0.862 0.909 0.970
```

Ahora para los parámetros de discriminación. La correlación media obtenida fue **0.568**

```
round(quantile(Resultados[, "CorrDiscrim"]), 3)
```

```
##    0%   25%   50%   75%  100%
## 0.276 0.471 0.580 0.659 0.858
```

Encontramos que sí que se puede asumir que la dificultad permanece constante entre muestras (el 75 % de las correlaciones están por encima de 0.827), no se puede decir lo mismo de los parámetros de discriminación (el 75 % de las correlaciones están por debajo de (0.659).

Por tanto la invarianza de medida aplica más para el parámetro de dificultad que para la discriminación, pero en ningún caso llega a ser perfecta.