Teoría Clásica de los Tests

Manuel Torres Acosta

Índice

Tarea

La tarea de ésta semana consiste en adaptar el código que nos han proporcionado para analizar los ítems de una prueba de inglés. Tenemos que seleccionar al azar un conjunto de 20 ítems para obtener y comentar las propiedades vistas en clase.

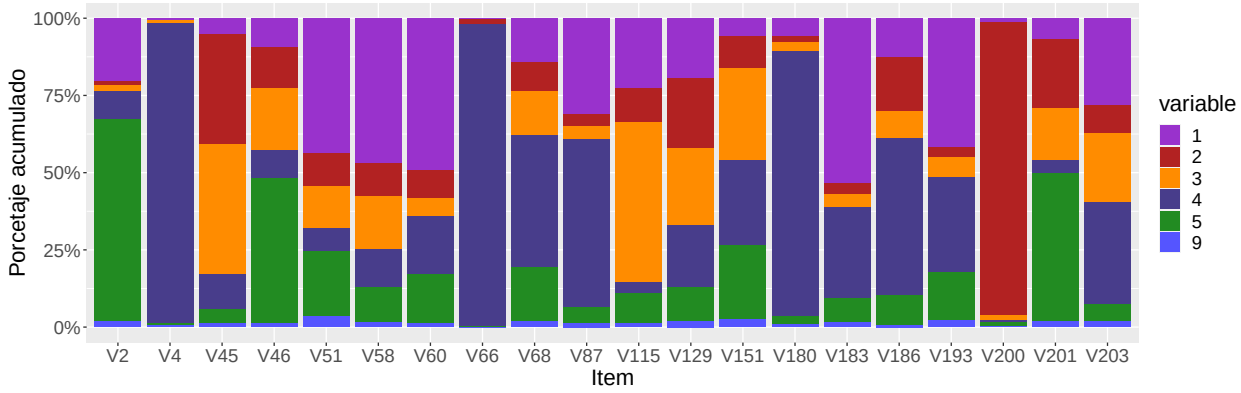
Selección de los ítems y visualización

Para seleccionar los ítems se utilizo la función sample que genera 20 números aleatorios sin reposición.

Se obtuvieron los siguientes, que se introducirían manualmente en el código:

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
#(guarde la tabla del word como csv)
Items <- read.csv(Rutas$Items,
                  stringsAsFactors = FALSE)[idxItems, ]
Respuestas <- Items$Respuesta


#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") +
  scale_y_continuous(labels = percent) +
  ylab("Porcetaje acumulado") +
  scale_fill_manual(values = Palette) +
  theme(text = element_text(size = PlotFont / 1.2))
```



Vemos que los ítems 4, 66, 180 y 200 son demasiado fáciles, ya que la mayoría de la muestra ha seleccionado los valores correctos. Es posible que sean palabras demasiado fáciles o que los distractores no estén funcionando bien.

Nota: En el gráfico, el 9 hace referencia a los perdidos.

Calificación de los ítems

Conociendo las puntuaciones correctas para cada item, podemos utilizar la función `score()` para obtener una puntuación dicotómica que nos indica si el sujeto acertó el ítem o no.

```
idxMantener <- !colnames(Datos) %in%
                 c("V4", "V66", "V180", "V200")
#Quitamos los items fáciles
Datos <- Datos[, idxMantener]
Respuestas <- Respuestas[idxMantener]
Items <- Items[idxMantener, ]
Corregidos <- score(Datos, Respuestas,
                    output.scored = TRUE)
Temp <- as.data.frame(Corregidos$scored)
Puntuaciones <- unname(Corregidos$score)
rownames(Temp) <- names(Corregidos$score)
Corregidos <- Temp
```

`Corregidos$scored` nos permitiría ver la matriz que indica si se ha acertado o no cada ítem, mientras que `Corregidos$score` nos muestra las puntuaciones ya calculadas para cada sujeto.

Análisis de los distractores

Podemos utilizar las puntuaciones para crear tres grupos en función del rendimiento.

```
Corte <- quantile(Puntuaciones,
         c(0.33, 0.66))
Grupos <- rep("", nrow(Datos))
Grupos[Puntuaciones < Corte[1]] = "Bajo"
Grupos[Puntuaciones >= Corte[1] &
         Puntuaciones < Corte[2]] = "Medio"
Grupos[Puntuaciones >= Corte[2]] = "Alto"
GruposFactor <- factor(Grupos,
                       levels = c("Bajo", "Medio", "Alto"))
```

A continuación, podemos utilizar éstos grupos para ver el comportamiento de los distractores y de la respuesta correcta.

He decidido crear una función alternativa a `plot.distractors()`, ya que en mi ordenador las leyendas no se muestran correctamente.

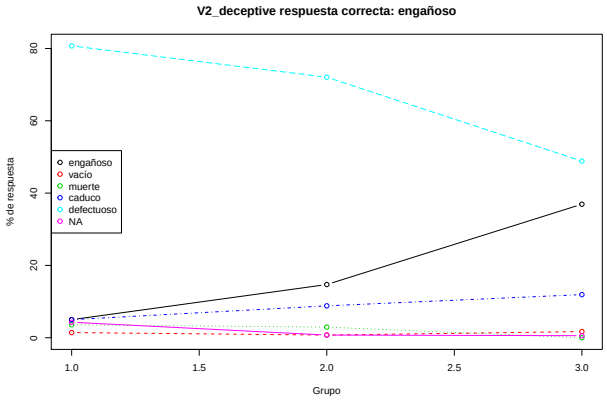
Además, ésta versión permite representar todos los items de un conjunto de datos sin necesidad de escribir los nombres, también acepta un vector con los items a representar. Muestra la palabra a traducir, las opciones y la respuesta correcta en cada gráfico de modo que la interpretación de los patrones de respuesta es más fácil.

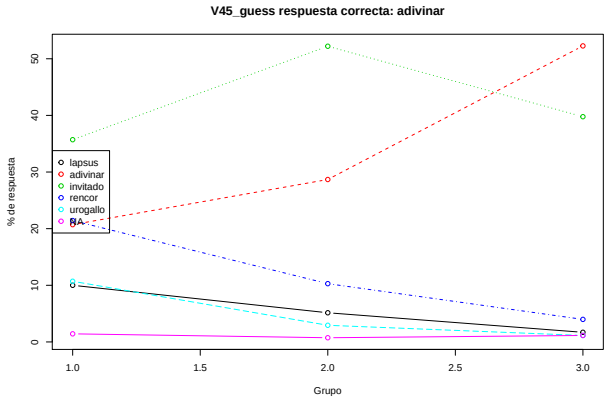
En caso de que se le pida que represente más de un item, representa cada uno en un subgráfico para mayor comodidad en la visualización de los datos.

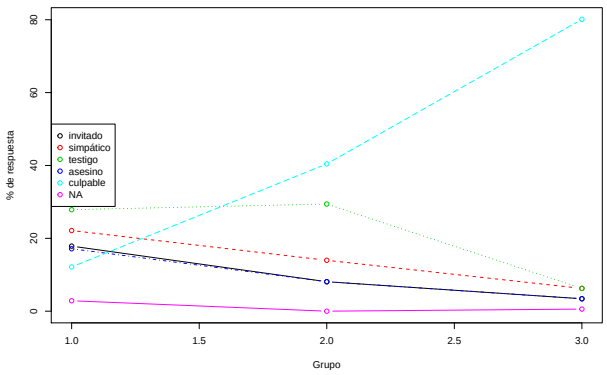
```
plot.distractors <- function(data, groups,
                             items = "all",
                             correct = "none",
                             itemChoices = "none"){
  #data: conjunto de datos, debe contener solo los items
  #items: nombres de columna de items a representar como string
  #si no se espefica ninguno representa todos los items
  #groups: vector con grupos de rendimiento, conviene que sea
  #correct: vector con las respuestas
  #un factor ordenado para representar en orden los grupos
  #itemChoices: dataframe con opciones de items en cada columna

  #Convertimos las variables en factores
  #De lo contrario table() no aporta los valores correctamente
  DatosFactor <- lapply(X = data,
                        FUN = factor,
                        levels = as.character(c(1:5, 9)))
  DatosFactor <- as.data.frame(DatosFactor)
```

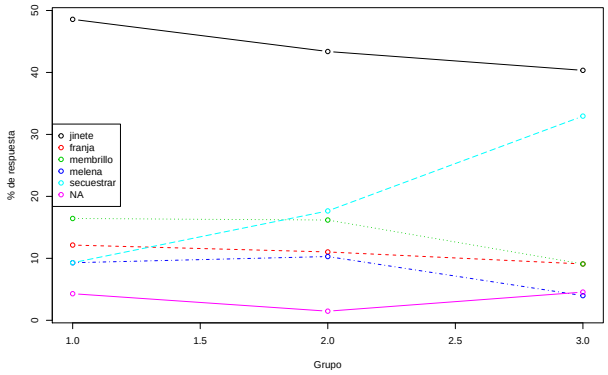
```
if(items[1] != "all"){
  DatosFactor <- data.frame(DatosFactor[, items])
  colnames(DatosFactor) <- items
} else {
  items <- colnames(DatosFactor)
}

if(length(items) > 1){
  layout(
    matrix(c(1, 1, 2, 2, 3, 3, 4, 4),
           nrow = 2,
           byrow = TRUE))
}

TablaPorc <- function(x){
  Total <- sum(x)
  round((x / Total) * 100, 2)
}

for(i in 1:ncol(DatosFactor)){
  DatosGrafico <- tapply(DatosFactor[, i],
                         groups,
                         table)

  DatosGrafico <- lapply(DatosGrafico, TablaPorc)
  DatosGrafico <- as.data.frame(do.call(rbind, DatosGrafico))

  if(correct[1] == "none"){
    Titulo <- colnames(DatosFactor)[i]
  } else {
    if(itemChoices[1] != "none"){
      Titulo <- paste(colnames(DatosFactor)[i],
                      "respuesta correcta:",
                      itemChoices[i, correct[i]])
    } else {
      Titulo <- paste(colnames(DatosFactor)[i],
                      "respuesta correcta:", correct[i])
    }

  }


  matplot(DatosGrafico, type = "b",
          main = Titulo,
```

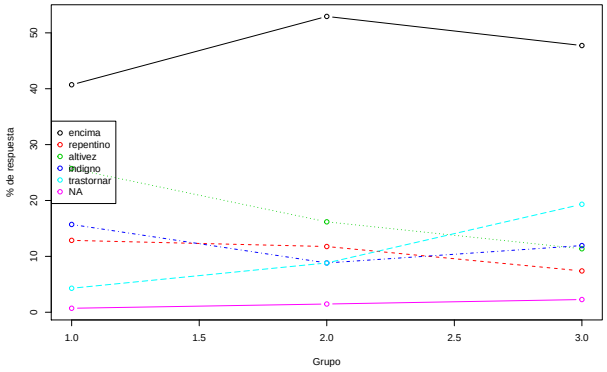
```
            xlab = "Grupo",
            ylab = "% de respuesta",
            col = 1:ncol(DatosGrafico),
            pch = 1)

      if(itemChoices[1] == "none"){
        legend("left", legend = colnames(DatosGrafico),
             col = 1:ncol(DatosGrafico),
             pch = 1)
      } else {
        legend("left", legend = c(itemChoices[i, ], "NA"),
             col = 1:ncol(DatosGrafico),
             pch = 1)
      }

  }

  if(length(items) > 1){
    par(mfrow = c(1, 1)) #Resetea la disposicion de los graficos
  }
}

#Añadimos las preguntas a los nombres de los items para verlas
#en los graficos
Temp <- Datos
colnames(Temp) <- paste(colnames(Temp), "_", Items$Pregunta,
                        sep = "")
plot.distractors(Temp, GruposFactor,
                 correct = Respuestas,
                 itemChoices = Items[4:ncol(Items)])
```

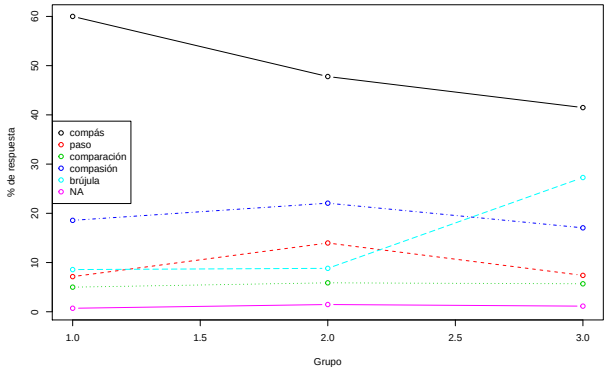

V2_deceptive respuesta correcta: engañoso
% de respuesta
Grupo
engañoso
vacío
muerte
caduco
defectuoso
NA

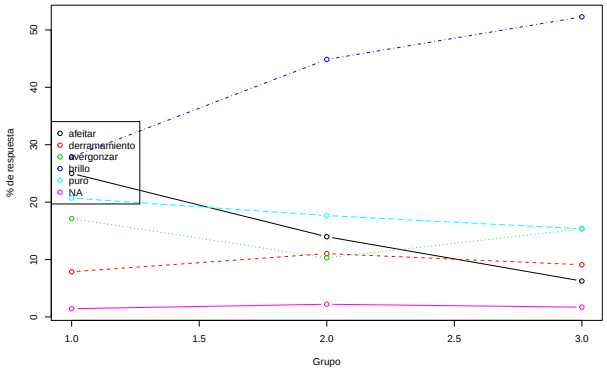

V45_guess respuesta correcta: adivinar
% de respuesta
Grupo
lapsus
adivinar
invitado
rencor
urogallo
NA

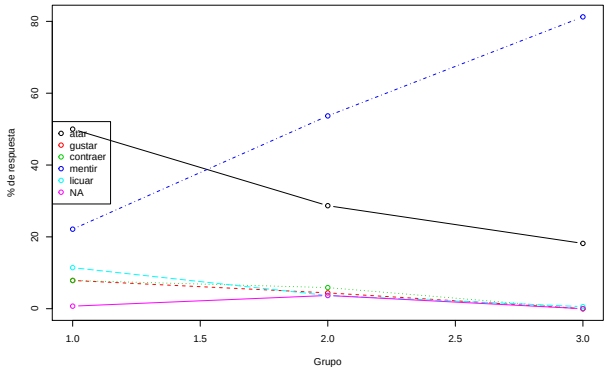

V46_guilty respuesta correcta: culpable
% de respuesta
Grupo
invitado
simpático
testigo
asesino
culpable
NA

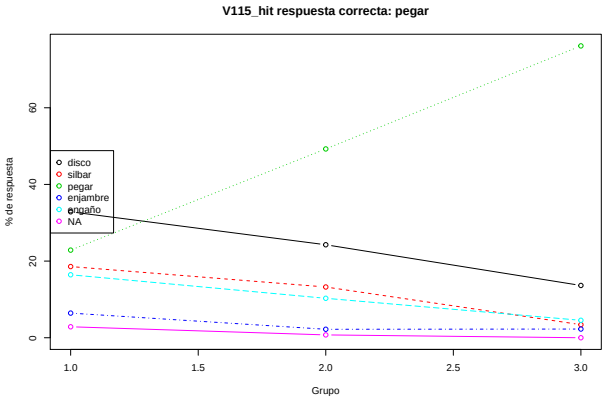

V51_hijack respuesta correcta: secuestrar
% de respuesta
Grupo
jinete
franja
membrillo
melena
secuestrar
NA

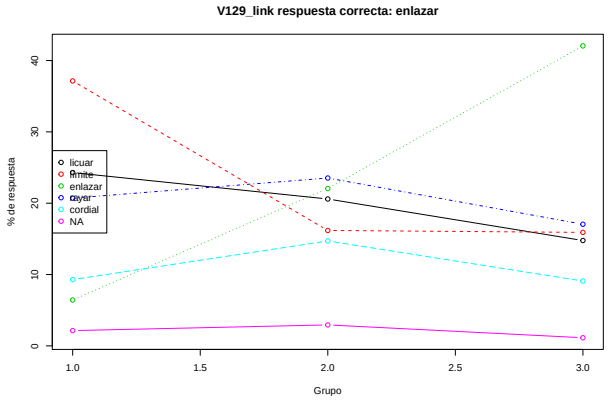

V58_upset respuesta correcta: trastornar
% de respuesta
Grupo
encima
repentino
altivez
indigno
trastornar
NA

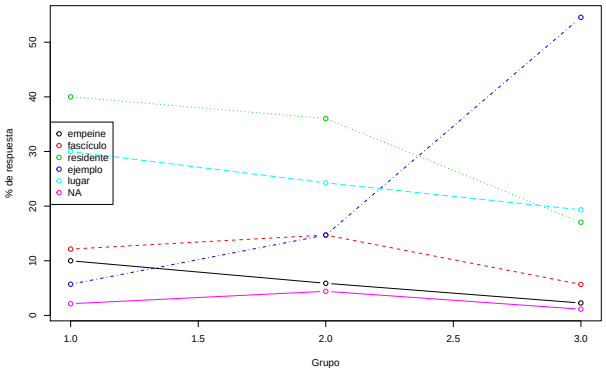

V60_compass respuesta correcta: brújula
% de respuesta
Grupo
compás
paso
comparación
compasión
brújula
NA

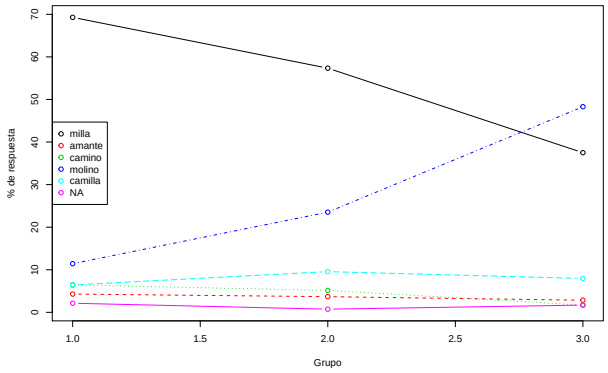

V68_sheen respuesta correcta: brillo
% de respuesta
Grupo
afeitar
derramamiento
avergonzar
brillo
puro
NA

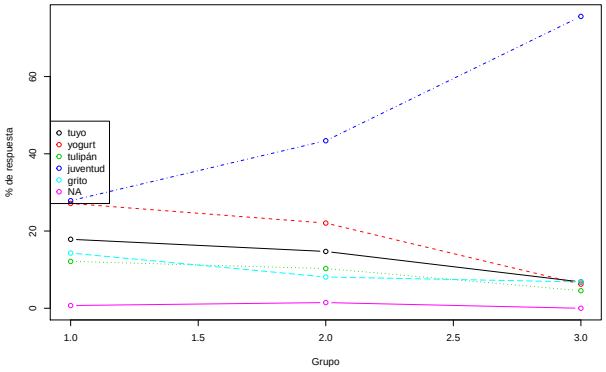

V87_lie respuesta correcta: mentir
% de respuesta
Grupo
gustar
contraer
mentir
licuar
NA

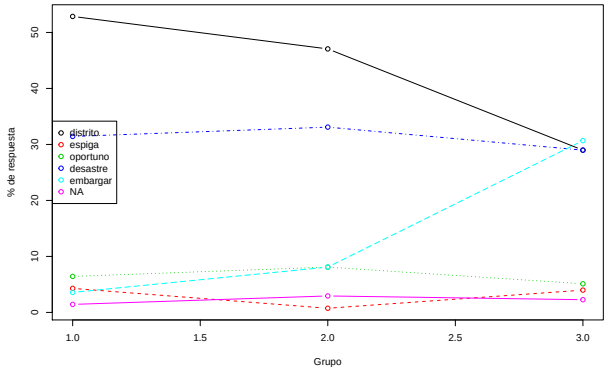

V115_hit respuesta correcta: pegar
% de respuesta
Grupo
disco
silbar
pegar
enjambre
engaño
NA

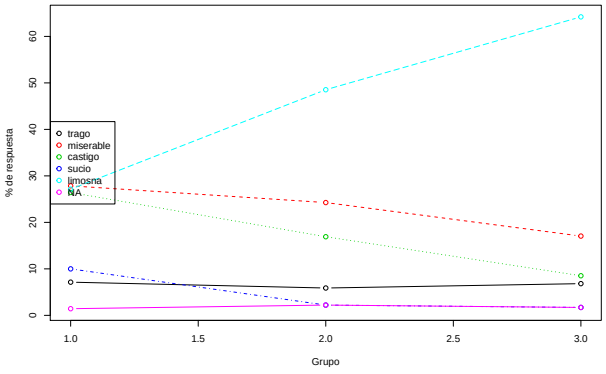

V129_link respuesta correcta: enlazar
% de respuesta
Grupo
licuar
mitin
enlazar
cavar
cordial
NA

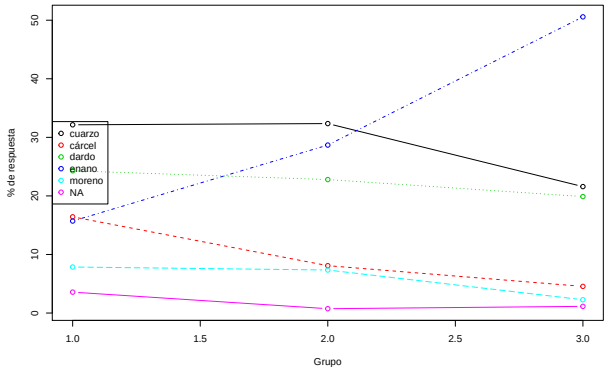

V151_instance respuesta correcta: ejemplo
% de respuesta
Grupo
empeine
fascículo
residente
ejemplo
lugar
NA


V183_mill respuesta correcta: molino
% de respuesta
Grupo
milla
amante
camino
molino
camilla
NA


V186_youth respuesta correcta: juventud
% de respuesta
Grupo
tuyo
yogurt
tulipán
juventud
grito
NA


V193_distrain respuesta correcta: embargar
% de respuesta
Grupo
distrito
espiga
oportuno
desastre
embargar
NA


V201_alms respuesta correcta: limosna
% de respuesta
Grupo
trago
miserable
castigo
sucio
limosna
NA


V203_dwarf respuesta correcta: enano
% de respuesta
Grupo
cuarzo
cárcel
dardo
enano
moreno
NA

Gracias a éstos gráficos podemos ver por ejemplo que en el ítem 2 uno de los distractores acumula una mayor cantidad de respuestas que la opción correcta, lo que nos permite intuir que es un ítem difícil. Además, el porcentaje de respuestas correctas, incluso en los niveles de competencia más altos, es bastante bajo.

Algunos casos similares e incluso más extremos son los de los items 51 o 58.

El ítem 68 también parece bastante difícil, ya que a penas lo aciertan el 50 % en el nivel de competencia más alto. Ademas, en los niveles de competencia bajos encontramos que los distractores funcionan de forma más intensa que en los niveles más altos.

Un ítem más fácil para los sujetos competentes podría ser el 87, y seguramente tendrá mayor discriminación. El 80 % de los sujetos del grupo de alta competencia lo acierta frente al 20 % de los de baja competencia.

Las métricas de discriminación y dificultad se deben explorar mediante otras mediciones más precisas, pero éstos gráficos nos ayudan a presentar la información de forma más visual.

Consistencia interna

En el paquete CTT tenemos disponible la función `itemAnalysis()`, que nos permite obtener, entre otras cosas, el indicador de consistencia interna alfa de Cronbach. Nos aporta además una tabla que indica el valor del indicador si se quita cada uno de los ítems.

```
fiabilidad <- itemAnalysis(Corregidos)

#Vemos que items generan un aumento de alfa al quitarlos
fiabilidad$itemReport[fiabilidad$itemReport$alphaIfDeleted
                      > fiabilidad$alpha, ]
```

```
##    itemName  itemMean       pBis        bis alphaIfDeleted
## 7       V68 0.4247788 0.01945952 0.02454899      0.6927784
## 15     V201 0.4800885 0.14138112 0.17727525      0.6770448
```

El alfa de Cronbach adopta el valor **0.67**. El indicador tiene un valor de 0.69 si se incluyen los 4 ítems eliminados por ser demasiado fáciles, pero es un incremento modesto comparado con la cantidad de ítems que hay que añadir para conseguirlo. Sería más conveniente coger otros ítems de la base de datos, pero continuemos con los que tenemos hasta la fecha.

Encontramos que tanto el ítem 68, como el 201 hacen que disminuya la consistencia interna, de modo que los retiramos. En ambos casos, al analizar el comportamiento de los distractores encontramos que la cantidad de personas que escogen los distractores decrece conforme aumenta su nivel de competencia, y que la cantidad de personas que seleccionan la opción correcta va aumentando conforme lo hace su competencia. En éste sentido, los ítems no dan la impresión de tener un comportamiento inadecuado.

Sin embargo, presentan unos índices de correlación ítem/test demasiado bajos, por lo que los retiramos de todas formas.

```
idxMantener <- !colnames(Datos) %in%
                 c("V68", "V201")
#Quitamos los items que reducen el alfa
Datos <- Datos[, idxMantener]
Respuestas <- Respuestas[idxMantener]
Items <- Items[idxMantener, ]
Corregidos <- Corregidos[, idxMantener]

#Repetimos el calculo de alfa
fiabilidad <- itemAnalysis(Corregidos)
```

Ahora alfa vale **0.6996**

Propiedades de los ítems

Podemos presentar las propiedades de los ítems que quedan ordenándolos por su índice de dificultad (la media del ítem, que representa la proporción de sujetos que lo han acertado debido a que está codificado como un 0 para los fallos y un 1 para los aciertos).

```
PropItems <- fiabilidad$itemReport
Temp <- PropItems[rev(order(PropItems$itemMean)), ]
kable(Temp)
```

	itemName	itemMean	pBis	bis	alphaIfDeleted
7	V87	0.5464602	0.3407807	0.4281643	0.6800508
8	V115	0.5154867	0.3424670	0.4293363	0.6798145
12	V186	0.5110619	0.3146424	0.3944008	0.6838700
3	V46	0.4712389	0.4022173	0.5045817	0.6709817
2	V45	0.3539823	0.2363205	0.3038603	0.6943524
14	V203	0.3318584	0.2436916	0.3161482	0.6931538
11	V183	0.2942478	0.3193822	0.4223207	0.6828913
10	V151	0.2743363	0.4660236	0.6239002	0.6634787
9	V129	0.2500000	0.3665525	0.4994764	0.6769663
4	V51	0.2101770	0.1997843	0.2822981	0.6969634
1	V2	0.2035398	0.2830547	0.4028240	0.6873692
6	V60	0.1592920	0.2440174	0.3680762	0.6915969
13	V193	0.1548673	0.3546508	0.5387500	0.6801145
5	V58	0.1150442	0.2828274	0.4647987	0.6881993

Vemos que la información que se extrae intuitivamente de los gráficos sobre la dificultad de los ítems no se traduce necesariamente en los índices de dificultad.

Podemos representar la información sobre el comportamiento de los distractores en tablas

```
Distractores <- distractorAnalysis(Datos, Respuestas)
for(i in 1:3){
  print(paste("Tabla del ítem ", Items$Item[i],
                          ": ", Items$Pregunta[i]))
  Temp <- Distractores[[i]]
  Temp[4:ncol(Temp)] <- round(Temp[4:ncol(Temp)], 2)
  print(kable(Temp))
  cat("\n")
}
```

[1] "Tabla del ítem 2 : deceptive"

	correct	key	n	rspP	pBis	discrim	lower	mid50	mid75	upper
1	*	1	92	0.20	0.28	0.46	0.04	0.15	0.21	0.51
2		2	6	0.01	-0.06	-0.02	0.02	0.01	0.03	0.00
3		3	9	0.02	-0.13	-0.02	0.02	0.04	0.00	0.00
4		4	40	0.09	-0.03	0.07	0.06	0.07	0.13	0.12
5		5	297	0.66	-0.48	-0.46	0.82	0.72	0.63	0.36
9		9	8	0.02	-0.13	-0.03	0.04	0.01	0.00	0.01

[1] "Tabla del ítem 45 : guess"

	correct	key	n	rspP	pBis	discrim	lower	mid50	mid75	upper
1		1	24	0.05	-0.23	-0.10	0.11	0.06	0.01	0.01
2	*	2	160	0.35	0.24	0.47	0.17	0.28	0.45	0.64
3		3	191	0.42	-0.24	-0.04	0.38	0.48	0.45	0.35
4		4	51	0.11	-0.36	-0.23	0.23	0.12	0.05	0.00
5		5	21	0.05	-0.23	-0.09	0.09	0.06	0.01	0.00
9		9	5	0.01	-0.06	-0.02	0.02	0.00	0.03	0.00

[1] "Tabla del ítem 46 : guilty"

	correct	key	n	rspP	pBis	discrim	lower	mid50	mid75	upper
1		1	42	0.09	-0.27	-0.13	0.17	0.12	0.01	0.03
2		2	61	0.13	-0.30	-0.17	0.22	0.16	0.07	0.04
3		3	90	0.20	-0.36	-0.25	0.28	0.26	0.17	0.02
4		4	41	0.09	-0.29	-0.15	0.15	0.12	0.06	0.00
5	*	5	213	0.47	0.40	0.73	0.16	0.35	0.69	0.89
9		9	5	0.01	-0.11	-0.02	0.03	0.00	0.00	0.01

En la mayoría de los casos vemos que las opciones incorrectas tienen correlaciones negativas con las puntuaciones del instrumento, así como índices de discriminación negativos. Interesa que éstas correlaciones sean altas en valor absoluto y que tengan el signo negativo. Las opciones correctas tienen correlaciones positivas con las puntuaciones.

Si miramos las propiedades de la respuesta correcta podemos ver el índice de dificultad del ítem (rspP), así como el índice de discriminación del ítem (discrim). Para aislarlo, podemos construir una función que procesa la salida de `distractorAnalysis()`.

Discriminación de los ítems

```
discrimItem <- function(df){
  df$discrim[df$correct == "*"]
}


Temp <- sort(sapply(Distractores, discrimItem),
             decreasing = TRUE)
Temp[1:5]
```

```
##      V151       V46       V87      V115      V186
## 0.7307116 0.7293071 0.7213483 0.6764045 0.6064607
```

```
Temp[6:10]
```

```
##      V129      V183      V203       V45        V2
## 0.5792135 0.5462547 0.4900749 0.4737828 0.4639513
```

```
Temp[11:length(Temp)]
```

```
##      V193       V51       V60       V58
## 0.4048689 0.3544944 0.3486891 0.2729401
```

Gracias a ésta función, podemos observar qué ítems tienen una mayor capacidad de discriminación. Podríamos quedarnos con los n ítems que presenten una mayor discriminación, pero como ya tenemos una cantidad bastante pequeña de ítems mantendremos todos los que tenemos hasta ahora.

Fiabilidad (técnica de las dos mitades)

```
#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)
```

El coeficiente de fiabilidad para el test mitad vale **0.3**. Tras aplicar la corrección de Spearman Brown para obtener la fiabilidad del test con la longitud original el valor asciende a **0.46**.

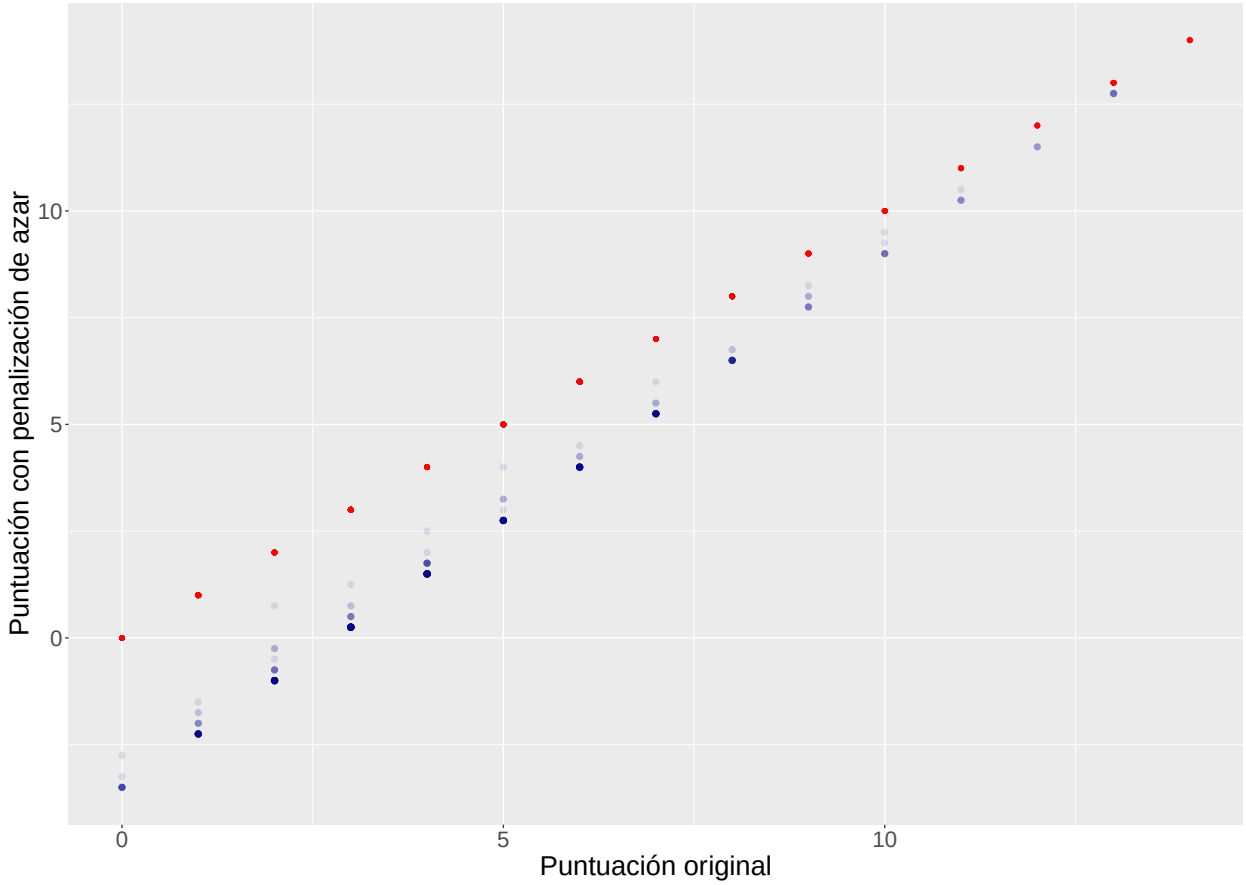
Con éstos datos, si deseamos obtener un coeficiente de fiabilidad de 0.8 necesitamos un total de 132 ítems.

Corrección con penalización del azar

```
k <- 5 #Numero de opciones de respuesta

Correcion <- data.frame(
  Aciertos = rowSums(Corregidos),
  Omisiones = rowSums(Datos == 9)
)

Correcion$Errores <- ncol(Datos) - Correcion$Aciertos -
  Correcion$Omisiones

Correcion$PuntCorreg <- Correcion$Aciertos - ((1 / (k - 1)) *
                        Correcion$Errores)
```

A continuación, podemos representar las puntuaciones en un diagrama de puntos.

```
ggplot(Correcion, aes(x = Aciertos, y = PuntCorreg)) +
  geom_point(alpha = 0.1,
             size = 2,
             color = "darkblue") +
  geom_point(x = Correcion$Aciertos, y = Correcion$Aciertos,
             color = "red") +
  xlab("Puntuación original") +
  ylab("Puntuación con penalización de azar") +
  theme(text = element_text(size = PlotFont))
```



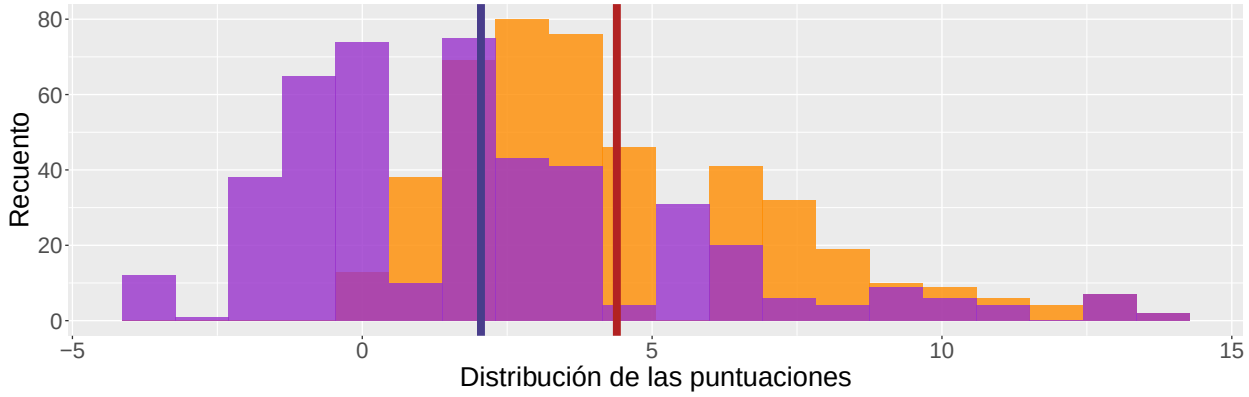
En el gráfico los puntos azules representan los valores para la puntuación original VS la puntuación corregida.

Se ha añadido además una línea de puntos roja que representa la puntuación original (es decir la línea roja sería la puntuación original contra sí misma). Ésta línea roja nos permite apreciar que las puntuaciones corregidas son siempre inferiores a las originales.

Además, la diferencia entre la puntuación con corrección al azar y la original es mayor en los niveles de competencia más bajos. Ésto es así dado que los sujetos con un menor nivel de competencia deben una mayor proporción de sus respuestas correctas al azar.

En el siguiente gráfico, podemos ver la distribución de ambas puntuaciones. Las puntuaciones originales aparecen representadas en naranja, y su media es la barra vertical roja. Las puntuaciones con corrección por azar están representadas en morado, y su media por la barra vertical roja.

```
ggplot() +
  geom_histogram(aes(Correcion$Aciertos),
                 bins = 20, fill = Palette[3], alpha = 0.8) +
  geom_histogram(aes(Correcion$PuntCorreg),
                 bins = 20, fill = Palette[1], alpha = 0.8) +
  geom_vline(xintercept = mean(Correcion$Aciertos),
             color = Palette[2], size = 3) +
  geom_vline(xintercept = mean(Correcion$PuntCorreg),
             color = Palette[4], size = 3) +
  xlab("Distribución de las puntuaciones") +
  ylab("Recuento") +
  theme(text = element_text(size = PlotFont))
```



La correlación entre ambas puntuaciones es muy alta: 0.9987

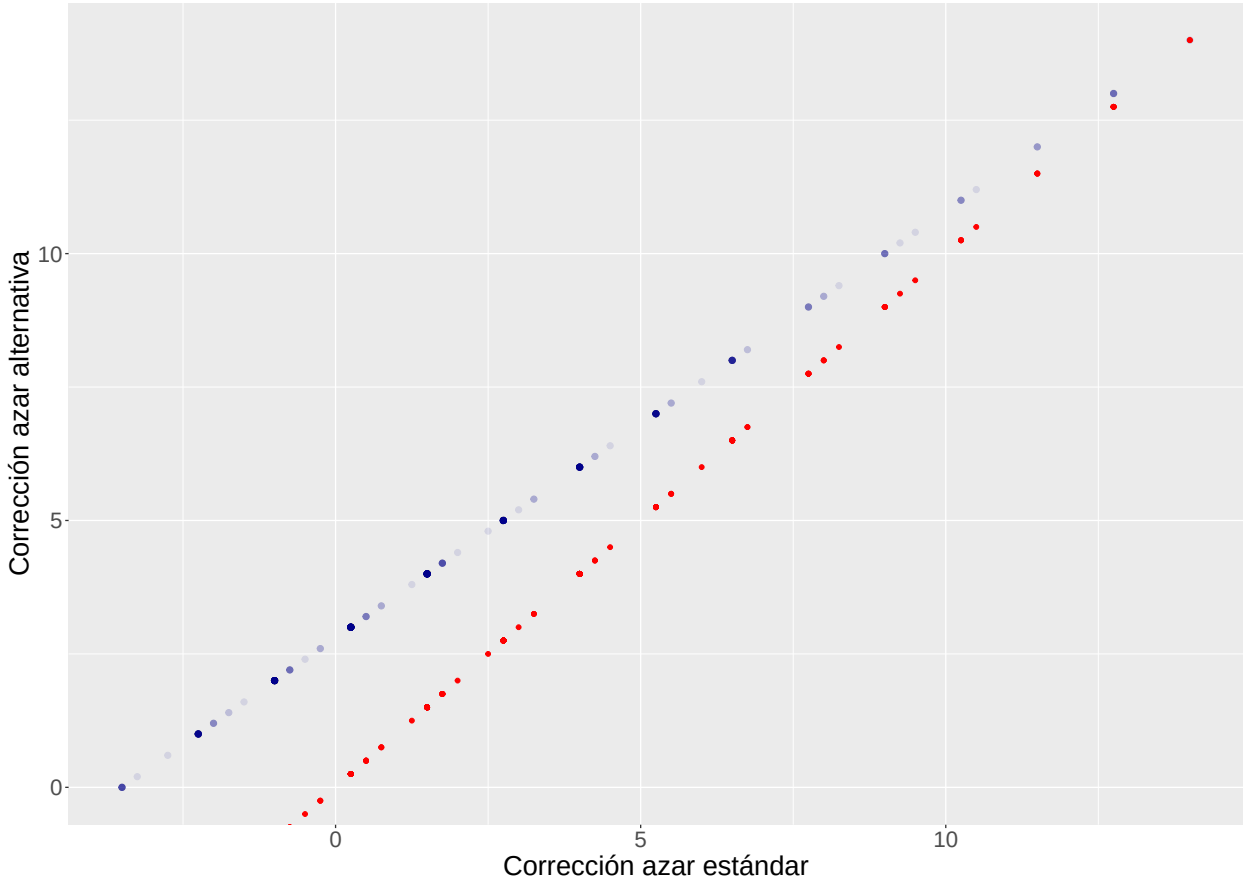
Tarea voluntaria

Corrección del azar alternativa

```
#Aplicamos una correccion del azar alternativa
Correcion$PuntCorreg2 <- Correcion$Aciertos +
  (Correcion$Omisiones / k)
```

Comparamos las puntuaciones con las correciones por azar realizadas mediante ambos métodos. En éste caso, los puntos rojos representan la puntuación con la primera corrección del azar contra sí misma, mientras que los puntos azules representan ambas correcciones una contra la otra.

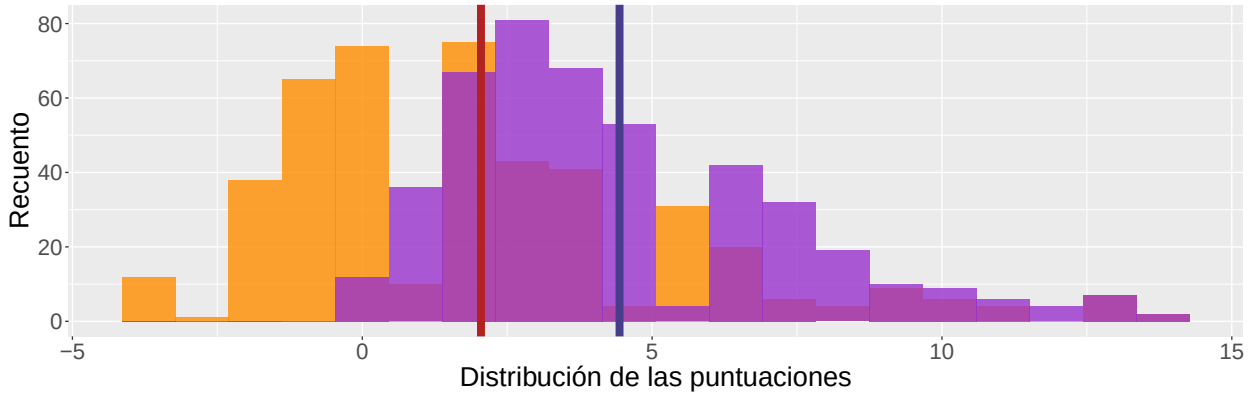
```
ggplot(Correcion, aes(x = PuntCorreg, y = PuntCorreg2)) +
  geom_point(alpha = 0.1,
             size = 2,
             color = "darkblue") +
  geom_point(x = Correcion$PuntCorreg, y = Correcion$PuntCorreg,
             color = "red") +
  xlab("Corrección azar estándar") +
  ylab("Corrección azar alternativa") +
  theme(text = element_text(size = PlotFont))
```



Ésta nueva forma de corregir por azar beneficia más a los sujetos con un nivel de competencia más bajo, ya que no penaliza los errores. A medida que aumenta el nivel de competencia, la distancia entre ambas puntuaciones se reduce.

En éste histograma tenemos la primera corrección en naranja (su media en rojo), y la segunda en morado (media en azul).

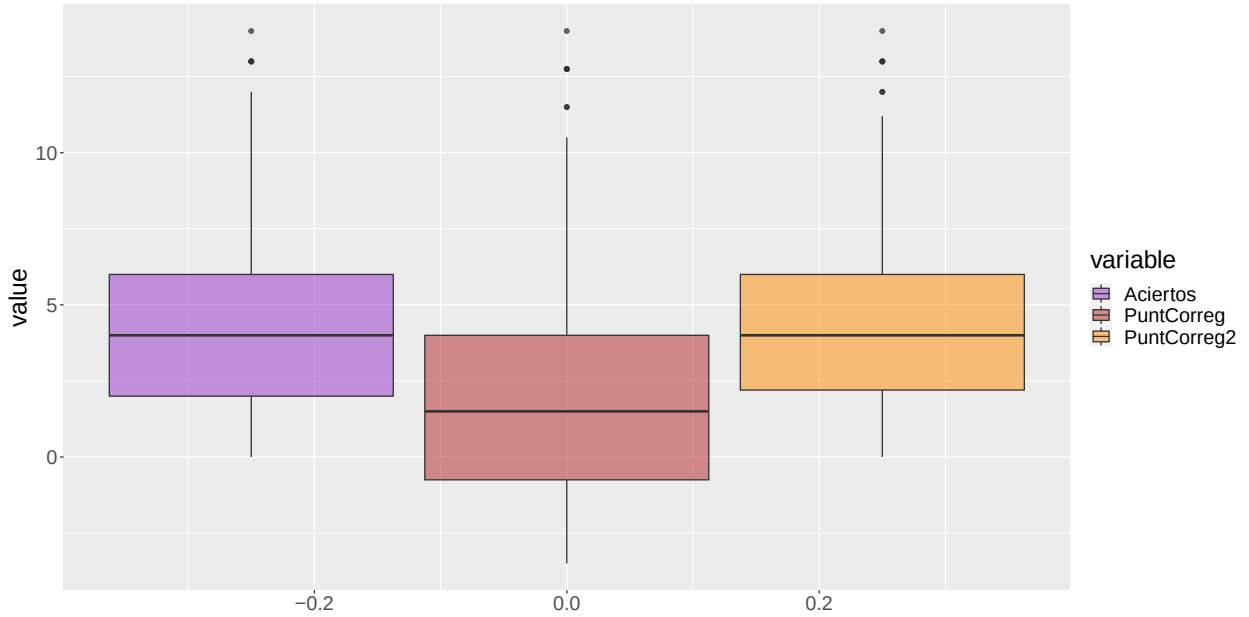
```
ggplot() +
  geom_histogram(aes(Correcion$PuntCorreg),
                 bins = 20, fill = Palette[3], alpha = 0.8) +
  geom_histogram(aes(Correcion$PuntCorreg2),
                 bins = 20, fill = Palette[1], alpha = 0.8) +
  geom_vline(xintercept = mean(Correcion$PuntCorreg),
             color = Palette[2], size = 3) +
  geom_vline(xintercept = mean(Correcion$PuntCorreg2),
             color = Palette[4], size = 3) +
  xlab("Distribución de las puntuaciones") +
  ylab("Recuento") +
  theme(text = element_text(size = PlotFont))
```



Claramente, podemos ver que la nueva corrección tiende a generar puntuaciones más altas aunque su distribución es similar. La correlación entre ambas es casi perfecta.

Podemos representar todas las puntuaciones en un diagrama de caja y bigotes. Aciertos representa las puntuaciones sin corrección del azar (suma de los aciertos). PuntCorreg representa la primera corrección del azar, mientras que PuntCorreg2 representa la segunda

```
DatosGrafico <- Correcion
DatosGrafico[, c("Omisiones", "Errores")] <- NULL
DatosGrafico$Sujeto <- rownames(DatosGrafico)
DatosGrafico <- reshape2::melt(DatosGrafico, id.vars = "Sujeto")

ggplot(DatosGrafico, aes(fill = variable, y = value)) +
  geom_boxplot(alpha = 0.5) +
  scale_fill_manual(values = Palette) +
  theme(text = element_text(size = 20))
```



Podemos ver que la segunda corrección del azar genera unas puntuaciones con una

distribución más semejante a la de las puntuaciones originales, y que la primera corrección del azar presenta una mayor distancia entre los percentiles 25 y 75, por lo que hay más dispersión en las puntuaciones.

Mediciones adicionales de fiabilidad

El paquete psych nos ofrece la posibilidad de explorar otras métricas para evaluar la fiabilidad del test. Las aplicamos sobre los datos de cálculo numérico.

```
Datos <- read.spss(Rutas$CalNum,
                   to.data.frame = TRUE)
Datos <- Datos[, 15:29]

Respuestas <- c(3, 1, 3, 4, 4, 2, 4, 4, 3, 4, 4, 3, 3, 3, 1)
Corregidos <- score(Datos, Respuestas,
                    output.scored = TRUE)
Corregidos <- Corregidos$scored

splitHalf(Corregidos)
```

```
## Split half reliabilities
## Call: splitHalf(r = Corregidos)
##
## Maximum split half reliability (lambda 4) =  0.76
## Guttman lambda 6                          =  0.7
## Average split half reliability            =  0.7
## Guttman lambda 3 (alpha)                  =  0.7
## Guttman lambda 2                          =  0.71
## Minimum split half reliability  (beta)    =  0.59
## Average interitem r =  0.14  with median =  0.14
```

Además, el paquete ltm nos permite construir unos intervalos de confianza en torno a los valores de alfa.

```
res <- cronbach.alpha(Corregidos,
                      CI = TRUE)
res
```

```
##
## Cronbach's alpha for the 'Corregidos' data-set
##
## Items: 15
## Sample units: 1000
## alpha: 0.71
##
## Bootstrap 95% CI based on 1000 samples
```

```
##  2.5% 97.5%
## 0.681 0.734
```

En determinadas situaciones el coeficiente alfa puede no ser del todo preciso. Una alternativa es el coeficiente omega, que se puede obtener combinando los paquetes lavaan y semTools.

```
fit1f <- cfa(mod1f,
             data = Corregidos)
semTools::reliability(fit1f)
```

```
##                F1
## alpha  0.7104368
## omega  0.7177343
## omega2 0.7177343
## omega3 0.7185286
## avevar 0.1613044
```