Bye, bye SPSS (y SAS): hola R para encuestas

VII Jornadas de usuarios de R
Salamanca, 5-6 noviembre 2015

José Ignacio Casas

Herramientas para el taller

- **R**: ¿R version 3.2.2 (2015-08-14)?
- **RStudio**: ¿Version 0.99.486?
- **Paquetes** (y sus adjuntos):
  - Imprescindibles: {memisc} {sjmisc} {sjPlot}
  - Muy convenientes: {foreign} {Hmisc} {haven} {MicroDatosEs} {sas7bdat} {likert}
  - Convenientes: {Rz} {splitstackshape}
  - Prescindibles: {Rsocialdata} {surveydata}
- Conexión a **Internet**, **navegador**, **editor** de ficheros planos (¿Notepad++?) y lector **.pdf**
- Conocimientos **básicos** de R
- **Materiales** de trabajo en:
  `www.jomialresearch.com/Rparaencuestas.zip`

Agenda

- Por qué este taller: compartir mi trayectoria y mi exploración
- SPSS (y SAS) vs. R para encuestas
- Volcando SPSS (y SAS) a R
- R sin SPSS (ni SAS) manejo de datos con **`{memisc}`** y **`{sjmisc}`**
- Tablas y gráficos básicos con **`{sjPlot}`**
- Otros:
  - casestovars
  - escalas tipo Likert
  - preguntas multi-respuesta
- Algunos análisis con {sjmisc} y {sjPlot}

R: una selva llena de tesoros...

… donde no hay mapas definitivos ni únicos

Propuesta de taller: no un tratado sino un libro de viajes

R para encuestas: no hay una autopista en la selva...

… sino una(s) vereda(s) que invito a recorrer

Objetivo: encontrar nuestro camino más conveniente

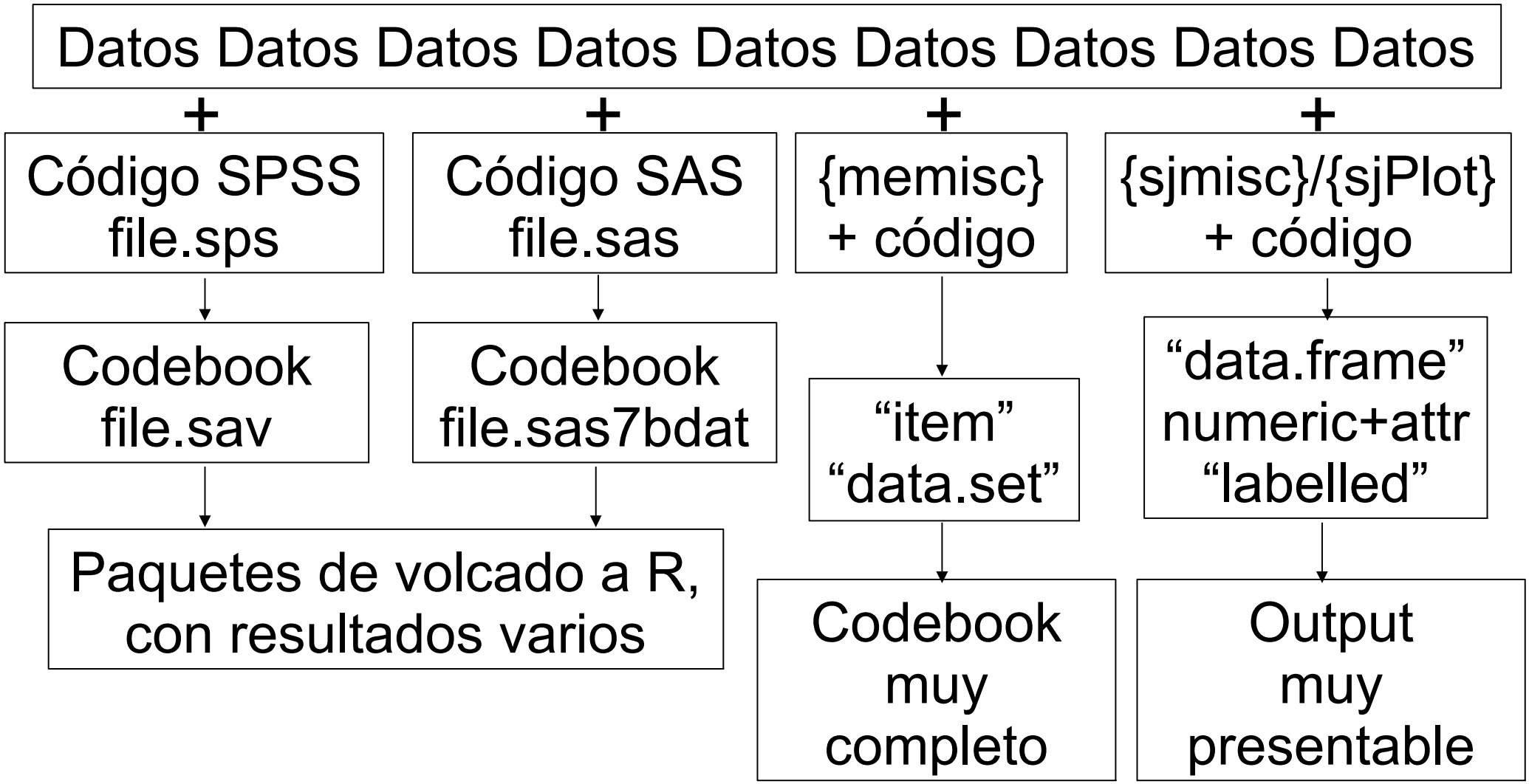
Agenda

- Por qué este taller: compartir mi trayectoria y mi exploración
- SPSS (y SAS) vs. R para encuestas
- Volcando SPSS (y SAS) a R
- R sin SPSS (ni SAS) manejo de datos con **`{memisc}`** y **`{sjmisc}`**
- Tablas y gráficos básicos con **`{sjPlot}`**
- Otros:
  - casestovars
  - escalas tipo Likert
  - preguntas multi-respuesta
- Algunos análisis con {sjmisc} y {sjPlot}

SPSS (y SAS) vs. R para encuestas

- El análisis de encuestas:
  - componente descriptivo central: tablas y gráficos de frecuencias y de doble entrada
  - importancia de los “labels”
  - gestión de los valores “missing”
- R para encuestas: un terreno abierto y en ebullición
  - R ofrece una gama extensa de operaciones: desde la captura de datos hasta la presentación de informes
  - muchos paquetes (en evolución) para el volcado de ficheros desde SPSS y otros
  - todavía no hay una única solución consolidada

Planteamiento general



Agenda

- Por qué este taller: compartir mi trayectoria y mi exploración
- SPSS (y SAS) vs. R para encuestas
- Volcando SPSS (y SAS) a R
- R sin SPSS (ni SAS) manejo de datos con **`{memisc}`** y **`{sjmisc}`**
- Tablas y gráficos básicos con **`{sjPlot}`**
- Otros:
  - casestovars
  - escalas tipo Likert
  - preguntas multi-respuesta
- Algunos análisis con {sjmisc} y {sjPlot}

Volcando SPSS a R: paquetes

- foreign
- Hmisc
- haven
- memisc
- sjmisc / sjPlot
- Otros paquetes:
  - Rz
  - Rsocialdata
  - surveydata

SPSS → R (1): read.spss {foreign}

```
> library(foreign)
> spssf1 <- read.spss ("Bar1502corto.sav")
```

Por defecto:

- Devuelve una “list” con un componente por variable
- Convierte todas las variables “character” a “factor”
- Attributes:
  - **label.table**: guarda los “value labels” en cada variable (= **levels** de cada “factor”)
  - **variable.labels**: guarda **juntos** todos los “variable labels” de las variables
  - **missings**: valores de cada variable definidos como missing en SPSS (no los sysmis)
  - **names**: los nombre de las variables pasados a MAYÚSCULAS

SPSS → R (2): read.spss {foreign}

```
> spssf1.df <- as.data.frame (spssf1)
```

Pierde los attributes “variable.labels”

Los “missings” se convierten en un factor más

Conversión directa a data.frame:

```
> spssf2 <- read.spss ("Bar1502corto.sav",
   to.data.frame = TRUE)
```

- Attributes:
  - Los “**value labels**” se convierten en levels de cada “factor”
  - **variable.labels**: guarda los “variable labels” de cada variable
  - **missings**: son convertidos a NAs

SPSS → R (3): spss.get {Hmisc}

```
> library (Hmisc)
> spsshm <- spss.get ("Bar1502corto.sav")
```

- Invoca read.spss {foreign} con to.data.frame = TRUE
- Las variables son clase "**labelled**" y factor o numeric
- Los NAs se convierten en **missings**
- Los variable labels y value labels se guardan como atributos **en cada variable** ("label" & "levels"). Ver:

```
> attributes(spsshm$CCAA)
```

SPSS → R (4): spss.get {haven}

```
> library (haven)
> spsshv <- read_spss ("Bar1502corto.sav")
```

- Crea un fichero de clase data.frame (también "tbl_df" y "tbl"), con variables clase "**labelled**"
- Todos los "**missings**" se convierten en NAs
- Los variable labels y value labels se guardan como atributos **en cada variable** ("label" & "labels"):

```
> attributes(spsshv$CCAA)
```

SPSS → R (5): variables “labelled”

- SPSS y SAS tienen variables “labelled”, parecidas a un “factor”, pero
  - Cualquier valor puede tener un “label”
  - Puede haber valores sin “label”
- En R un “factor” tiene siempre valores “integer” y todo “integer” debe tener un “label” (“level”)
- {Hmisc} y {haven} proporcionan la clase “labelled” para modelar estos objetos. El paquete {haven} incluye
  - **as_factor()**: convierte las variables **labelled integers** en **factors**. Los valores sin un **label** asociado se convierten en **missing**. Hay que usar esta función, ya que **as.factor()** no funciona con variables **labelled**
  - **zap_labels()**: convierte todos los valores **labelled** en **missing**. Esto es útil cuando se tiene una variable continua con algunos valores etiquetados como **missing** (ej. 99)

SPSS → R (6): spss.system.file {memisc}

```
> library (memisc)
> spssm1 <- spss.system.file
  ("Bar1502corto.sav")
```

- Crea un fichero de clase “spss.system.importer”, que refiere a un fichero de datos externo (ver otros “**importers**”)
- **`print (spssm1)`** # Comprobamos que tenemos un fichero SPSS.sav, con su número de variables y de casos
- **`names (spssm1)`** # nombres de las variables, en **minúsculas**
- **`description (spssm1)`** # “variable labels” de las variables

SPSS → R (6): spss.system.file {memisc}

- **`labels(spssm1["nombreVariable"])`** # value labels de esa variable
- **`codebook (spssm1)`** # descriptivos de todas las variables: tabla de frecuencias con value labels, abs y porcentajes totales y totales sin missing
- **`codebook(spssm1["ccaa"])`** # idem para una sola variable
- Los “missing” se almacenan en al atributo “value.filter” y distingue entre “sysmis” y “usermis”
- ¡OJO!: cuando una variable tiene **valores con y sin** “labels” **ignora los valores sin** “labels” en el codebook

SPSS → R (7): data.set {memisc}

- **`spssm2 <- as.data.set(spssm1)`** # obtenemos un "data.set"
- Las variables se convierten en un objeto **item**:
- **storage mode**: “integer”, “double” ...
- **measurement**: Tipo de medida del objeto "item": "nominal", "ordinal", "interval", "ratio". Determina su formato al volcarlo a un data.frame. Por ejemplo: "nominal" - > unordered
- **annotation**: description / wording
- **labels**: attributes asociados a cada **value**

SPSS → R (8): {sjmisc}/{sjPlot}

```
> library (sjmisc) # gestiona los datos
> library (sjPlot) # proporciona tablas y gráficos
> spsssj <- read_spss ("Bar1502corto.sav")
```

Por defecto utiliza la función del mismo nombre del paquete 'haven', pero las variables **YA NO SON DE CLASS LABELLED**

```
> view_df (spsssj)
```
 # presenta estructura y "labels" del fichero en formato HTML

```
> view_df (spsssj, showFreq = TRUE, showPerc = TRUE)
```
 # Lo mismo, pero ahora con frecuencias y porcentajes (como un "codebook")

```
> sjt.frq (spsssj$CCAA)
```
 # Para una sola variable

Se pueden cambiar los rótulos con las opciones `'string... = "..."'`

```
> frq (as_labelled (spsssj$CCAA))
```

SPSS → R (9): {sjmisc}/{sjPlot}

> **`get_label (spsssj)`** # devuelve los "variable labels"

> **`get_labels (spsssj)`** # devuelve los "value labels"

> **`get_na (spsssj$P31A)`** # devuelve los códigos asociados a los "missing values" de la variable

> **`spsssj$CCAA <- set_label (spsssj$CCAA, "nuevo rótulo")`** # Asigna un "variable label" a la variable. Lo mismo para un data.frame, con un vector de la misma longitud

> **`spsssj$CCAA <- set_labels (spsssj$CCAA, c("Nuevo1", "Nuevo2", "Nuevo3", "Nuevo4", ...))`** # Asigna "value labels" a la variable

> **`spsssj$P31A <- set_na(spsssj$P31A, c(98, 99), as.attr = TRUE)`** # Asigna "missing values" a la variable

La opción (**`as.attr = TRUE`**) convierte a missing pero no a NA. Por defecto (**`as.attr = FALSE`**) pasan a ser NAs

SPSS → R (10): otros paquetes

- **Rz**
  - Crea un GUI tipo menú para manejar datos (tablas y gráficos) al estilo SPSS
  - Usa el paquete “memisc”
- **Rsocialdata**
  - `spssrsd <- get.spss.file ("Bar1502corto.sav")`
  - En desarrollo
- **surveydata**
  - En desarrollo
  - No gestiona los missing values

Volcando SAS a R: paquetes

- foreign
- Hmisc
- haven
- sas7bdat (experimental)
- sjmisc / sjPlot

- El paquete **SASxport** está retirado de CRAN
- **memisc** no captura ficheros SAS

SAS → R (1): con {foreign}

```
> library (foreign)
```

> **`read.ssd("file.ssd")`** # convierte fichero SAS a formato xport: **necesita tener SAS instalado**

> **`lookup.xport("DEMO_H.XPT")`** # inspecciona un fichero SAS en formato xport y devuelve una **list** conteniendo información

> **`sasf <- read.xport("DEMO_H.XPT")`** # lee fichero SAS en formato xport y devuelve una **list** de data.frames o un **data.frame**

- Parece que se pierden los “variable labels” y “value labels”
- Los NAs vuelcan a NAs

SAS → R (2): con {Hmisc}

```
> library (Hmisc)
```

> **sas.get("file")** # convierte fichero SAS a data.frame: **necesita tener SAS instalado**

```
> sashm <- sasxport.get("DEMO_H.XPT")
```

lee fichero SAS en formato xport y devuelve un **data.frame**

- Las variables son clase “labelled” y factor o numeric
- Los NAs se convierten en missings
- Los variable labels se guardan como atributos en cada variable (“label”). Parece que se pierden los “value labels”:

```
> attributes(sashm$bpaen2)
```

SAS → R (3): con {haven}

```
> library (haven)
> sashv <- read_sas("airline.sas7bdat")
```

lee fichero SAS en formato **sas7bdat** y devuelve un **data.frame**

- Los NAs se convierten en missings
- Las variables **NO** son de la class “labelled”
- Los “variable labels” se guardan como atributos en cada variable (“label”). Parece que se pierden los “value labels”:

```
> attributes(sashv$Y)
```

SAS → R (4): con {sas7bdat} (1)

Está en fase experimental:

```
> library (sas7bdat)
> sas7 <- read.sas7bdat("airline.sas7bdat")
```

lee fichero SAS en formato **sas7bdat** y devuelve un **data.frame**

- Los NAs se convierten en missings
- Las variables **NO** son de la class "labelled"
- Los "variable labels" se guardan como atributos en una **list**. Parece que se pierden los "value labels":

```
> attr(sas7, "column.info")[[2]]$label
```

SAS → R (5): con {sas7bdat} (2)

El atributo **column.info** es una **list** de listas, conteniendo:

- **name**: nombre de la variable
- **offset**: ”The field offset in packed binary row data (bytes)” (?)
- **length**: longitud de la variable (bytes)
- **type**: tipo de variable: ’character’ o ’numeric’

Cuando la base de datos especifica el formato y/o “label” también aparece:

- **format**: formato de display de la variable
- **label**: la “variable label” de la variable

```
> attr(sas7, "column.info")[[2]]
```

SAS → R (6): con {sjmisc} / {sjPlot}

```
> library (sjmisc)
> library (sjPlot)
> sassj <- read_sas ("airline.sas7bdat")
```

lee fichero SAS en formato **`sas7bdat`** y devuelve un **data.frame**

- Los NAs se convierten en missings
- Las variables **NO** son de la class “labelled”
- Los “variable labels” se guardan como atributos en cada variable (“label”). Parece que se pierden los “value labels”:

```
> attributes(sassj$Y)
> get_label (sassj)
```

Agenda

- Por qué este taller: compartir mi trayectoria y mi exploración
- SPSS (y SAS) vs. R para encuestas
- Volcando SPSS (y SAS) a R
- R sin SPSS (ni SAS) manejo de datos con `{memisc}` y `{sjmisc}`
- Tablas y gráficos básicos con `{sjPlot}`
- Otros:
  - casestovars
  - escalas tipo Likert
  - preguntas multi-respuesta
- Algunos análisis con {sjmisc} y {sjPlot}

Creando un objeto 'item' {memisc} (1)

Creamos un vector "normal":

```
> voto <- sample (c (1, 2, 3, 4, 5, 8, 9, 97, 99), size = 90, replace = TRUE)
```

Le añadimos "value labels" con memisc::labels, para convertirlo en 'item' (se pueden añadir (+) o quitar (-)):

```
> labels(voto) <- c( "Partido Popular" = 1, PSOE = 2, Ciudadanos = 3, Podemos = 4, "Otros partidos" = 5, "No sabe" = 8, "No contesta" = 9, "No votará" = 97, "No aplicable" = 99)
```

Determinamos los valores "missing":

```
> missing.values(voto) <- c(97,99)
```

También podemos usar "**valid.values**" o "**valid.range**"

Creando un objeto 'item' {memisc} (2)

Añadimos "literatura":

```
> description(voto) <- "Intención de voto"
```

```
> wording(voto) <- "Si mañana se celebraran las elecciones al Parlamento Nacional, ¿a qué partido o coalición votaría?"
```

Comprobamos lo que hemos añadido:

```
> annotation(voto)
```

```
> annotation(voto)["Mi comentario"] <- "Todo parecido con la realidad es pura coincidencia"
```

```
> annotation(voto)["Mi segundo comentario"] <- "¡Qué más quisieran algunos...!"
```

Contemplamos nuestra obra:

```
> codebook(voto)
```

Creando un objeto 'item' {memisc} (3)

```
voto 'Intención de voto'

"Si mañana se celebraran las elecciones al Parlamento Nacional, ¿a qué partido o coalición votaría?"

-------------------------------------------------------------------------------------------------------

Storage mode: double
Measurement: nominal
Missing values: 97, 99

     Values and labels    N    Percent

 1   'Partido Popular'    5    7.4  5.6
 2   'PSOE'              14   20.6 15.6
 3   'Ciudadanos'        12   17.6 13.3
 4   'Podemos'           12   17.6 13.3
 5   'Otros partidos'     6    8.8  6.7
 8   'No sabe'           10   14.7 11.1
 9   'No contesta'        9   13.2 10.0
97 M 'No votará'          9        10.0
99 M 'No aplicable'      13        14.4

Mi comentario:

    Todo parecido con la realidad es pura coincidencia

Mi segundo comentario:

    ¡Qué más quisieran algunos...!
```

Creando un objeto 'item' {memisc} (4)

Cambiar el tipo de "values":

```
> voto <- as.item (voto, measurement = "interval")
```

Opciones: "nominal", "ordinal", "interval", or "ratio"

Recodificar valores (y sus “labels”):

```
> distribucion <- recode(voto, Bipartidismo = 1 <- range(min,2), "Nuevos partidos" = 2 <- 3:4, otherwise = "copy")
```

Cambiamos uno o varios "labels":

```
> completo <- relabel (voto, PSOE = "Partido Socialista")
```

Creando un 'data.set' {memisc} (1)

```
> library (memisc)
```

Creamos un 'data.set' a partir de los datos del Barómetro del CIS de feb.2015, volcando los datos desde un fichero de ancho fijo:

```
> posiciones.fichero <- c(-49, 2, -54, 1, -143, 1) #
```
Creamos el vector de las posiciones que vamos a incorporar

```
> nombre.vars <- c("P9", "P32", "ESTATUS") # Creamos el
```
vector de los nombres de las variables

```
> bar1502.ds <- read.fwf
  ("C:/Taller_R_para_Encuestas/BarómetroCIS15M02/DA3052",
  widths = posiciones.fichero, col.names = nombre.vars)
```

Lo hacemos data.set:

```
> bar1502.ds <- data.set(bar1502.ds)

> names (bar1502.ds) <- nombre.vars
```

"Amueblamos" el data.set: (ver script **RpE6_memisc.r**, líneas 60ss)

Creando un 'data.set' {memisc} (2)

Se puede crear un subgrupo (= subset):

```
> mujer.bar1502.ds <- subset(bar1502.ds,
    P32 == "Mujer")
```

Confeccionamos una primera tabla de ejemplo:

```
> xtabs(~ESTATUS+P32,data=bar1502.ds)
```

Se pueden incluir los "missing":

```
> xtabs (~ESTATUS + P32, data = within
    (bar1502.ds, ESTATUS <- include.missings
    (ESTATUS)))
```

Creando un data.frame con {memisc}

```
> bar1502.df <- as.data.frame(bar1502.ds)
```

- "as.data.frame()" convierte cada 'item' en un **vector numérico** o en un **factor ordenado o no ordenado**, dependiendo del "**measurement**" de cada 'item'
- Los **missing values** se convierten en **NA**s y los **user-declared missing** values también se convierten en **NA**s
- Muchos de los métodos aplicables a un data.frame son **también aplicables** a un data.set

{MicroDatosEs} (1)

- Paquete creado por **Carlos G.Bellosta**, para el tratamiento de **micro-datos** del **INE**
- En CRAN la versión **0.6.3.1** y en R-Forge la **0.7**, más ampliada:

```
> install.packages ("MicroDatosEs", repos =
  "http://R-Forge.R-project.org")
```

- Lee directamente el fichero de datos y añade todas las características de un data.set, class memisc
- Es necesario que los ficheros originales mantengan su estructura

```
> library(MicroDatosEs)
```

Cargamos un fichero de ejemplo:

```
epa15q2 <- epa2005 ("
  C:/Taller_R_para_Encuestas/INE_EPA15Q2/EPA15Q2.dat
  ")
```

{MicroDatosEs} (2)

- Obtenemos un objeto de clase **data.set** formado por variables de clase **item**
- Podemos obtener tablas de valores absolutos ponderados:

```
> Table(epa15q2$situ, weights =
  (epa15q2$factorel/100))
```

- También podemos obtener porcentajes ponderados:

```
> percent (epa15q2$situ, weights =
  (epa15q2$factorel/100))
```

- También podemos calcular la distribución porcentual de una variable ('situ') en función de otra ('sexo'):

```
> Aggregate (percent (situ, weights =
  (factorel/100)) ~sexo, data = epa15q2)
```

Creando un 'labelled' con {sjmisc}

```
> sjl1 <- labelled (  # función
   c (X, Y, Z, ...),  # vector de valores numeric / character
   c ("cadena" = Y, ...), # labels. Mismo “typeof”. Fija el orden
   c (FALSE, FALSE, TRUE, FALSE, …)  # Señala los “missing” (=TRUE)
   )
> sjl1 <- set_label (sjl1, "cadena") # Añadimos el "variable label"
> class (sjl1) # Comprobamos qué tipo clase de objeto tenemos
> sjl1 # Más completo
> frq(sjl1)  # Tabla de frecuencias. Al ser  "labelled" no necesita adaptación
# En formato "bonito" (HTML), se necesita "to_factor":
> library(sjPlot)
> sjt.frq(to_factor(sjl1))
```

Creando un 'data.frame' {sjmisc}

```
> sjdf1 <- data.frame (sexo = c (1, 2, 2, ...), interes
=  c (1, 2, 1, 9, ...), habitat = c (1, 1,  ...))
```

Los valores **no** tienen que ser correlativos; se ordenan siempre de **menor a mayor**; sólo utilizar **numeric**

```
> sjdf1$sexo <- set_labels (sjdf1$sexo, c("Hombre",
"Mujer"))
```
los 'value labels' se adjuntan **en el orden de los valores**, de menor a mayor

```
> sjdf1 <- set_label (sjdf1, vectorDeLabels)
```
añadimos los "variable label" como un atributo que se incorpora **a cada variable**

```
> view_df(sjdf1, enc = "latin")
```
Comprobación

```
> sjdf1$interes <- set_na(sjdf1$interes,9,as.attr =
TRUE)
```
Asigna **missing** values

```
> get_na (sjdf1$interes)
```
Comprobamos 'missing'

```
> get_na_flags (sjdf1$interes)
```
Comprobamos 'missing'

Con 'value labels' incompletos {sjmisc}

- Cuando no todos los valores tienen 'value label'
- Ver script **RpE10_incompl_value_labels.r**
- **`frq`** funciona bien, pero **`sjt.frq`** mal

> **`sjt.frq(to_factor(sj.ll.in, add.non.labelled = T))`**

> **`sjt.frq(to_factor(fill_labels(sj.ll.in)))`** # Rellena los 'value labels' que faltan

> **`sj.ll.com <- fill_labels (sj.ll.in)`** # Idem, pero lo hacemos permanente

- Intercambio entre '**`labelled`**' y vector “normal”:

> **`sj.in <- unlabel (sj.ll.in)`** # de '**`labelled`**' a vector “normal”

> **`acuerdo.ll <- as_labelled (acuerdo)`** # La operación contraria

Agenda

- Por qué este taller: compartir mi trayectoria y mi exploración
- SPSS (y SAS) vs. R para encuestas
- Volcando SPSS (y SAS) a R
- R sin SPSS (ni SAS) manejo de datos con **`{memisc}`** y **`{sjmisc}`**
- Tablas y gráficos básicos con **`{sjPlot}`**
- Otros:
  - casestovars
  - escalas tipo Likert
  - preguntas multi-respuesta
- Algunos análisis con {sjmisc} y {sjPlot}

Formateado de tablas {sjPlot} (1)

- Ver script **RpE11_tablas_basico_sjPlot.r**

> `sjt.frq(sjdf1$interes)` # Tabla de frecuencias por defecto

- El **Summary** puede contener: n.º total, n.º válidos, media, sd, asimetría, curtosis
- Los valores pueden ser **ponderados** con: **`weightBy = "vector_de_pesos"`**
- Al ser un fichero HTML, se puede formatear con tags **CSS**

Formateado de tablas {sjPlot} (2)

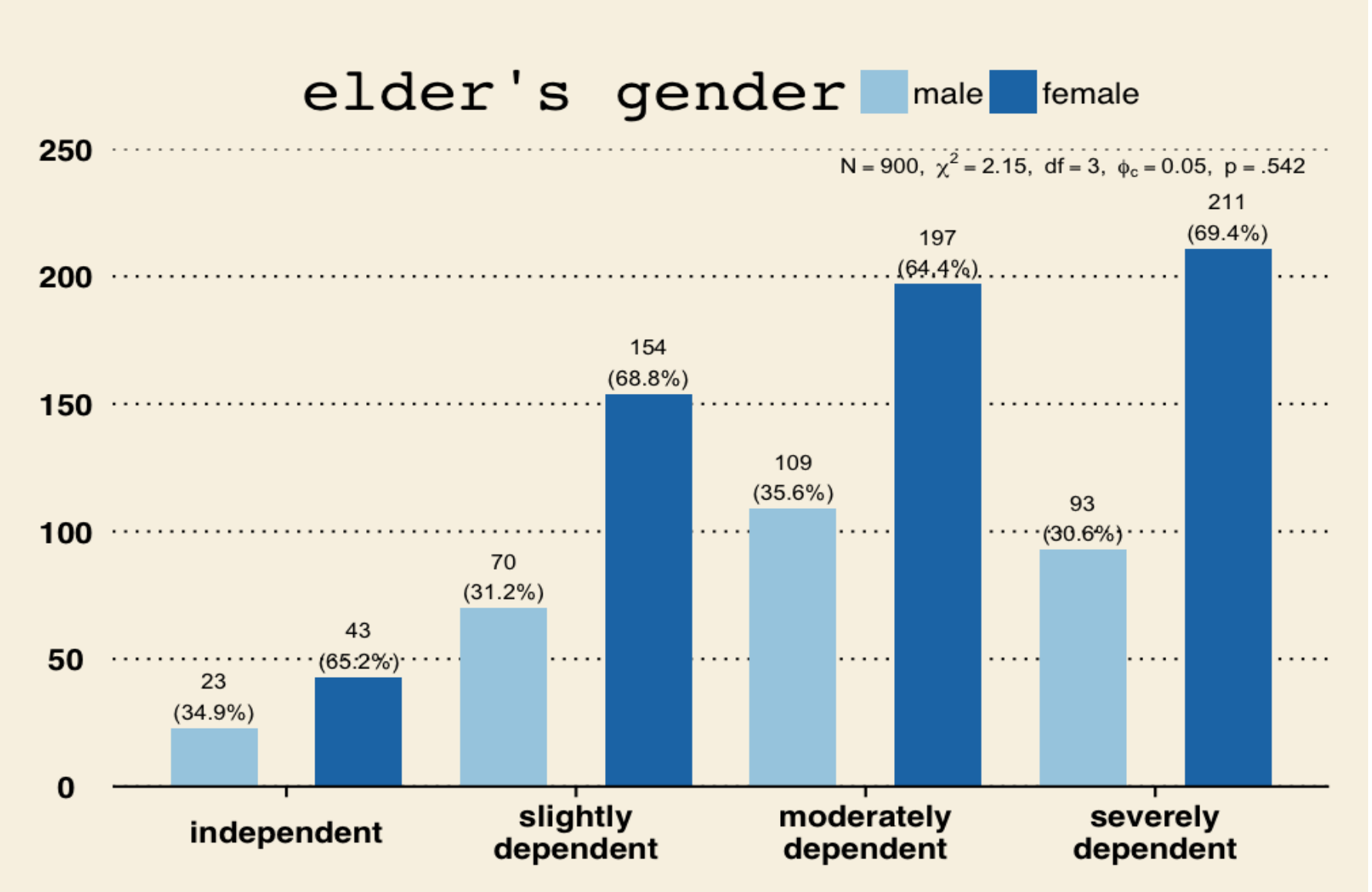
- Formateado básico:

```
> sjt.frq(sjdf1$interes,
      stringValue = "Valor",
      stringPerc = "Porcentaje",
      stringValidPerc = "Porcentaje valido",
      stringMissingValue = "Perdidos",
      stringCumPerc = "Porcentaje acumulado",
      alternateRowColors = T,
      showSummary = F,
      digits = 1)
```

Gráficos con {sjPlot}

- Utiliza el paquete **`{ggplot2}`**
- Ofrece un amplio abanico de opciones de formateo
- Cada función **`sjp.xxx()`** devuelve un objeto **`ggplot`**, que puede posteriormente ser procesado
- También puede utilizar **`{ggthemes}`** **y** **`{ggthemr}`**
- Permite crear un **theme** propio con **`sjp.setTheme()`**

Gráficos con {sjPlot}: ejemplo



Agenda

- Por qué este taller: compartir mi trayectoria y mi exploración
- SPSS (y SAS) vs. R para encuestas
- Volcando SPSS (y SAS) a R
- R sin SPSS (ni SAS) manejo de datos con **`{memisc}`** y **`{sjmisc}`**
- Tablas y gráficos básicos con **`{sjPlot}`**
- Otros:
  - casestovars
  - escalas tipo Likert
  - preguntas multi-respuesta
- Algunos análisis con {sjmisc} y {sjPlot}

Otros: casestovars

- Acumular casos en una unidad superior: por ej. todos los miembros de un mismo hogar
- Ver script **RpE13_casestovars.r**
- Utilizamos la función **'getanID'** de **{splitstackshape}** y la función **'reshape'** de **{stats}**

```
> library("splitstackshape")
> reshape (getanID (mis.datos, 1), direction = "wide", idvar = names (mis.datos)[1], timevar = ".id")
```

Otros: escalas tipo Likert (1)

- Respuestas tipo "Total acuerdo", "Acuerdo", "Desacuerdo", "Total desacuerdo". Ver script **RpE14_likert.r**
- 1º caso: tabla con **`{sjmisc}`** y **`{sjPlot}`** y n.º par de valores. **`sjt.stackfrq`** sólo funciona con un número par de valores

```
> sjt.stackfrq(miLikert1[, c(1,3,4)], valuelabels = value.labels)
```

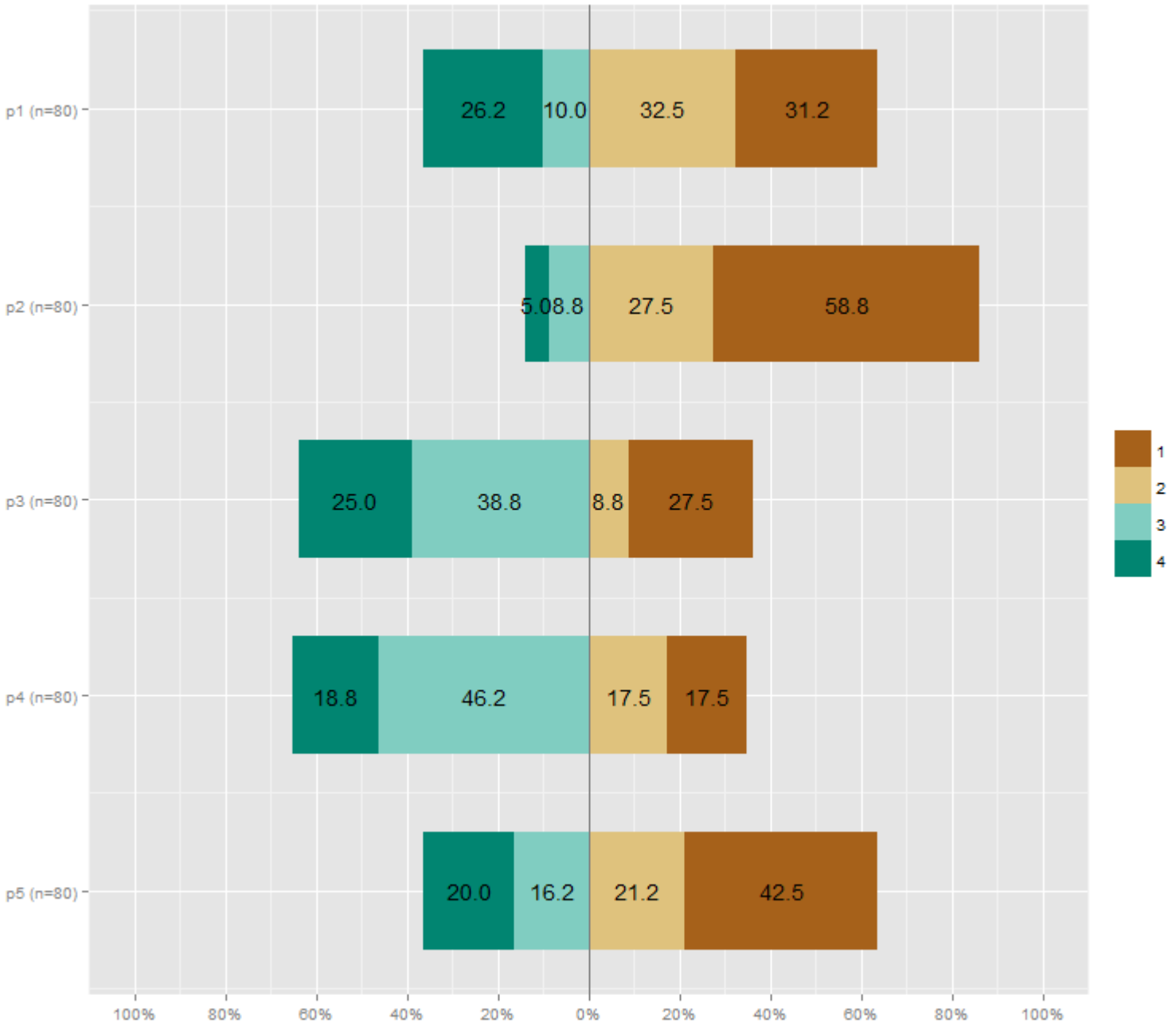
	Total acuerdo	Acuerdo	Desacuerdo	Total desacuerdo
Pregunta 1	30.00 %	31.25 %	11.25 %	27.50 %
Pregunta 3	21.25 %	8.75 %	37.50 %	32.50 %
Pregunta 4	13.75 %	31.25 %	43.75 %	11.25 %

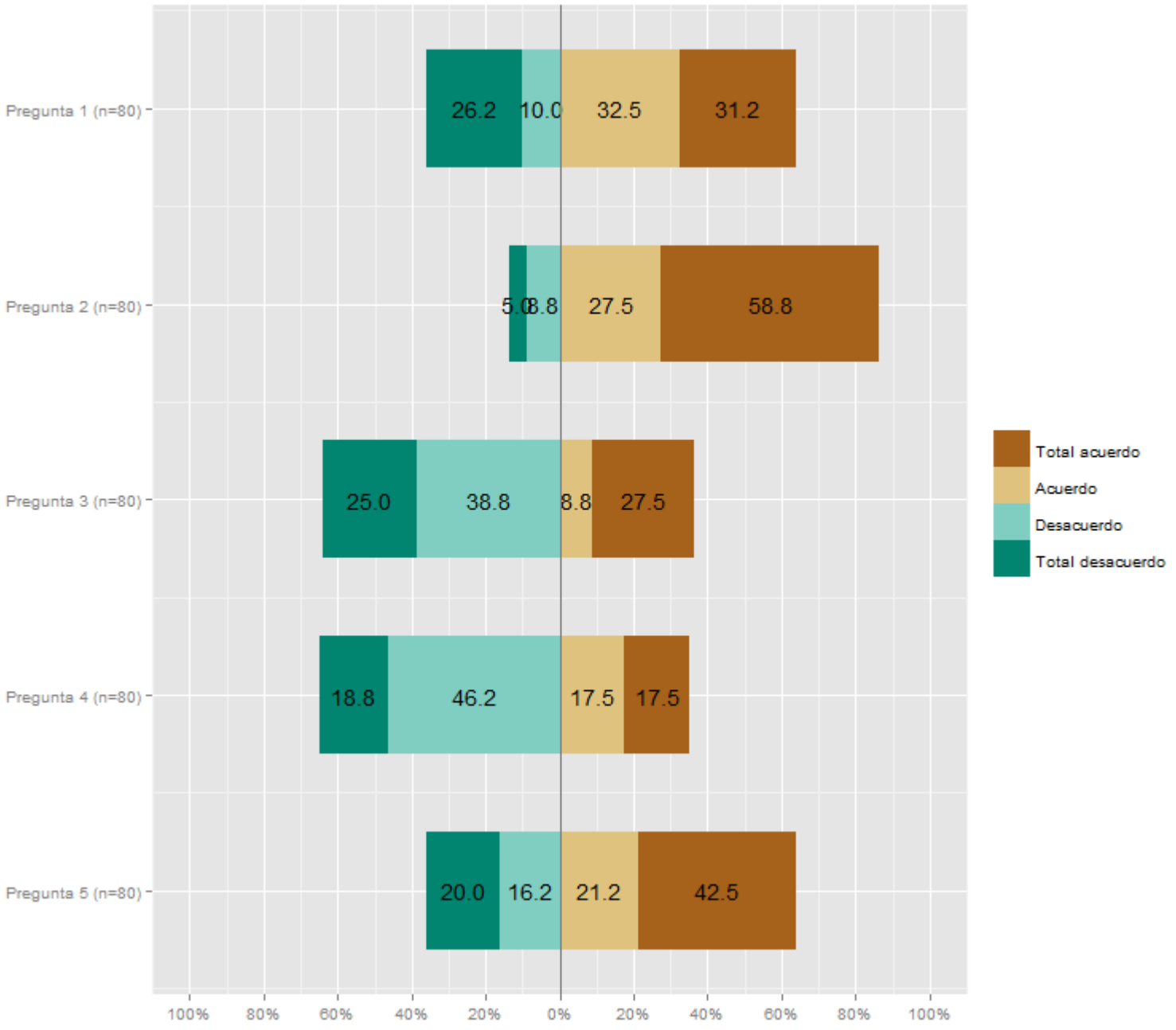
Existen múltiples opciones de configuración

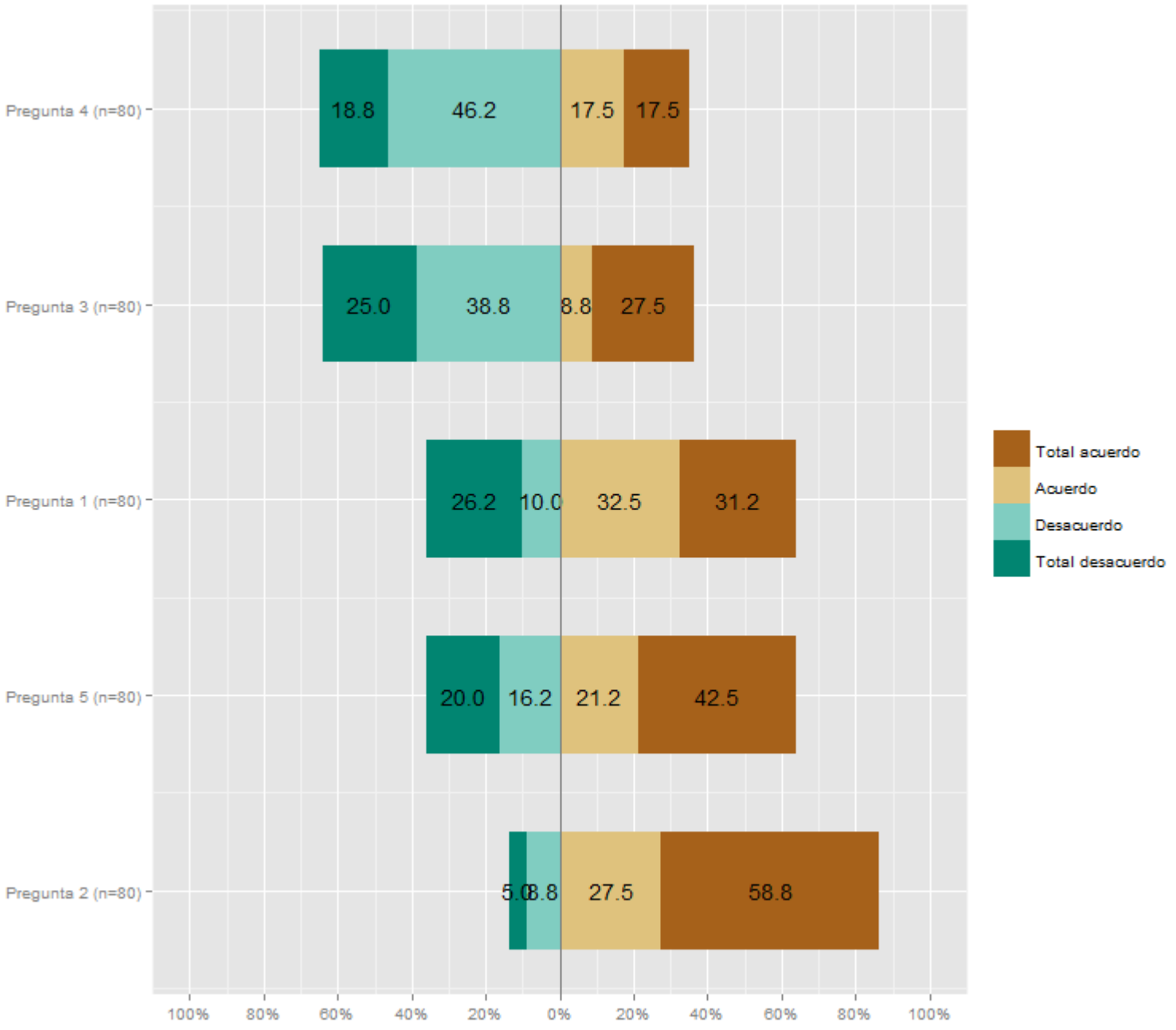
Otros: escalas tipo Likert (2)

- Ver script **RpE14_likert.r** líneas 31ss
- 2º caso: gráfico con **{sjmisc}** y **{sjPlot}** y n.º par de valores. **sjp.likert** sólo funciona con un número par de valores. Para valores impares, ver más adelante

```
> sjp.likert(miLikert) # Sin 'label(s)'
> sjp.likert(miLikert, axisLabels.y = variable.labels, legendLabels = value.labels)# Añadimos los 'variable' y 'value' labels
> sjp.likert(miLikert, axisLabels.y = variable.labels, legendLabels = value.labels, sort.frq = "pos.asc")# Se pueden ordenar las preguntas según los valores positivos o negativos en orden ascendente o descendente
```


p1 (n=80)
p2 (n=80)
p3 (n=80)
p4 (n=80)
p5 (n=80)
26.2
10.0
32.5
31.2
5.0
8.8
27.5
58.8
25.0
38.8
8.8
27.5
18.8
46.2
17.5
17.5
20.0
16.2
21.2
42.5
100%
80%
60%
40%
20%
0%
20%
40%
60%
80%
100%
1
2
3
4

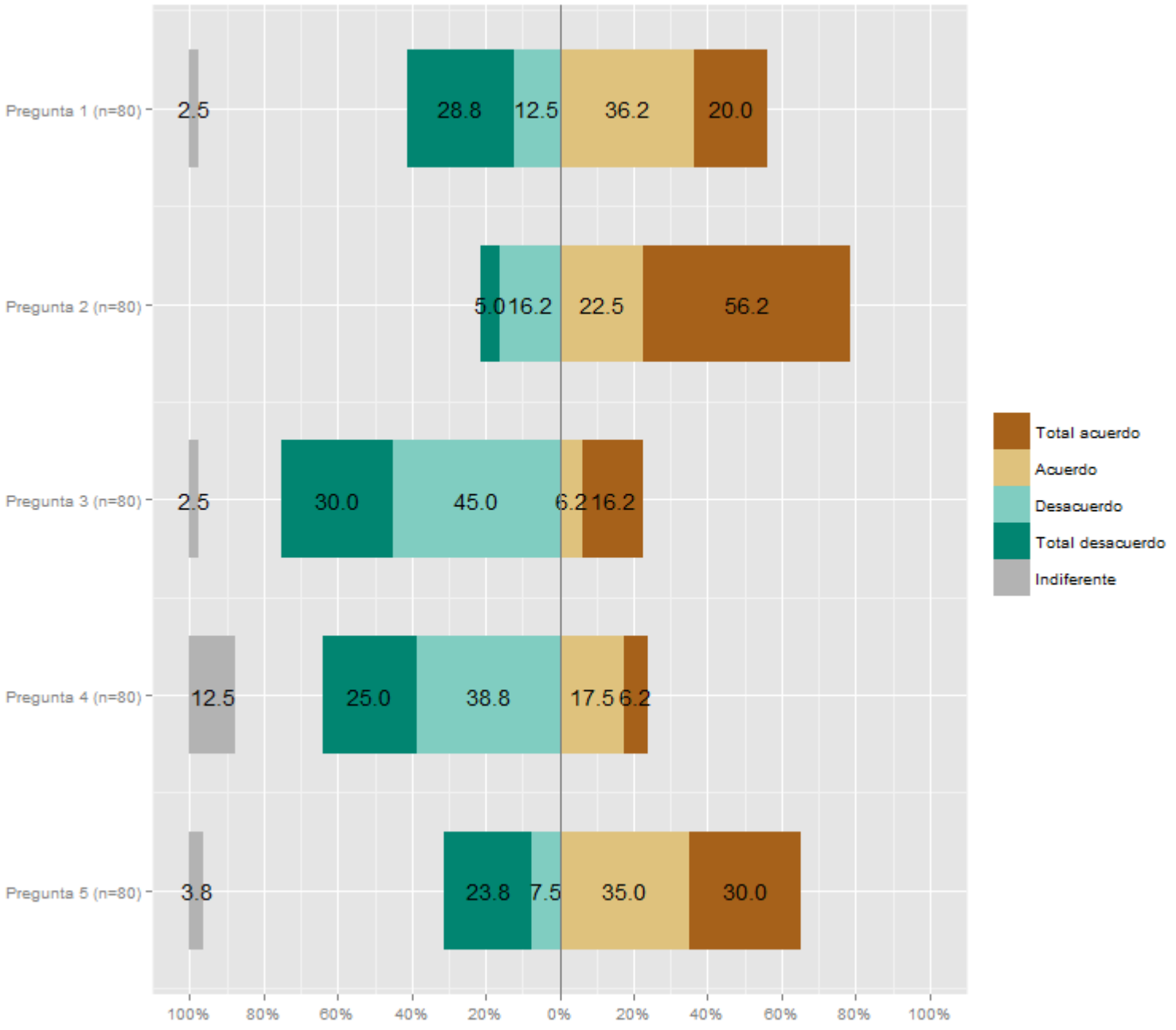

Pregunta 1 (n=80)
Pregunta 2 (n=80)
Pregunta 3 (n=80)
Pregunta 4 (n=80)
Pregunta 5 (n=80)
26.2
10.0
32.5
31.2
5.0
8.8
27.5
58.8
25.0
38.8
8.8
27.5
18.8
46.2
17.5
17.5
20.0
16.2
21.2
42.5
100%
80%
60%
40%
20%
0%
20%
40%
60%
80%
100%
Total acuerdo
Acuerdo
Desacuerdo
Total desacuerdo


Pregunta 4 (n=80)
Pregunta 3 (n=80)
Pregunta 1 (n=80)
Pregunta 5 (n=80)
Pregunta 2 (n=80)
18.8
46.2
17.5
17.5
25.0
38.8
8.8
27.5
26.2
10.0
32.5
31.2
20.0
16.2
21.2
42.5
5.0
8.8
27.5
58.8
100%
80%
60%
40%
20%
0%
20%
40%
60%
80%
100%
Total acuerdo
Acuerdo
Desacuerdo
Total desacuerdo

Otros: escalas tipo Likert (6)

- 2º caso: con un nº impar de respuestas: una neutra o indiferente
- Ver script **RpE14_likert.r** líneas 52ss
- El valor "neutro" tiene que ser el último valor (¿?) y aparece en el lado izquierdo del gráfico:

```
> sjp.likert (miLikert2,
cat.neutral = 5,
legendLabels = value.labels)
```


Pregunta 1 (n=80)
Pregunta 2 (n=80)
Pregunta 3 (n=80)
Pregunta 4 (n=80)
Pregunta 5 (n=80)
2.5
28.8
12.5
36.2
20.0
5.0
16.2
22.5
56.2
2.5
30.0
45.0
6.2
16.2
12.5
25.0
38.8
17.5
6.2
3.8
23.8
7.5
35.0
30.0
100%
80%
60%
40%
20%
0%
20%
40%
60%
80%
100%
Total acuerdo
Acuerdo
Desacuerdo
Total desacuerdo
Indiferente

Otros: Likert con {likert} (8)

Funciona perfectamente con un nº impar de respuestas, la neutra en el punto central

Ver script **RpE15_likert2.r**

```
> library ("likert")
> library ("reshape")
```

Las variables son factores, de la misma longitud y **"levels"**

Creamos el objeto likert:

```
> likert5 <- likert(otroLikert)
> summary (likert5)
  > plot(likert5)
```

Invertimos los "levels":

```
> otroLikert.reverse <- reverse.levels(otroLikert)
```

Agenda

- Por qué este taller: compartir mi trayectoria y mi exploración
- SPSS (y SAS) vs. R para encuestas
- Volcando SPSS (y SAS) a R
- R sin SPSS (ni SAS) manejo de datos con **`{memisc}`** y **`{sjmisc}`**
- Tablas y gráficos básicos con **`{sjPlot}`**
- Otros:
  - casestovars
  - escalas tipo Likert
  - preguntas multi-respuesta
- Algunos análisis con {sjmisc} y {sjPlot}

Algunos análisis con {sjmisc} y {sjPlot}

Tablas y gráficos: ver script **RpE16_analisis_sjmisc-sjPlot**

- Tablas de contingencia (líneas 15-36)
- Correlaciones (líneas 38-67)
- Regresión lineal (lm), comparación de modelos y modelos lineales generalizados (glm) (líneas 69-86)
- Modelos lineales de efectos mixtos (Linear Mixed Effects Models) (líneas 88-118)
- Análisis de componentes principales (Principal Components Analysis, PCA) (líneas 120-131)
- Análisis de items de una escala o índice (líneas 133-134)
- Gráfico ANOVA de un factor (líneas 136-141)

En fin… ¿qué queda por hacer?

Temas menores:

- Preguntas multi-respuesta
- Post-estratificación

Temas mayores:

- "Parseado" de **`item{memisc}`** a **`labelled{sjmisc}`**
- "Parseado" de código SPSS (.sps) y SAS (.sas) a objetos R (**`item`**, **`labelled`**, etc.)

Autor

- José Ignacio Casas Álvarez
- Sociólogo: +30 años de experiencia
- Socio-Fundador de Jomial Research & Consultants SL www.jomialresearch.com
    - Desk Research (> Internet)
    - Análisis de datos
- Miembro de la Foundation for Open Access Statistics
- Miembro de la Comunidad R Hispano
- Miembro de AEDEMO
- e-mail: ji.casas@jomialresearch.com
- LinkedIn: http://es.linkedin.com/pub/jose-ignacio-casas-alvarez/7/495/71b

Muchas gracias por su atención

e-Mail para post-consultas:
rparaencuestas@jomialresearch.com