FactoMineR en tiempos modernos

José Luis Cañadas Reche

Noviembre 2018

Yo estuve allí !!

Figure 1: imagen

Antonio también !!

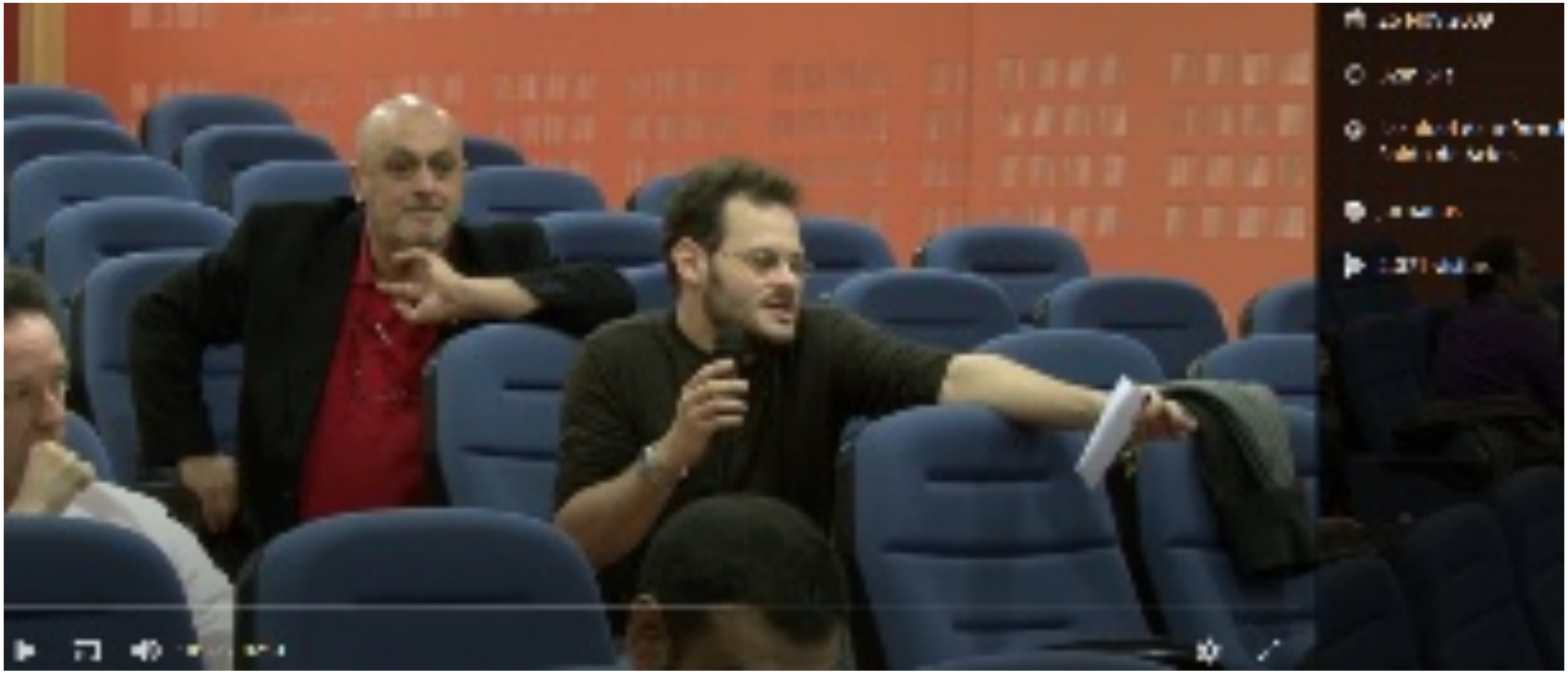


Figure 2: antonio

Qué usamos

- `sparklyr` hace fácil el big data
- No siempre necesitamos todas las filas. Usar frecuencias
- Análisis geométrico de datos `FactoMineR` y `factoextra` nos ayudan a extraer información útil.
- Centrarse en las variables importantes

Ejemplo

¿La *satisfacción* de nuestros clientes está relacionada con dónde viven, qué edad tienen o de dónde vienen ? Consideramos 4 variables

- Satisfacción cliente: Nex promotor Score. 3 categorías: Detractor, Neutro, Promotor
- Provincias: 52 categorías más NA's
- Entrada. Nuevas entradas, portabilidades desde otros operadores
- Edad del cliente. Grupos de edad

Enfoques

- Análisis de correspondencias múltiples. Igual importancia de todas las variables
- Análisis de correspondencias simples. Filas: NPS, Columnas: Resto de variables

Spark

```
library(sparklyr)
library(tidyverse)
library(FactoMineR)
library(factoextra)
library(cowplot)
```

```
sc <- spark_connect(master = "local")
```

Spark

Ejemplo de datos

```
## # Source: spark<?> [?? x 6]
##   nps_cat  prov     tiene_alguna_porta  edad entrada
## * <chr>    <chr>                 <int> <int> <chr>
## 1 neutro   Asturias                  1    40 Porta_Otros
## 2 detract  Alicante                  1    57 Porta_Comp
## 3 detract  Murcia                    1    44 Porta_Comp
## 4 promotor Madrid                    1    55 Porta_Comp
```

Dplyr y spark

```
datos_tbl <- datos_tbl  %>%
    mutate(
        tiene_incidencia = ifelse(is.na(tiene_incidencia),
        tiene_alguna_porta = ifelse(is.na(tiene_alguna_port
        entrada = ifelse(
            is.na(entrada) |
                entrada == "" | entrada == "NA" ,
            "Sin_porta",
            entrada
        ),
        prov = ifelse(is.na(prov) |
                          prov == "" |
                          prov == "NA", "Prov_desconocida",
        edad = case_when(
            edad < 0 ~ "Edad_desconocida",
            edad < 35 ~ "18-35",
            edad < 45 ~ "35-44",
            edad < 55 ~ "45-54",
```

Datos agrupados

```
dat_agrup <-
    datos_tbl %>% group_by(nps_cat, prov, entrada, edad) %>

dat_agrup
```

```
## # A tibble: 10 x 5
##    nps_cat  prov              entrada              edad
##  * <chr>    <chr>             <chr>                <chr>
##  1 promotor Prov_desconocida  Entrada_desconocida  Edad_de
##  2 neutro   Prov_desconocida  Porta_Comp_B         45-54
##  3 neutro   Málaga            Porta_Otros          35-44
##  4 promotor Segovia           Porta_N              55-64
##  5 neutro   Valencia          Porta_M              45-54
##  6 neutro   Madrid            Porta_Comp_B         45-54
##  7 promotor Murcia            Entrada_desconocida  Edad_de
##  8 promotor Prov_desconocida  Porta_Comp_B         55-64
##  9 promotor Prov_desconocida  Porta_N              45-54
## 10 promotor Madrid            Entrada_desconocida  Edad_de
```

MCA con FactoMiner y factoextra

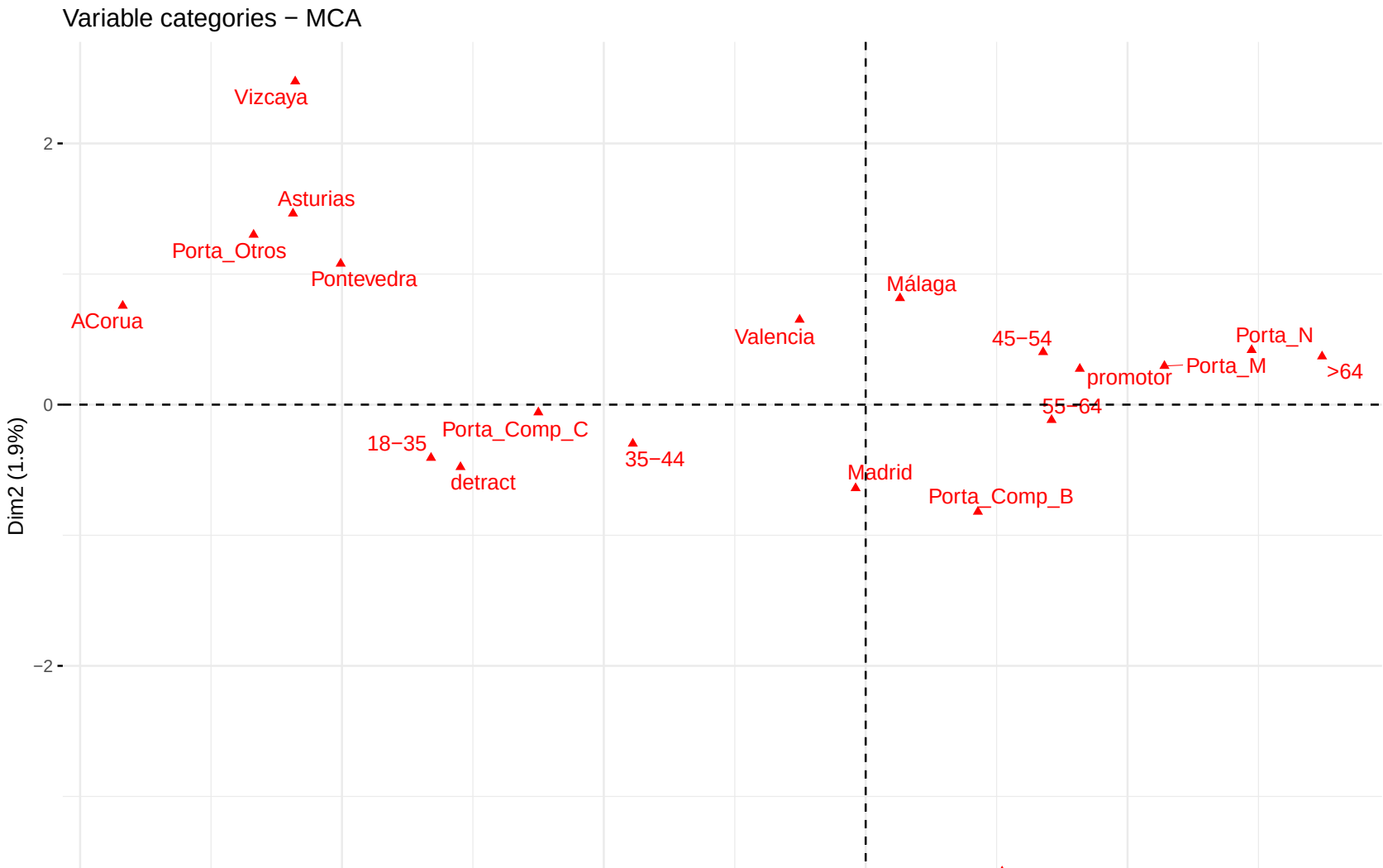
La tabla con las frecuencias es suficientemente pequeña para tratarla en local. Con `FactoMineR` podemos usar las frecuencias como ponderación de las filas.

```
dat_mca <- dat_agrup %>% collect() # traemos a local

dat_mca <- dat_mca %>%
    filter(prov != "Prov_desconocida", edad!="Edad_desconoc

res_mca <- MCA(dat_mca[1:4], ncp = 100, row.w = dat_mca$n,
```

MCA con FactoMiner y factoextra

- ¿Relación entre Porta_Comp_C , 18-35 y detract?
- Poca variabilidad explicada por las dos primeras dimensiones

```
fviz_mca_var(res_mca, repel=TRUE, select.var = list(contrib
```



Correspondencias simple

En vez de considerar la tabla multivía construimos una tabla a 2 vías

- Filas: Categorías de la variable de satisfacción, nps_cat
- Columnas: Categorías del resto de las variables
- Valores: Porcentajes por columna.

De esta forma estandarizamos

Correspondencias simple

Construimos las tablas de contingencia que necesitamos en spark, usando la función `sdf_pivot`

```
tabla_edad <- datos_tbl %>%
    select(nps_cat, edad, prov) %>%
    sdf_pivot(nps_cat ~ edad ,
              fun.aggregate = list(edad = "count"))

tabla_edad <- collect(tabla_edad)

tabla_edad
```

```
## # A tibble: 3 x 7
##   nps_cat  `18-35` `35-44` `45-54` `55-64` `>64` Edad_de
##   <chr>      <dbl>   <dbl>   <dbl>   <dbl> <dbl>
## 1 neutro       796    1561    1704     855   391
## 2 detract     1179    1797    1685     813   362
## 3 promotor    1654    2865    3444    1914  1153
```

Correspondencias simple

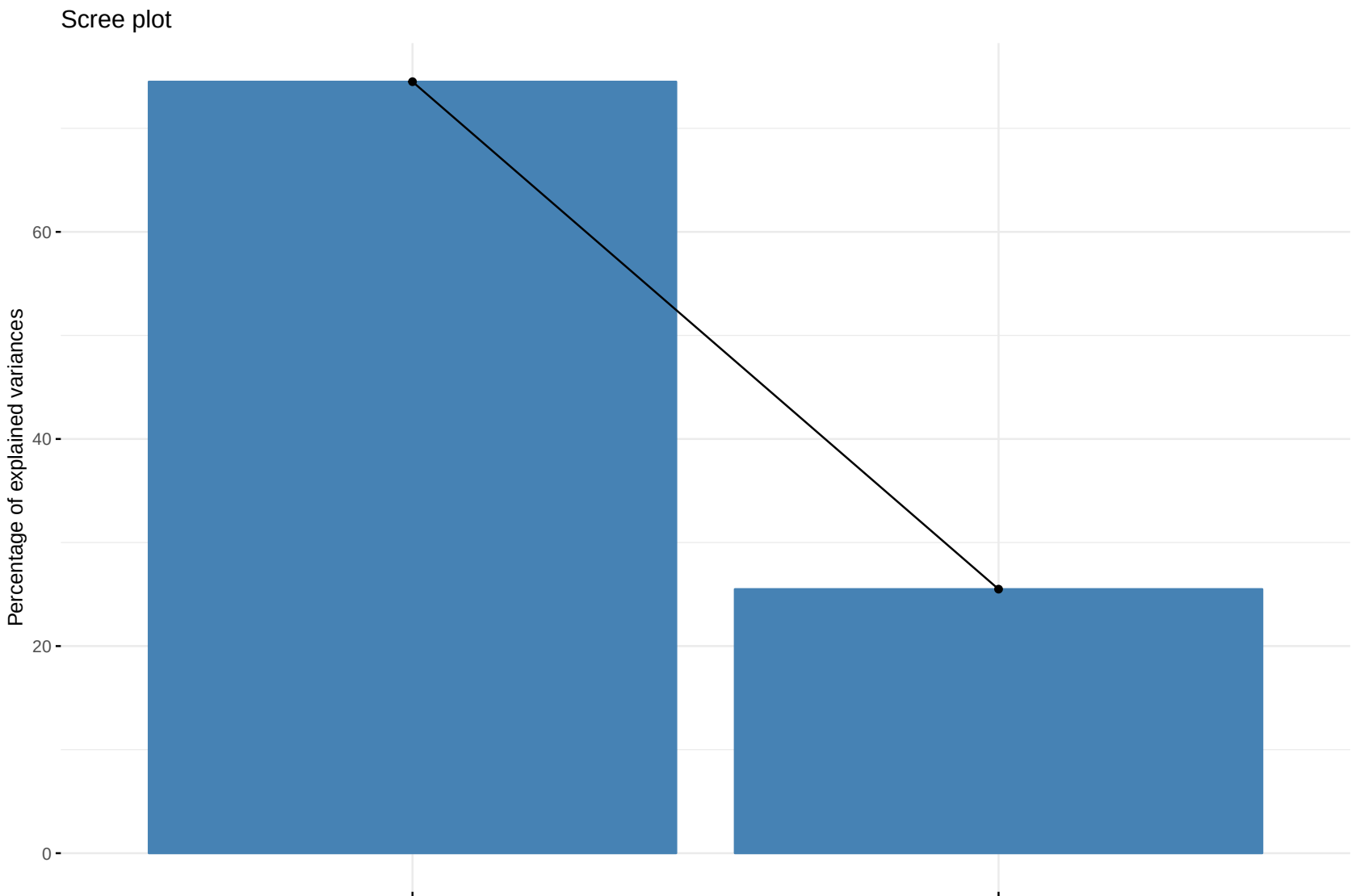
- Todos los datos de la tabla en la misma escala. Porcentajes por columnas
- Podemos aplicar análisis de correspondencias

```
##                          neutro   detract promotor
## 18-35                  21.93442 32.488289 45.57729
## 35-44                  25.08436 28.876748 46.03889
## 45-54                  24.93780 24.659739 50.40246
## 55-64                  23.86935 22.696817 53.43384
## >64                    20.51417 18.992655 60.49318
## Edad_desconocida       24.78287 32.338308 42.87883
## ACorua                 24.27984 27.160494 48.55967
## Albacete               22.88136 26.271186 50.84746
## Alicante               22.30889 26.833073 50.85803
## Almería                26.08696 28.804348 45.10870
## Asturias               22.22222 24.494949 53.28283
## Badajoz                26.29969 26.605505 47.09480
## Barcelona              26.45337 24.307452 49.23917
## Burgos                 22.88136 26.271186 50.84746
```

Correspondencias simple

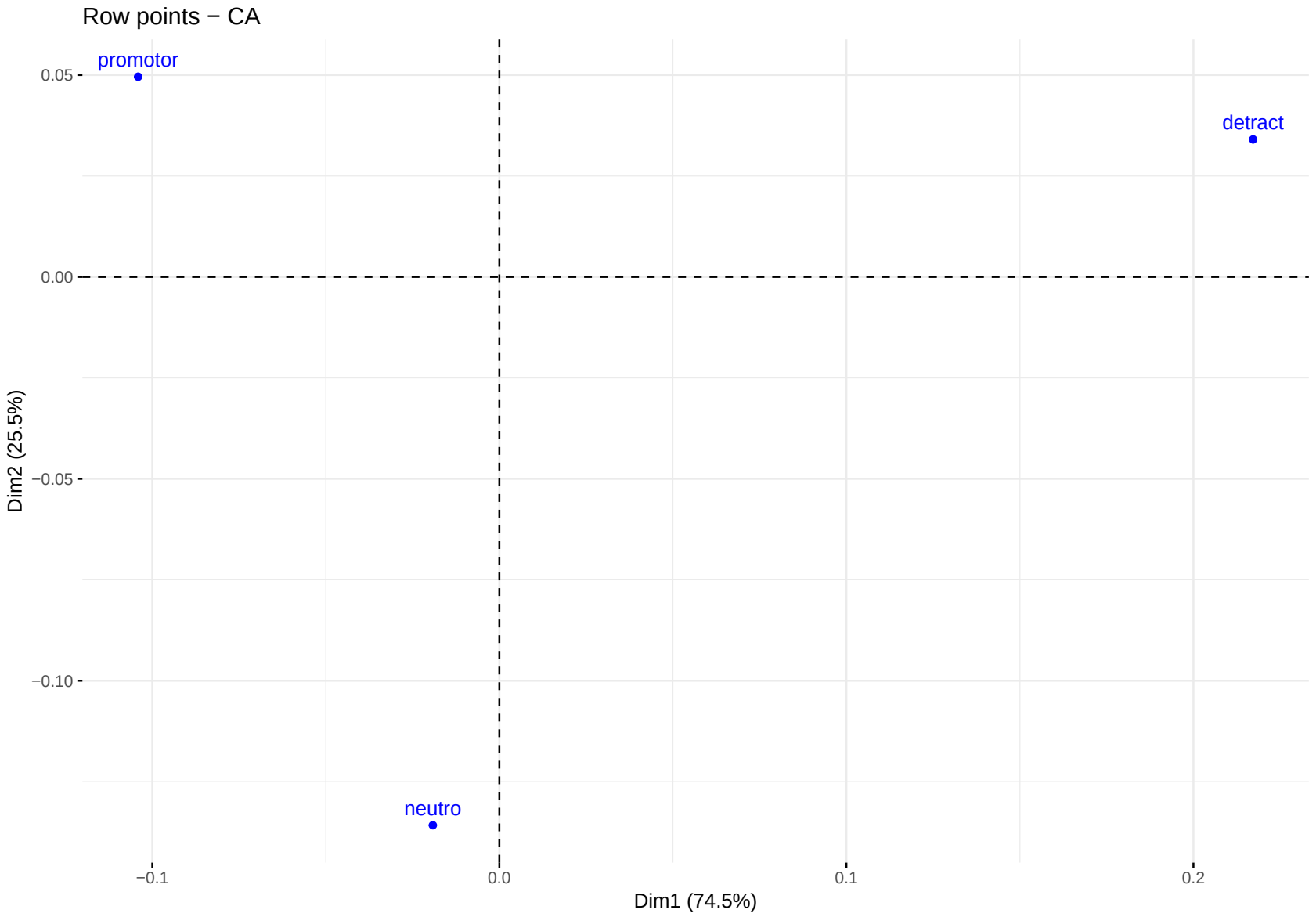
```
res_ca <- CA(mat_final, ncp = 10, graph = FALSE)
```

```
fviz_screeplot(res_ca)
```



Coordenadas filas

```
fviz_ca_row(res_ca)
```



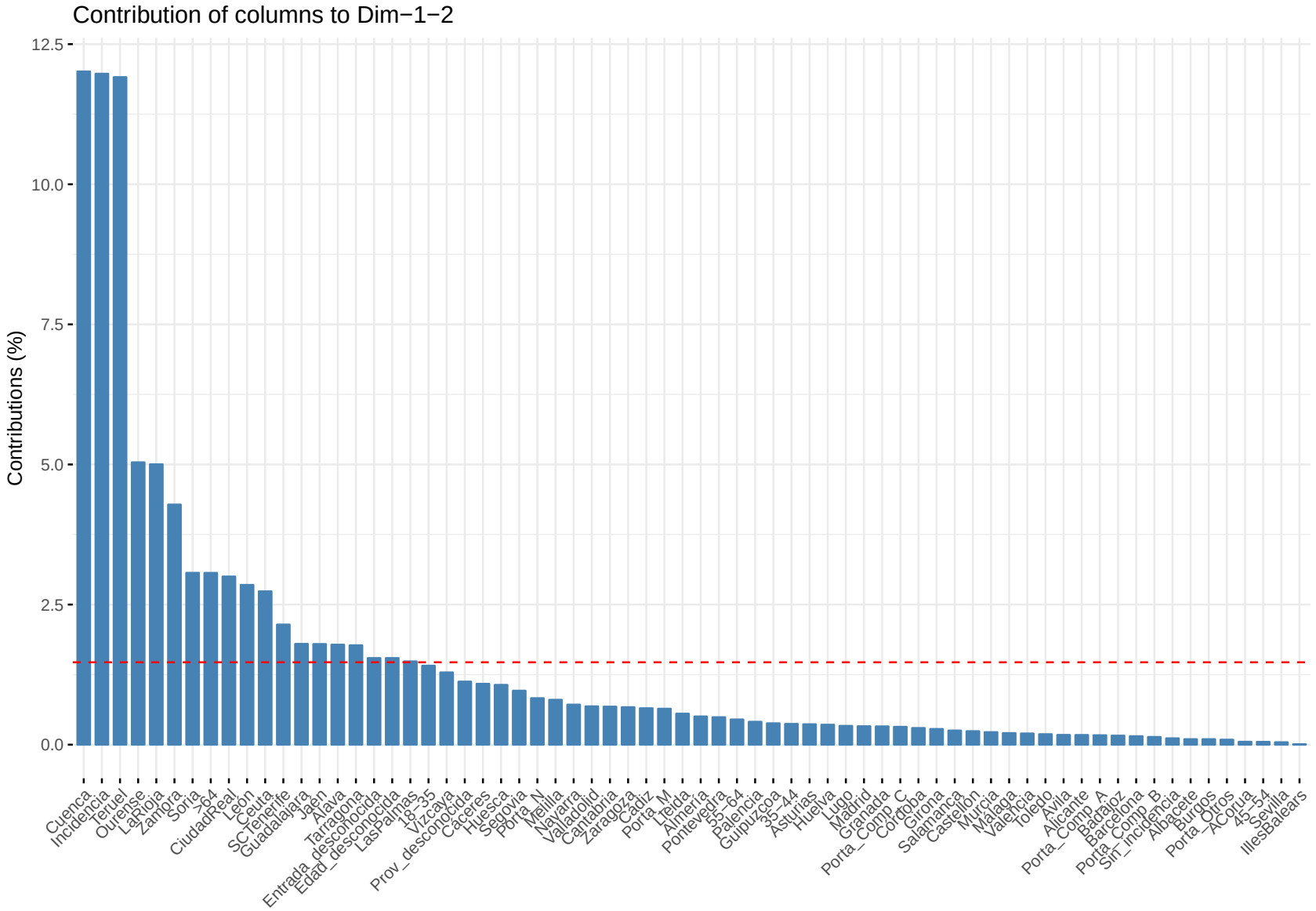
Coordenadas columnas

Interpretación

- Las dimensiones hay que interpretarlas en base a la contribución
- Hay relación entre ser “detractor” y haber tenido “incidencia”. Obvio
- Se detectan provincias dónde el % de detractores es mayor que en otras, así como provincias dónde hay más % de neutros.

Contribución de columnas

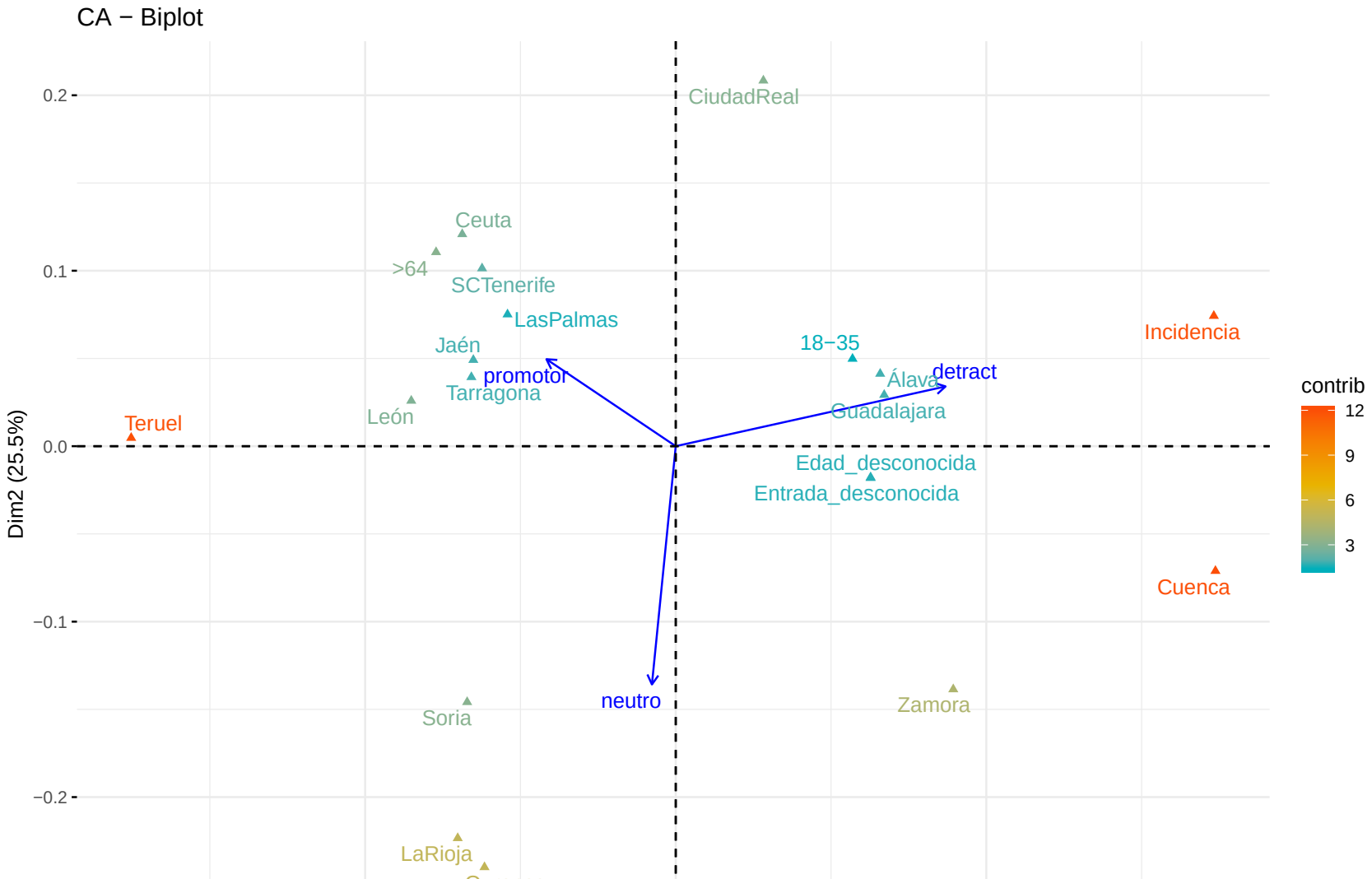
```
fviz_contrib(res_ca , choice = "col", axes = c(1,2))
```



Biplots

- El biplot por defecto “mapa simétrico” no permite interpretar las distancias entre filas y columnas, lo que se debe interpretar es el ángulo.
- A menor ángulo mayor relación,

```
fviz_ca_biplot(res_ca, repel = TRUE,
               select.col = list(contrib = 20 ),
               col.col = "contrib",
               gradient.cols = c("#00AFBB", "#E7B800", "#FC
               arrows = c(TRUE, FALSE))
```



```
fviz_ca_biplot(res_ca, repel = TRUE,
               select.col = list(name = c("18-35","35-44","
                                          "Incidencia",
                                          "Sin_incidencia" ,"Porta_Comp_A","Po
               arrows = c(TRUE, FALSE))
```