

Práctica 2: Limpieza y validación de los datos

Erika Bracamonte Anthony Alarcon

5 de Enero, 2021

Contents

Descripción del dataset	1
Variables Numéricas:	3
Variables Categóricas:	3
Integración y selección de los datos	3
Limpieza de los datos	6
Normalización	9
Gestión de valores nulos y outliers	13
Discretización	14
Reducción de dimensionalidad	14
Análisis de los datos	17
Normalidad y homocedasticidad	17
Regresión	20
Correlaciones	21
Clustering k-means	22
Clustering JERÁRQUICO	37
Clustering de Partición entorno a Centroides (PAM)	53
Resultados	66
Análisis visual del conjunto de datos	66
Resolución del problema	73

Descripción del dataset

En los últimos 10 años, la guerra de ofertas en descuentos, afiliación, incentivo de consumo, etc., ha crecido en el sector bancario, siendo hoy muy común que a cada cliente le llegue un sinfín de ofertas por diversos canales (correo, sms, etc.). Las ofertas que pueda promocionar una entidad bancaria es consecuencia de una negociación con los establecimientos o empresas, en la que la entidad bancaria paga el descuento u oferta que el cliente. Este gasto que realiza la entidad, muchas veces no resulta óptimo, porque el cliente no llega a consumir la oferta.

El objetivo que nos planteamos es conocer el perfil de consumo del cliente, de tal manera que las ofertas se asignen de manera adecuada. Consiguiéndolo, optimizaríamos el gasto realizado por la entidad bancaria, y conseguiríamos ubicar soluciones a las necesidades de los clientes, de manera menos azarosa.

La base de datos está a nivel de transacciones por cliente y mes. La transformaremos a nivel de clientes, evaluando el mejor indicador para segmentar.

```

# En caso no tener instalado algún paquete, lo importamos con install.packages("nombre_paquete")
library(funModeling)
library(dplyr)
library(tidyr)
library(ggplot2)
library(clusterSim)

DataConsumo = read.csv(file="BaseConsumo.csv", header=TRUE, sep=";", na.strings=c("", "NA"))
head(DataConsumo)

##   codmes cliente     edad     ingreso sexo flgLimaProv flgAfBxi grupoGiro
## 1 201611    32572 [1. <=25] [2. <=1500]   F          1          1  restbar
## 2 201611    29843 [1. <=25] [3. <=2500]   F          0          1 prodlocal
## 3 201611    30033 [2. <=30] [2. <=1500]   F          0          1  restbar
## 4 201612     3847 [7. > 55] [3. <=2500]   M          1          0  restbar
## 5 201611    33131 [1. <=25] [2. <=1500]   F          1          1  restbar
## 6 201611    31433 [2. <=30] [3. <=2500]   M          1          1 prodlocal
##   monto trx
## 1     12  1
## 2     22  1
## 3     10  1
## 4     350  4
## 5     65  2
## 6     16  1

```

Haciendo una inspección a los datos, tenemos que:

- Son 40 mil clientes
- Son transacciones de clientes durante 12 meses
- Son 21 rubros de consumo:
 1. Prodsuper: grupo supermercados
 2. Restbar: grupo restaurantes y bares
 3. Salud: grupo farmacia, clínicas, seguro salud, masaje, etc.
 4. Vehrep: grupo repuestos, compra vehículo, gasolina, etc.
 5. Entretenimiento: grupo entrenamiento, guía turística.
 6. Tiendadepar: grupo tienda por departamento.
 7. Ropamoda: grupo tienda de ropa, tienda independiente de mall.
 8. Prodpersondiv: grupo diversos productos personales, joyería, etc.
 9. Telcom: grupo telecomunicaciones, pago celular, recargas, etc.
 10. Financiero: grupo financiero, pago de impuestos, seguros, etc.
 11. Transplaerea: grupo transporte, aerolíneas, bus interprovincial, etc.

12. Clubmkt: grupo clubes.
13. Prodlocal: grupo productos locales.
14. Enseñanza: grupo enseñanza, pago universidades, academia, etc.
15. Belleza: grupo belleza, gimansio, maquillaje, etc.
16. Prodelectro: grupo producto electrónico.
17. Alqbiens: grupo alquiler de bienes, hoteles, pago de departamento, etc.
18. Artcultura: grupo arte y cultura, teatro, librería, etc.
19. Profdiverso: grupo profesional, servicio legal, consultoría, etc.
20. Hogaroficina: grupo hogar y oficina, utensilios para la oficina u hogar.
21. Informática: grupo informática, software, reparación Pc, etc.

Variables Numéricas:

flgLimaProv: Esta variable es una dummy que nos indica si la transacción fue hecha en Lima (1) o Provincia (0).

flgAfBxi: Esta variable es una dummy que indica si la transacción fue por Banca por internet.
monto: El monto consumido.

trx: La cantidad de transacciones. Recordemos la base de datos está a nivel de cliente y mes.

Variables Categóricas:

codmes: Es el código de mes que es la combinación del año y mes 'yyyymm'.

cliente: Es el id del cliente, va de 1 a 40 000.

edad: Esta variable está categorizada en 7 niveles.

ingreso: Esta variable está categorizada en 8 niveles.

sexo: Esta variable tiene 2 categorías.

grupoGiro: Es el rubro consumido.

Integración y selección de los datos

Convertiremos los valores categóricos que han sido importados como integer: codmes, cliente.

```
# En caso no tener instalado algún paquete, lo importamos con install.packages("nombre_paquete")
df_status(DataConsumo)
```

##	variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
## 1	codmes	0	0.00	0	0.00	0	0	integer	12
## 2	cliente	0	0.00	0	0.00	0	0	integer	40000
## 3	edad	0	0.00	12999	3.83	0	0	factor	7
## 4	ingreso	0	0.00	31328	9.23	0	0	factor	8
## 5	sexo	0	0.00	11755	3.46	0	0	factor	2
## 6	flgLimaProv	74619	21.98	11751	3.46	0	0	integer	2
## 7	flgAfBxi	64224	18.92	11751	3.46	0	0	integer	2
## 8	grupoGiro	0	0.00	0	0.00	0	0	factor	22
## 9	monto	347	0.10	0	0.00	0	0	integer	4366
## 10	trx	0	0.00	0	0.00	0	0	integer	57

```
#Convertiremos los valores categóricos que han sido importados como integer: codmes, cliente
DataConsumo$codmes = as.factor(DataConsumo$codmes)
DataConsumo$cliente = as.factor(DataConsumo$cliente)
```

Como el objetivo propuesto es conocer el perfil de consumo de los clientes, necesitamos hacer la segmentación de los clientes. La base de datos, como ya vimos, está a nivel transacciones, por lo que necesitamos generar otra base a nivel de clientes. Para ello, tenemos varios tipos de agregaciones:

1-La cantidad de meses en los que hizo transacciones. Ejemplo: Si presenta transacciones en 201609 y 201610, tendríamos 2 meses en los que consumió. 2-La cantidad de transacciones. 3-De los dos anteriores, podemos obtener la cantidad de transacciones mensuales.

Primero, inspeccionemos la base de datos con estas agregaciones:

Veamos la cantidad de transacciones por rubro:

En el top 5 de rubros donde se tiene más transacciones encontramos en orden jerárquico a los supermercados, bares y restaurantes, salud, vehículos y repuestos, y entretenimiento.

```
# Usamos la librería dplyr
```

```
DataConsumo %>%
  group_by(grupoGiro) %>%
  summarise(Cant_Trx = sum(trx)) %>%
  arrange(desc(Cant_Trx))
```

```
## # A tibble: 22 x 2
##   grupoGiro      Cant_Trx
##   <fct>          <int>
## 1 prodsuper      142313
## 2 restbar        103507
## 3 salud           63109
## 4 vehrep          45978
## 5 entretenimiento 43009
## 6 tiendadepar    37290
## 7 ropamoda        28205
## 8 prodpersondiv  15649
## 9 telcom          12606
## 10 financiero     9889
## # ... with 12 more rows
```

Veamos la cantidad de transacciones por mes:

-El mes de mayor cantidad de transacciones es diciembre, como es de esperarse.

```
# Creamos la agrupación
trx_mes=DataConsumo %>%
  group_by(codmes) %>%
  summarise(Cant_Trx = sum(trx)) %>%
  arrange(codmes)

# Lo convertimos en data frame
trx_mes=data.frame(trx_mes)

# Imprimimos la salida
print(trx_mes)

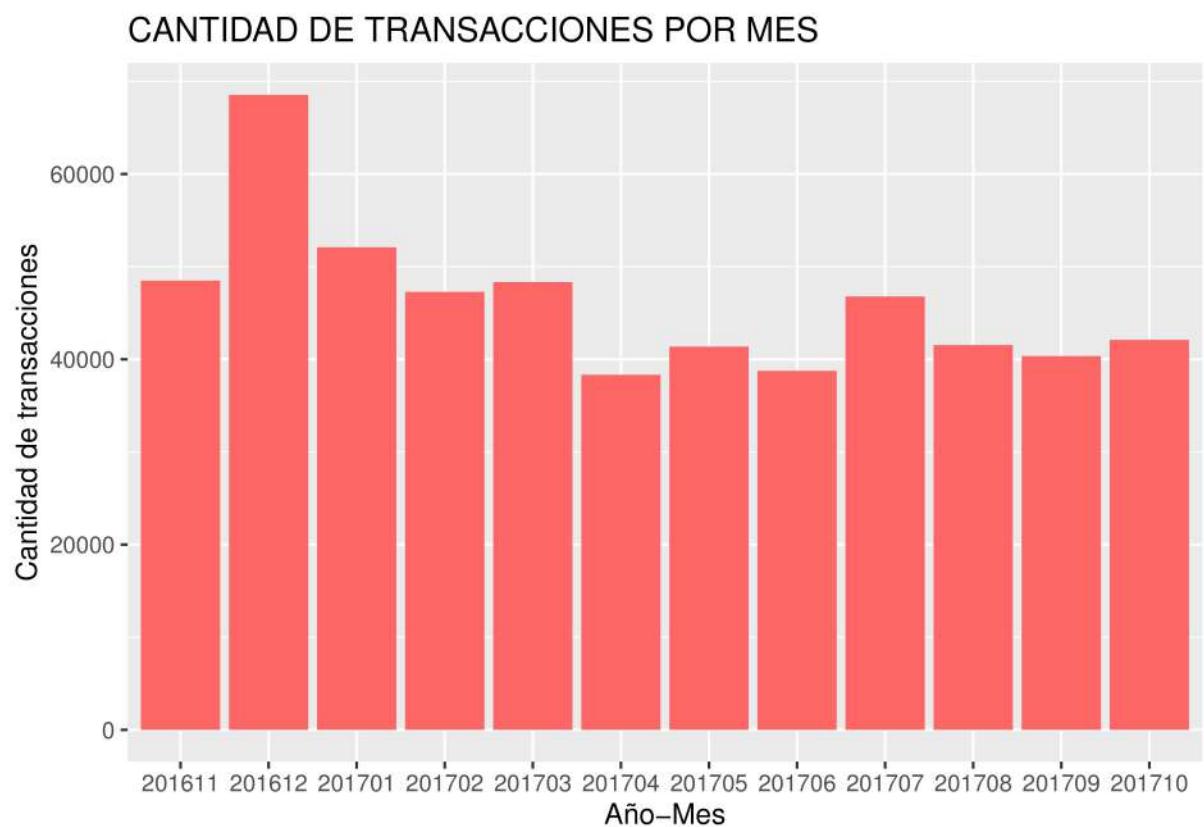
##      codmes Cant_Trx
```

```

## 1 201611 48494
## 2 201612 68542
## 3 201701 52089
## 4 201702 47275
## 5 201703 48331
## 6 201704 38326
## 7 201705 41368
## 8 201706 38751
## 9 201707 46775
## 10 201708 41534
## 11 201709 40342
## 12 201710 42091

# Graficamos
ggplot(trx_mes, aes(x = codmes, y = Cant_Trx)) +
  geom_bar(stat = "identity", fill = "#FF6666")+
  labs(x = 'Año-Mes', y = 'Cantidad de transacciones', title = "CANTIDAD DE TRANSACCIONES POR MES")

```



Analizamos los posibles indicadores a nivel de cliente:

- Cantidad de transacciones
- Cantidad consumida (en soles, moneda de Perú)
- Promedio de transacciones por mes

El promedio de transacciones por mes es el mejor indicador que resume la información de lo que buscamos (segmentos en los que consume, independientemente de los montos porque buscamos que el cliente consuma la oferta).

Limpieza de los datos

Evaluaremos también la data que vamos a analizar, porque si bien el promedio de transacciones mensuales puede ser un buen indicador, existen casos que sesgúen los resultados. Por ejemplo, clientes que solo realizaron transacciones en un solo mes, puede llevarnos a obtener segmentos de clientes que no consumen frecuentemente, por lo que las ofertas que podamos ofrecerle no tendrían el impacto que buscamos sobre el universo de todos nuestros clientes. Lo mismo con la cantidad de transacciones promedio de los clientes que han consumido los 12 meses, puede llevarnos también a caracterizar un segmento con un promedio no característico de todo el universo.

Para ver cuán estable resulta el promedio de transacciones por frecuencia de consumo:

```
# Hallamos la cantidad meses en los que consumió cada cliente (va de 1 mes a 12)
meses_trx=DataConsumo %>%
  group_by(cliente) %>%
  summarise(distinct_visit_ids = n_distinct(codmes))

# Hallamos la cantidad de transacciones por cliente
sum_trx = DataConsumo %>%
  group_by(cliente) %>%
  summarise(sum(trx))

# Unimos las dos bases anteriores para generar una a nivel de cliente
consumo_cliente = merge(x = meses_trx, y = sum_trx, by = "cliente", all.x = TRUE)

# Cambiamos el nombre de las columnas
##### Frecuencia de consumo
colnames(consumo_cliente)[2] = "frec_mes"
##### Cantidad de transacciones
colnames(consumo_cliente)[3] = "cant_trx"

# Ahora agrupamos por frecuencia de consumo
##### Cantidad de clientes por frecuencia de consumo
fm_cli=consumo_cliente %>%
  group_by(frec_mes) %>%
  summarise(cant_cli = n())
##### Cantidad de transacciones por frecuencia de consumo
fm_trx=consumo_cliente %>%
  group_by(frec_mes) %>%
  summarise(cdt_trx = sum(cant_trx))

# Unimos las dos bases anteriores para obtener la cantidad de transacciones y clientes por frecuencia de consumo
frecuencia_mes = merge(x = fm_cli, y = fm_trx, by = "frec_mes", all.x = TRUE)

# Calculamos la cantidad de transacciones promedio en cada frecuencia de consumo. Es decir, el promedio de transacciones que consumieron en 1, 2, 3, ..., 12 meses. Para esto dividimos la cantidad de transacciones entre la frecuencia de consumo
frecuencia_mes$trx_mes = frecuencia_mes$cdt_trx/frecuencia_mes$frec_mes

# Finalmente obtenemos el promedio de transacciones que el cliente realiza, según la frecuencia de consumo
# han consumido 3 meses en el año, en promedio, cada mes han realizado 2.44 transacciones.
frecuencia_mes$prom_trx_cli = frecuencia_mes$trx_mes/frecuencia_mes$cant_cli

# Le damos el formato de data frame
frecuencia_mes=data.frame(frecuencia_mes)
```

```
# Imprimimos la salida
frecuencia_mes
```

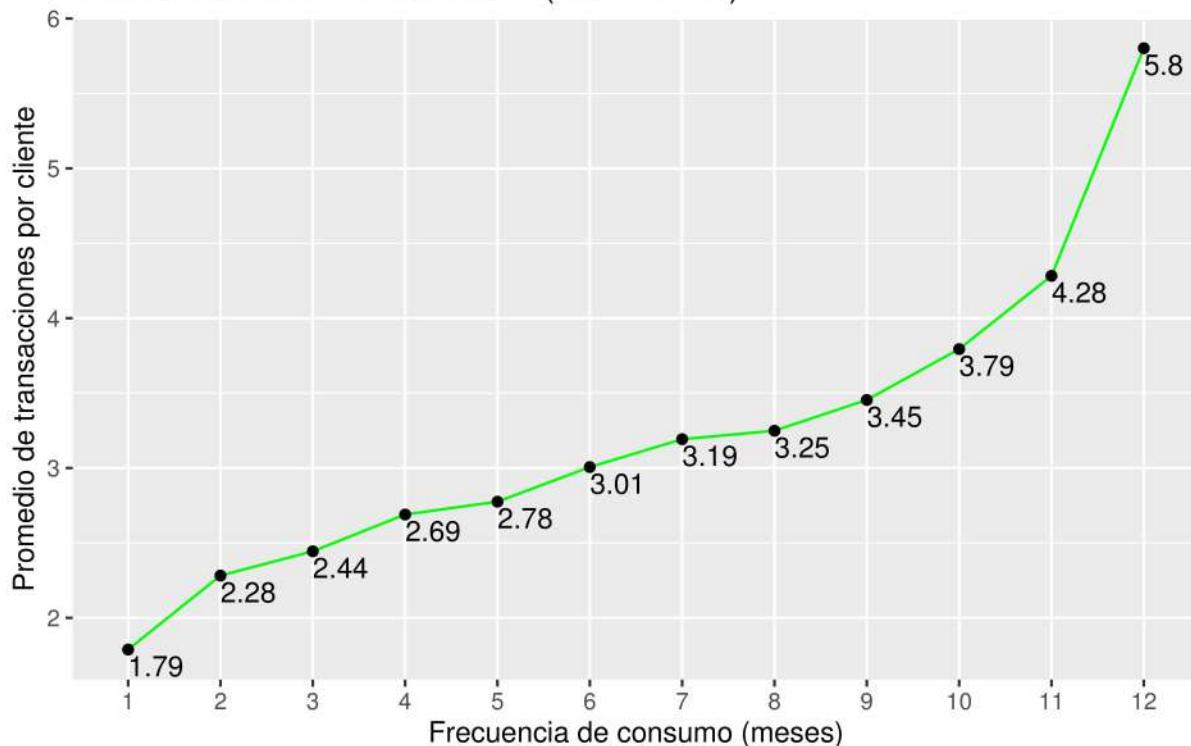
```
##   freq_mes cant_cli cdt_trx  trx_mes prom_trx_cli
## 1      1    11461  20497 20497.000  1.788413
## 2      2     6626  30243 15121.500  2.282146
## 3      3     4578  33569 11189.667  2.444226
## 4      4     3437  36978  9244.500  2.689700
## 5      5     2702  37495  7499.000  2.775352
## 6      6     2176  39250  6541.667  3.006281
## 7      7     1773  39623  5660.429  3.192571
## 8      8     1584  41168  5146.000  3.248737
## 9      9     1310  40729  4525.444  3.454538
## 10     10    1281  48600  4860.000  3.793911
## 11     11    1249  58841  5349.182  4.282772
## 12     12    1823 126925 10577.083  5.802020
```

Con la tabla obtenida, graficamos el indicador de cantidad de transacciones según frecuencia de consumo.

```
# Antes de graficar, convertimos en integer la frecuencia de consumo
frecuencia_mes$frec_mes = as.factor(frecuencia_mes$frec_mes)
```

```
# Graficamos el indicador
ggplot(frecuencia_mes, aes(x=factor(frecuencia_mes$frec_mes), y=frecuencia_mes$prom_trx_cli, group=1)) +
  geom_line(color='green') +
  geom_point() +
  labs(x = 'Frecuencia de consumo (meses)', y = 'Promedio de transacciones por cliente', title = 'C.
FRECUENCIA DE CONSUMO (EN MESES)') +
  geom_text(aes(label=round(frecuencia_mes$prom_trx_cli,2)), hjust=0, vjust=1.3)
```

CANTIDAD DE TRANSACCIONES POR CLIENTE SEGÚN FRECUENCIA DE CONSUMO (EN MESES)



El promedio de transacciones mes tiene un comportamiento marcado en aquellos que consumieron en al menos 4 meses, y lo mismo con aquellos que consumieron en más de 10 meses. Si incorporamos a los clientes que tienen esos comportamientos de consumo marcado en la segmentación, podríamos correr el riesgo de tener una sobredimensión del comportamiento de estos grupos mencionados. La acción a seguir es quitar a estos clientes del análisis clúster, y quedarnos con los de un comportamiento “estable”: aquellos cuya frecuencia de consumo es 5 a 10 meses.

```
# Usamos la base que creamos a nivel de cliente, y excluimos a aquellos clientes que consumen en al menos 4 meses, incluyendo aquellos que consumen en más de 10 meses
consumo_cliente_dep = consumo_cliente[consumo_cliente$frec_mes >= 5 & consumo_cliente$frec_mes <= 10,]
```

```
# Filtramos la base inicial y nos quedamos solo con los clientes que consumen en al menos 4 meses, incluyendo aquellos que consumen en más de 10 meses
DataConsumo_dep = filter(DataConsumo, DataConsumo$cliente %in% consumo_cliente_dep$cliente)
```

Una vez obtenida la base de clientes, con los comportamiento “atípicos” excluidos, seguiremos con el agrupamiento para conseguir la base final con la que haremos la segmentación.

La idea es tener una base de datos a nivel de cliente con las transacciones que realizan en cada rubro, esto nos permitirá obtener segmentos de rubros de consumo, con los que podremos brindar una oferta según el segmento en el que está el cliente

```
# Con base en la data anterior, generamos una agregación de la cantidad de transacciones
Data_Consumo_Cli=DataConsumo_dep %>%
  group_by(cliente,grupoGiro,edad,ingreso,sexo) %>%
  summarise(sum_trx = sum(trx))
```

```
# Pivotearemos la data anterior, para tener los consumos de rubros a nivel de cliente. Para ello usaremos la función spread
Data_Consumo_Rubros=spread(data=Data_Consumo_Cli, key=grupoGiro,value = sum_trx)
```

```
head(Data_Consumo_Rubros)
```

```

## # A tibble: 6 x 26
## # Groups: cliente, edad, ingreso [2,240,000]
##   cliente edad ingreso sexo  ``` alqbienes artcultura belleza clubmkt
##   <fct>   <fct> <fct> <fct> <int>    <int>    <int>    <int>    <int>
## 1 1      [7. ~ [3. <= M     NA      1      NA      2      NA
## 2 2      [7. ~ [8. > ~ M     NA      NA      NA      NA      NA
## 3 6      [7. ~ [3. <= M     NA      NA      NA      NA      NA
## 4 9      [7. ~ [4. <= F     NA      NA      NA      1      NA
## 5 11     [7. ~ [4. <= M     NA      NA      NA      NA      NA
## 6 13     [7. ~ [3. <= F     NA      NA      NA      NA      NA
## # ... with 17 more variables: `enseñanza` <int>, entretenimiento <int>,
## #   financiero <int>, hogaroficina <int>, informatica <int>,
## #   prodelectro <int>, prodlocal <int>, prodpersondiv <int>,
## #   prodsuper <int>, profdiverso <int>, restbar <int>, ropamoda <int>,
## #   salud <int>, telcom <int>, tiendadepar <int>, transplaerea <int>,
## #   vehrep <int>

```

Normalización

Con la base anterior, hacemos un resumen de los principales estadísticos, vemos que, si bien están en una misma escala (cantidad de transacciones), existe una dispersión alta en cada rubro (ver gráfico de histogramas y cajas).

```

# Existe un rubro `**`, al cual cambiaremos de nombre por 'otros'
colnames(Data_Consumo_Rubros)[5] = "otros"

```

```

# Sacamos los principales estadísticos de cada rubro. primero, sacamos otro data frame de las columnas
DataRubros = Data_Consumo_Rubros[,c(5:26)]
summary(DataRubros)

```

```

##      otros      alqbienes      artcultura      belleza
## Min.   :1.0      Min.   : 1.000      Min.   : 1.000      Min.   : 1.000
## 1st Qu.:1.0      1st Qu.: 1.000      1st Qu.: 1.000      1st Qu.: 1.000
## Median :1.0      Median : 1.000      Median : 1.000      Median : 1.000
## Mean   :1.4      Mean   : 1.999      Mean   : 1.756      Mean   : 1.941
## 3rd Qu.:2.0      3rd Qu.: 2.000      3rd Qu.: 2.000      3rd Qu.: 2.000
## Max.   :4.0      Max.   :29.000      Max.   :14.000      Max.   :18.000
## NA's   :10771     NA's   :9968       NA's   :10257     NA's   :9627
##      clubmkt      enseñanza      entretenimiento      financiero
## Min.   : 1.000      Min.   : 1.000      Min.   : 1.000      Min.   : 1.000
## 1st Qu.: 1.000      1st Qu.: 1.000      1st Qu.: 1.000      1st Qu.: 1.000
## Median : 3.000      Median : 2.000      Median : 2.000      Median : 1.000
## Mean   : 3.653      Mean   : 2.547      Mean   : 5.076      Mean   : 2.833
## 3rd Qu.: 5.000      3rd Qu.: 3.000      3rd Qu.: 4.000      3rd Qu.: 3.000
## Max.   :26.000      Max.   :20.000      Max.   :182.000     Max.   :28.000
## NA's   : 9851      NA's   :9678       NA's   :6967       NA's   :9666
##      hogaroficina      informatica      prodelectro      prodlocal
## Min.   : 1.000      Min.   :1.000      Min.   : 1.000      Min.   : 1.000
## 1st Qu.: 1.000      1st Qu.:1.000      1st Qu.: 1.000      1st Qu.: 1.000
## Median : 1.000      Median :1.000      Median : 1.000      Median : 1.000
## Mean   : 1.972      Mean   :1.065      Mean   : 1.431      Mean   : 1.924
## 3rd Qu.: 2.000      3rd Qu.:1.000      3rd Qu.: 2.000      3rd Qu.: 2.000
## Max.   :24.000      Max.   :3.000      Max.   :24.000      Max.   :49.000
## NA's   :10545      NA's   :10795      NA's   :9305       NA's   :9221
##      prodpersondiv      prodsuper      profdiverso      restbar

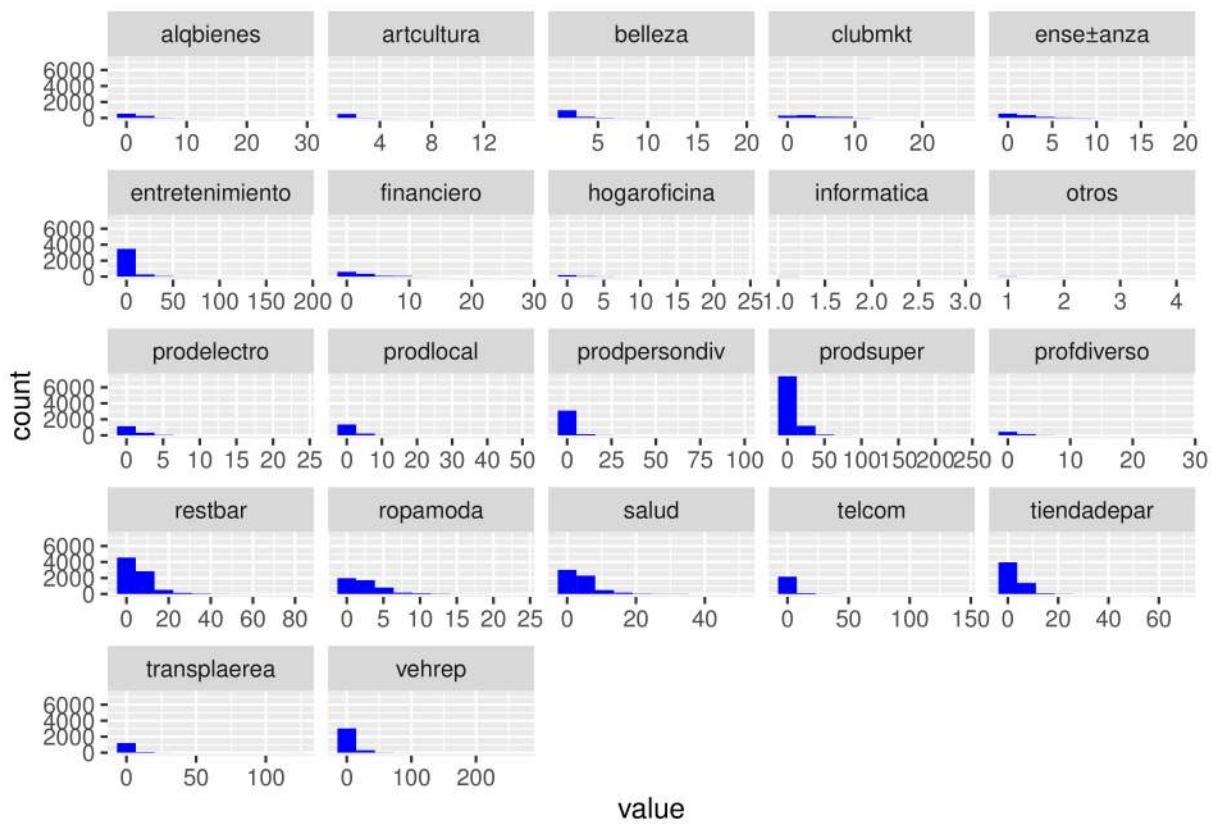
```

```

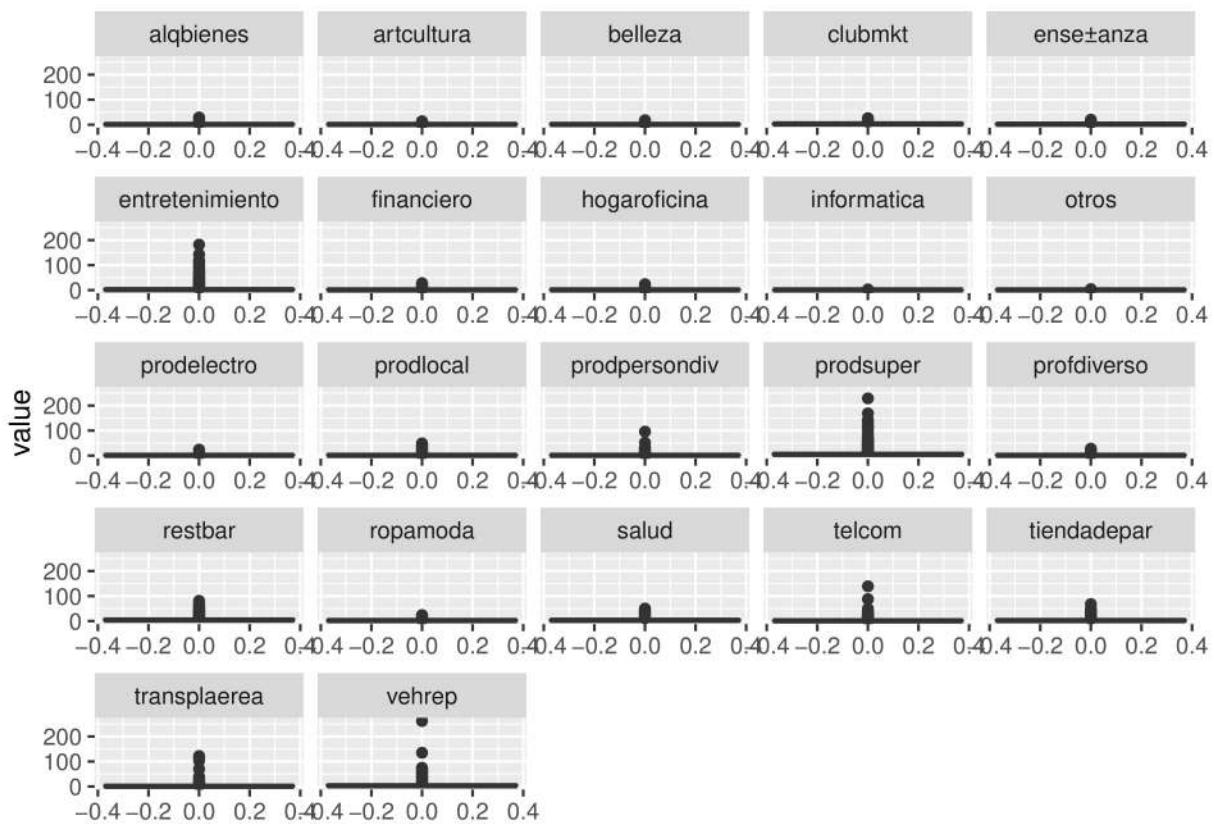
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.00
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.00
##  Median : 1.000   Median : 5.000   Median : 1.000   Median : 4.00
##  Mean   : 2.171   Mean   : 7.252   Mean   : 1.908   Mean   : 5.96
##  3rd Qu.: 2.000   3rd Qu.: 9.000   3rd Qu.: 2.000   3rd Qu.: 8.00
##  Max.   :96.000   Max.   :229.000  Max.   :28.000   Max.   :81.00
##  NA's   :7556    NA's   :2105    NA's   :10188   NA's   :2671
##  ropamoda      salud      telcom      tiendadepar
##  Min.   : 1.000   Min.   : 1.00   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 1.00   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 2.000   Median : 3.00   Median : 1.000   Median : 2.000
##  Mean   : 2.703   Mean   : 4.26   Mean   : 2.665   Mean   : 3.092
##  3rd Qu.: 3.000   3rd Qu.: 5.00   3rd Qu.: 2.000   3rd Qu.: 4.000
##  Max.   :24.000   Max.   :50.00   Max.   :139.000  Max.   :68.000
##  NA's   :5971    NA's   :4745    NA's   :8527    NA's   :5357
##  transplaerea  vehrep
##  Min.   : 1.000   Min.   : 1.00
##  1st Qu.: 1.000   1st Qu.: 1.00
##  Median : 1.000   Median : 3.00
##  Mean   : 2.914   Mean   : 6.12
##  3rd Qu.: 3.000   3rd Qu.: 7.00
##  Max.   :122.000  Max.   :262.00
##  NA's   :9535    NA's   :7469

# Graficamos mediante histogramas, las cantidades de transacciones de cada rubro
ggplot(gather(DataRubros), aes(value)) +
  geom_histogram(bins = 10, fill="blue") +
  facet_wrap(~key, scales = 'free_x')

```



```
# Graficamos mediante diagramas de cajas, las cantidades de transacciones de cada rubro. Vemos que existen
ggplot(gather(DataRubros), aes(y=value)) +
  geom_boxplot(bins = 10, fill="green") +
  facet_wrap(~key, scales = 'free_x')
```



Normalizamos con la estandarización normal

```
# Convertimos en ceros los NA, porque son valores de cantidad de transacciones en el rubro.
for (i in colnames(DataRubros)){
  Data_Consumo_Rubros[,i][is.na(Data_Consumo_Rubros[,i])] = 0
}

# Estandarizamos con la función scale
for (i in colnames(DataRubros)){
  Data_Consumo_Rubros[,i] = scale(unlist(Data_Consumo_Rubros[,i]))
}

# Mostramos la data estandarizada
head(Data_Consumo_Rubros)

## # A tibble: 6 x 26
## # Groups: cliente, edad, ingreso [2,240,000]
##   cliente edad  ingreso sexo otros[,1] alqbienes[,1] artcultura[,1]
##   <fct>   <fct> <fct>   <fct>   <dbl>      <dbl>      <dbl>
## 1 1       [7. ~ [3. <=~ M      -0.0633      0.969      -0.160
## 2 2       [7. ~ [8. > ~ M      -0.0633     -0.182      -0.160
## 3 6       [7. ~ [3. <=~ M      -0.0633     -0.182      -0.160
## 4 9       [7. ~ [4. <=~ F      -0.0633     -0.182      -0.160
## 5 11      [7. ~ [4. <=~ M      -0.0633     -0.182      -0.160
## 6 13      [7. ~ [3. <=~ F      -0.0633     -0.182      -0.160
```

```
## # ... with 19 more variables: belleza[,1] <dbl>, clubmkt[,1] <dbl>,
## # `enseñanza`[,1] <dbl>, entretenimiento[,1] <dbl>,
## # financiero[,1] <dbl>, hogaroficina[,1] <dbl>, informatica[,1] <dbl>,
## # prodelectro[,1] <dbl>, prodlocal[,1] <dbl>, prodpersondiv[,1] <dbl>,
## # prodsuper[,1] <dbl>, profdiverso[,1] <dbl>, restbar[,1] <dbl>,
## # ropamoda[,1] <dbl>, salud[,1] <dbl>, telcom[,1] <dbl>,
## # tiendadepar[,1] <dbl>, transplaerea[,1] <dbl>, vehrep[,1] <dbl>
```

Gestión de valores nulos y outliers

Existen valores nulos en las variables edad, ingreso y sexo, pero no representan más del 8.5% respecto al total de datos. En esta etapa, lo más importante es tener las variables de los rubros sin nulos, porque a partir de ella se construirán los clústeres. Al final, para caracterizar los clústeres, podemos omitir estos valores nulos, dada su poca cantidad.

```
# Mostramos la cantidad de ceros y NA por variable
df_status(Data_Consumo_Rubros)
```

##	variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
## 1	cliente	0	0	0	0.00	0	0	factor	10826
## 2	edad	0	0	374	3.45	0	0	factor	7
## 3	ingreso	0	0	907	8.38	0	0	factor	8
## 4	sexo	0	0	347	3.21	0	0	factor	2
## 5	otros	0	0	0	0.00	0	0	matrix	5
## 6	alqbienes	0	0	0	0.00	0	0	matrix	20
## 7	artcultura	0	0	0	0.00	0	0	matrix	15
## 8	belleza	0	0	0	0.00	0	0	matrix	17
## 9	clubmkt	0	0	0	0.00	0	0	matrix	17
## 10	enseñanza	0	0	0	0.00	0	0	matrix	18
## 11	entretenimiento	0	0	0	0.00	0	0	matrix	83
## 12	financiero	0	0	0	0.00	0	0	matrix	23
## 13	hogaroficina	0	0	0	0.00	0	0	matrix	12
## 14	informatica	0	0	0	0.00	0	0	matrix	3
## 15	prodelectro	0	0	0	0.00	0	0	matrix	12
## 16	prodlocal	0	0	0	0.00	0	0	matrix	28
## 17	prodpersondiv	0	0	0	0.00	0	0	matrix	30
## 18	prodsuper	0	0	0	0.00	0	0	matrix	86
## 19	profdiverso	0	0	0	0.00	0	0	matrix	17
## 20	restbar	0	0	0	0.00	0	0	matrix	67
## 21	ropamoda	0	0	0	0.00	0	0	matrix	24
## 22	salud	0	0	0	0.00	0	0	matrix	42
## 23	telcom	0	0	0	0.00	0	0	matrix	38
## 24	tiendadepar	0	0	0	0.00	0	0	matrix	37
## 25	transplaerea	0	0	0	0.00	0	0	matrix	31
## 26	vehrep	0	0	0	0.00	0	0	matrix	61

Lo que sí vamos a controlar serán los outliers, que ya vimos en el diagrama de cajas que eran muchos por rubro. Los excluiremos porque los outliers sí condicionan la segmentación, al ser comportamientos atípicos.

```
# Convertimos en numéricas nuestras variables de rubro
for (i in 5:26){
  Data_Consumo_Rubros[,i] = as.numeric(unlist(Data_Consumo_Rubros[,i]))
}

# Hacemos la gestión de outliers
Data_Consumo_Rubros$outliers = FALSE
```

```

for (i in 5:26){
  columna = Data_Consumo_Rubros[,i]
  media=mean(unlist(columna))
  desviacion = sd(unlist(columna))
  Data_Consumo_Rubros$outliers = (Data_Consumo_Rubros$outliers | columna>(media+3*desviacion) | col
}
# Marcamos los TRUE y FALSE
table(Data_Consumo_Rubros$outliers)

##
## FALSE TRUE
## 7963 2863

# Separamos el dataframe donde tenemos los outliers. Podría ser útil para un estudio nuevo sobre client
datosOutliers = Data_Consumo_Rubros[Data_Consumo_Rubros$outliers,]

# Marcamos los outliers en la data, los eliminamos y dibujamos
Data_Consumo_Rubros=Data_Consumo_Rubros[!Data_Consumo_Rubros$outliers,]

# Y ya no necesitamos que haya una columna "outliers"
Data_Consumo_Rubros$outliers=NULL

# Mostramos la data sin outliers
print(Data_Consumo_Rubros)

## # A tibble: 7,963 x 26
## # Groups: cliente, edad, ingreso [2,242,558]
##   cliente edad ingreso sexo otros alqbienes artcultura belleza clubmkt
##   <fct>   <fct> <fct>   <fct>   <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 1       [7. ~ [3. <= M   -0.0633    0.969    -0.160    2.04    -0.245
## 2 6       [7. ~ [3. <= M   -0.0633   -0.182    -0.160   -0.246   -0.245
## 3 11      [7. ~ [4. <= M   -0.0633   -0.182    -0.160   -0.246   -0.245
## 4 13      [7. ~ [3. <= F   -0.0633   -0.182    -0.160   -0.246   -0.245
## 5 14      [7. ~ [4. <= M   -0.0633   -0.182    -0.160   -0.246   -0.245
## 6 19      [7. ~ [7. <= F   -0.0633   -0.182    -0.160   -0.246   -0.245
## 7 26      [7. ~ [3. <= M   -0.0633   -0.182    -0.160   -0.246   1.24
## 8 43      [7. ~ [3. <= M   -0.0633   -0.182    -0.160   -0.246   -0.245
## 9 44      [7. ~ [8. > ~ M   -0.0633   -0.182    -0.160   -0.246   -0.245
## 10 47     [7. ~ [4. <= M   -0.0633   -0.182    -0.160   -0.246   -0.245
## # ... with 7,953 more rows, and 17 more variables: `enseñanza` <dbl>,
## #   entretenimiento <dbl>, financiero <dbl>, hogaroficina <dbl>,
## #   informatica <dbl>, prodelectro <dbl>, prodlocal <dbl>,
## #   prodpersondiv <dbl>, prodsuper <dbl>, profdiverso <dbl>,
## #   restbar <dbl>, ropamoda <dbl>, salud <dbl>, telcom <dbl>,
## #   tiendadepar <dbl>, transplaarea <dbl>, vehrep <dbl>

```

Discretización

Vemos que el conjunto de datos ya viene con las variables discretizadas: edad e ingreso. Lo que sí vamos a utilizar es la reducción de la dimensionalidad.

Reducción de dimensionalidad

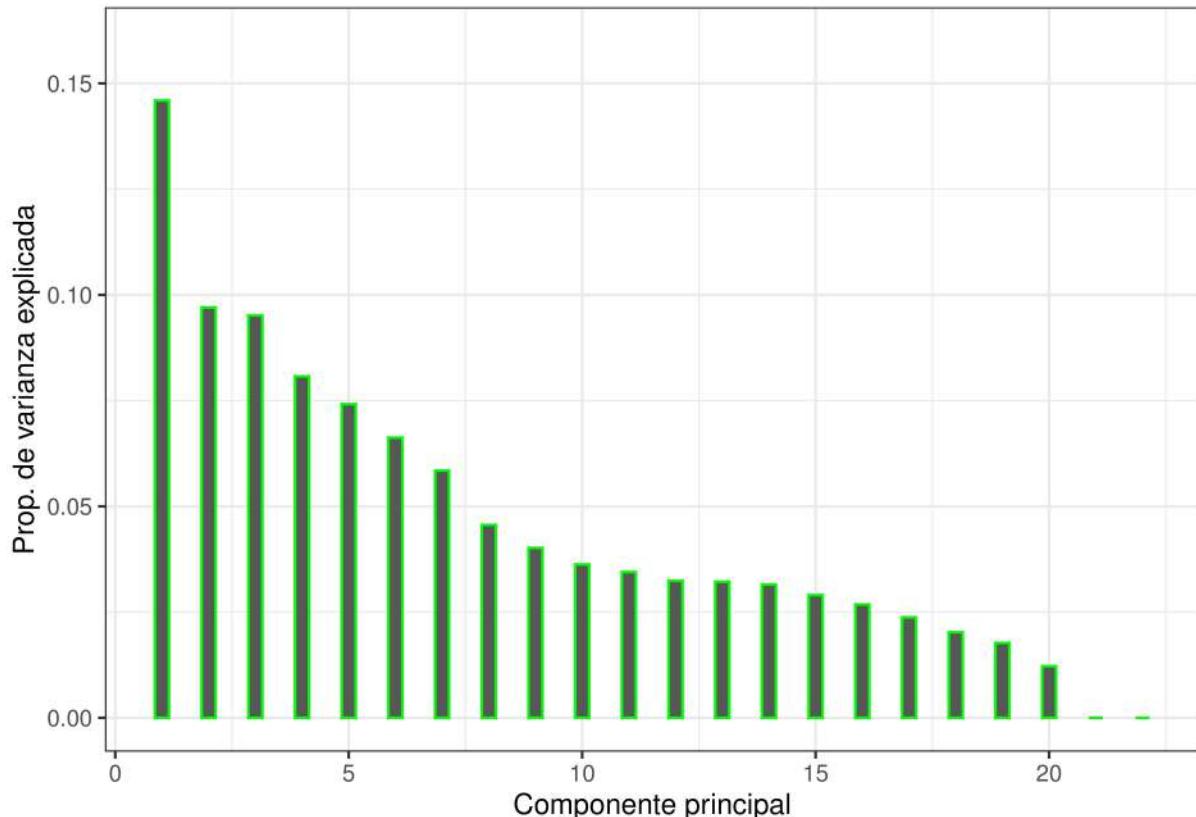
En este paso, verificaremos si las variables de los rubros pueden reducirse de 22 a un número menor. Para esto, usaremos el análisis de componentes principales.

Como vemos en el gráfico, todas las componentes aportan, en una proporción pequeña, a explicar la variabilidad de los datos. Buscaremos explicar al menos el 75% de la variabilidad (información), y esto lo encontramos en los 11 primeros componentes, 77.42%.

```
# Usaremos la función prcomp, para realizar el análisis de componentes principales
pca <- prcomp(Data_Consumo_Rubros[,5:26])

# Calculamos la proporción de varianza explicada por cada componente
prop_varianza <- pca$sdev^2 / sum(pca$sdev^2)

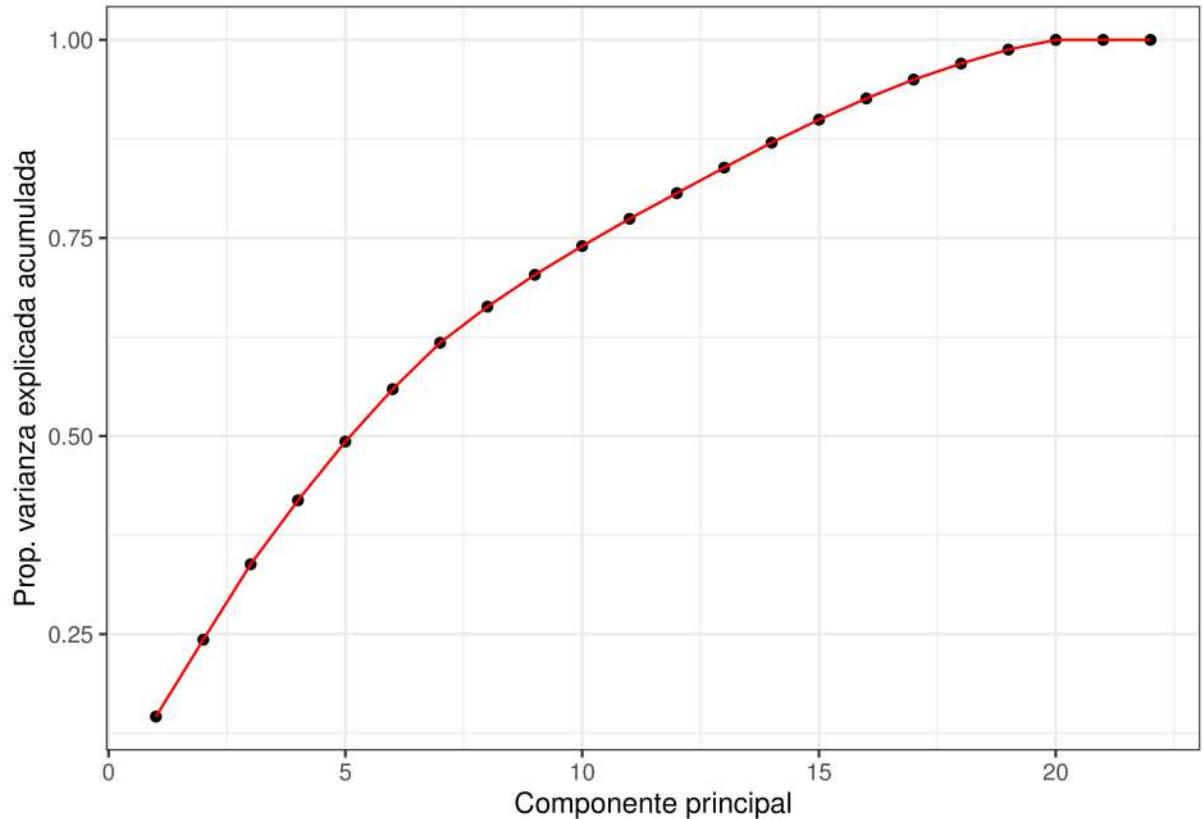
# Graficamos la proporción de varianza explicada por cada componente
ggplot(data = data.frame(prop_varianza),
       aes(x = 1:22, y = prop_varianza)) +
  geom_col(width = 0.3, color="green") +
  scale_y_continuous(limits = c(0,0.16)) +
  theme_bw() +
  labs(x = "Componente principal",
       y = "Prop. de varianza explicada")
```



```
# Calculamos la proporción acumulada de varianza explicada por cada componente
prop_varianza_acum <- cumsum(prop_varianza)

# Graficamos la proporción acumulada de varianza explicada por cada componente
ggplot(data = data.frame(prop_varianza_acum),
       aes(x = 1:22, y = prop_varianza_acum, group = 1)) +
  geom_point() +
  geom_line(color="red") +
```

```
theme_bw() +
  labs(x = "Componente principal",
       y = "Prop. varianza explicada acumulada")
```



```
# Mostramos la varianza acumulada explicada por la cantidad de componentes elegidos (11)
prop_varianza_acum[11]
```

```
## [1] 0.7742096
```

Ahora, trabajaremos en las variables capturadas por cada componente. En este paso, nos ayudaremos de un archivo excel para resaltar las variables con mayor aporte en cada componente (sea positivo o negativo).

Agregamos estos componentes y generamos el dataset reducido.

```
# Exportamos los componentes y las variables a un csv:
# write.csv(pca$rotation, file = 'pca_consumo.csv')
```

```
# Traemos a un data frame, los 11 componentes principales que elegimos:
comp_princ = data.frame(pca$x[,1:11])
```

```
# Los renombramos según lo trabajado en el archivo excel:
```

```
colnames(comp_princ)[1] = "sector_viaje"
colnames(comp_princ)[2] = "sector_rest_bar"
colnames(comp_princ)[3] = "sector_salud"
colnames(comp_princ)[4] = "sector_supermcd"
colnames(comp_princ)[5] = "sector_electronico"
colnames(comp_princ)[6] = "sector_tienda_depart"
colnames(comp_princ)[7] = "sector_vehiculo"
```

```

colnames(comp_princ)[8] = "sector_finanzas"
colnames(comp_princ)[9] = "sector_eduacion"
colnames(comp_princ)[10] = "sector_clubes"
colnames(comp_princ)[11] = "sector_entretenimiento"

# Traemos la data de consumo rubros, que está a nivel de cliente, y que tiene la información descriptiva
Data_Consumo_Rubros = data.frame(Data_Consumo_Rubros)

# Calculamos la base final, con los datos descriptivos a nivel de clientes, y los 11 componentes principales
Data_Consumo_Final = cbind(Data_Consumo_Rubros[,c(1:4)],comp_princ)
Data_Consumo_Final = data.frame(Data_Consumo_Final)

head(Data_Consumo_Final)

##   cliente      edad     ingreso sexo sector_viaje sector_rest_bar
## 1       1 [7. > 55] [3. <=2500]   M   -0.9633356   0.58862591
## 2       6 [7. > 55] [3. <=2500]   M    0.8703571  -0.07902649
## 3      11 [7. > 55] [4. <=3500]   M    0.2749834  -0.64359851
## 4      13 [7. > 55] [3. <=2500]   F    0.6653967  -0.36174006
## 5      14 [7. > 55] [4. <=3500]   M   -1.6889264  -1.46703423
## 6      19 [7. > 55] [7. <=8000]   F   -0.6831152  -0.08900762
##   sector_salud sector_supermcd sector_electronico sector_tienda_depart
## 1  -0.27410784   -0.18373126   -0.05231577  -0.52969381
## 2   0.41547798    0.06762205    0.15447993  -0.40728215
## 3  -0.02049689   -0.17529944    1.24848373  0.42599123
## 4   0.56954823   -0.04190454    0.13778052  -0.58239640
## 5   1.28671726    0.42862897    0.67979925  -0.26697215
## 6   0.64895396   -0.76949538   -0.66507316   0.07428788
##   sector_vehiculo sector_finanzas sector_eduacion sector_clubes
## 1   -1.5908802   -0.12483478   -0.6730529710  -0.63187684
## 2   -0.3707258   -0.15997458    0.0050167092   0.10406531
## 3   0.3366951   -0.24802621    0.0505777336   0.05697767
## 4   0.2394136    0.07208933   -0.0008614832   0.04901315
## 5   0.3199919   -0.89925496    0.0671777024   0.08353895
## 6   -1.1553028   -0.41198694    0.1862892514   0.12250945
##   sector_entretenimiento
## 1           -1.46337206
## 2            0.24722983
## 3            0.14442957
## 4            0.21419983
## 5            0.06170583
## 6            0.26440126

```

Análisis de los datos

Normalidad y homocedasticidad

Para la prueba de normalidad definimos la hipótesis

H_0 : La muestra proviene de una distribución normal. H_1 : La muestra no proviene de una distribución normal

Ahora fijamos un nivel de significancia de 0.05

Y el criterio de decisión es:

$$\text{Si } p < 0.05 \text{ Se rechaza } H_0 \text{ Si } p \geq 0.05 \text{ No se rechaza } H_0$$

Usamos el test de normalidad de Lilliefors que es el equivalente a Shapiro Wilks, pero que asume media y varianza desconocidas.

```
# install.packages("normtest")
library(nortest)

lillie.test(x = Data_Consumo_Final$sector_viaje)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_viaje
## D = 0.098419, p-value < 2.2e-16

lillie.test(x = Data_Consumo_Final$sector_rest_bar)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_rest_bar
## D = 0.052373, p-value < 2.2e-16

lillie.test(x = Data_Consumo_Final$sector_salud)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_salud
## D = 0.10684, p-value < 2.2e-16

lillie.test(x = Data_Consumo_Final$sector_supermcd)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_supermcd
## D = 0.076248, p-value < 2.2e-16

lillie.test(x = Data_Consumo_Final$sector_electronico)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_electronico
## D = 0.18557, p-value < 2.2e-16

lillie.test(x = Data_Consumo_Final$sector_tienda_depart)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_tienda_depart
## D = 0.071975, p-value < 2.2e-16
```

```

lillie.test(x = Data_Consumo_Final$sector_vehiculo)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_vehiculo
## D = 0.13484, p-value < 2.2e-16

lillie.test(x = Data_Consumo_Final$sector_finanzas)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_finanzas
## D = 0.18284, p-value < 2.2e-16

lillie.test(x = Data_Consumo_Final$sector_eduacion)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_eduacion
## D = 0.22445, p-value < 2.2e-16

lillie.test(x = Data_Consumo_Final$sector_clubes)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_clubes
## D = 0.28183, p-value < 2.2e-16

lillie.test(x = Data_Consumo_Final$sector_entretenimiento)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: Data_Consumo_Final$sector_entretenimiento
## D = 0.20252, p-value < 2.2e-16

```

Observamos que todas las variables tienen un p-valor menor que 5%. Por lo tanto, ninguna de las variables de transacciones en los sectores de consumos se distribuyen normalmente.

Ahora, probaremos la homogeneidad de varianzas usando la prueba de Levene.

```

library(car)

leveneTest(y = Data_Consumo_Cli$sum_trx, group = Data_Consumo_Cli$sexo, center = "mean")

## Levene's Test for Homogeneity of Variance (center = "mean")
##          Df F value    Pr(>F)
## group      1 42.194 8.337e-11 ***
##          55375
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

leveneTest(y = Data_Consumo_Cli$sum_trx, group = Data_Consumo_Cli$edad, center = "mean")

## Levene's Test for Homogeneity of Variance (center = "mean")

```

```

##           Df F value    Pr(>F)
## group      6 7.1717 1.159e-07 ***
##           55236
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
leveneTest(y = Data_Consumo_Cli$sum_trx, group = Data_Consumo_Cli$ingreso, center = "mean")

## Levene's Test for Homogeneity of Variance (center = "mean")
##           Df F value    Pr(>F)
## group      7 11.509 1.085e-14 ***
##           52311
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Observamos que existen diferencias significativas entre las varianzas de las categorías de las variables sexo, ingresos y edad.

Que las varianzas de las categorías de las variables sean distintas resulta bueno para el objetivo que perseguimos. Buscamos segmentar, y para ello necesitamos variables que aporten información que diferencie el consumo. Por el lado de la normalidad, si bien las transacciones por cada sector no se distribuyen normalmente, no importa tanto para nuestro modelo, dato que es no supervisado.

Regresión

Para esta parte, usaremos el análisis de regresión, más allá de ajustar un modelo lineal teórico, buscamos obtener la significancia de las variables independientes (categóricas) para explicar el consumo.

```

regresion_clustering <- lm(sum_trx ~ sexo+ingreso+edad, data = Data_Consumo_Cli)
summary(regresion_clustering)

```

```

##
## Call:
## lm(formula = sum_trx ~ sexo + ingreso + edad, data = Data_Consumo_Cli)
##
## Residuals:
##    Min     1Q   Median     3Q    Max
## -4.068 -3.151 -2.074  0.623 257.493
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 4.46640   0.30409 14.688 < 2e-16 ***
## sexoM       0.24505   0.05815  4.214 2.51e-05 ***
## ingreso[2. <=1500] -0.16337  0.31361 -0.521  0.60242
## ingreso[3. <=2500] -0.52527  0.31731 -1.655  0.09786 .
## ingreso[4. <=3500] -0.84732  0.32471 -2.609  0.00907 ** 
## ingreso[5. <=4500] -0.92535  0.33769 -2.740  0.00614 ** 
## ingreso[6. <=6000] -0.87875  0.35359 -2.485  0.01295 *  
## ingreso[7. <=8000] -1.01321  0.36714 -2.760  0.00579 ** 
## ingreso[8. > 8000] -0.96623  0.37704 -2.563  0.01039 *  
## edad[2. <=30]      0.07411  0.11156  0.664  0.50652
## edad[3. <=35]      0.23109  0.11655  1.983  0.04740 *  
## edad[4. <=40]      0.07523  0.12087  0.622  0.53367
## edad[5. <=45]      0.20938  0.12793  1.637  0.10171
## edad[6. <=55]      0.32052  0.12197  2.628  0.00860 ** 
## edad[7. > 55]      0.52007  0.12944  4.018  5.88e-05 ***
## ---

```

```

## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.581 on 52170 degrees of freedom
## (5212 observations deleted due to missingness)
## Multiple R-squared: 0.001837, Adjusted R-squared: 0.001569
## F-statistic: 6.857 on 14 and 52170 DF, p-value: 2.849e-14

```

Observamos que los clientes con ingresos menores que 1500 y de edades menores que 30 y las comprendidas entre 35 y 45 son variables no significativas. Esto nos habla sobre el aporte de cada variable, y observamos que, en general, las independientes aportan información relevante, por lo que la clusterización es idónea.

Correlaciones

Las variables numéricas que analizaremos son los componentes calculados. Evidentemente todas tienen correlación 0.

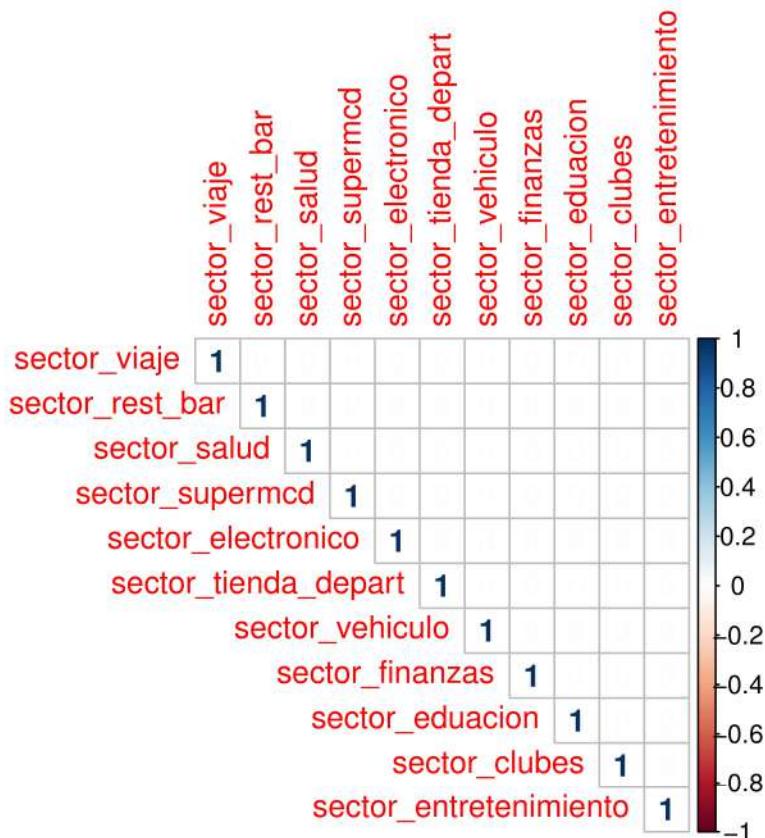
```

# Llamamos a la librería corrplot
library(corrplot)

# Hallamos la matriz de correlación
correlacion=cor(Data_Consumo_Final[,c(5:15)])

# Graficamos
corrplot(correlacion, method="number", type="upper")

```



Clustering k-means

En este apartado obtendremos la segmentación de consumo, a través del método de clasificación k-means. Es importante recordar que la principal dificultad de este método, está en encontrar el número óptimo de clústers. Para ello, usaremos diferentes indicadores.

Codo de Jambu

Bajo este método, si bien la caída de la curva no se acelera en ningún punto en particular, podemos observar que con 7 clústers empieza a caer más despacio. Este método sugiere usar 7 clústers.

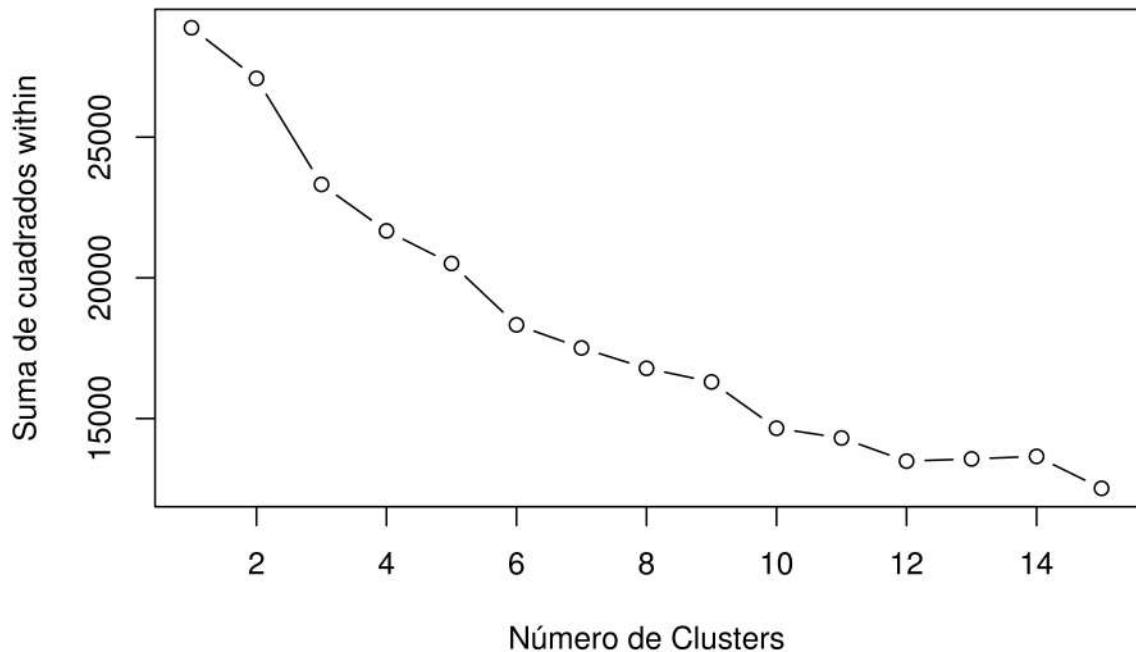
```
head(Data_Consumo_Final[,5:15])

##   sector_viaje sector_rest_bar sector_salud sector_supermcd
## 1   -0.9633356    0.58862591  -0.27410784   -0.18373126
## 2    0.8703571   -0.07902649    0.41547798    0.06762205
## 3    0.2749834   -0.64359851  -0.02049689   -0.17529944
## 4    0.6653967   -0.36174006    0.56954823   -0.04190454
## 5   -1.6889264   -1.46703423    1.28671726    0.42862897
## 6   -0.6831152   -0.08900762    0.64895396   -0.76949538
##   sector_electronico sector_tienda_depart sector_vehiculo sector_finanzas
## 1   -0.05231577   -0.52969381   -1.5908802   -0.12483478
## 2    0.15447993   -0.40728215   -0.3707258   -0.15997458
## 3    1.24848373    0.42599123    0.3366951   -0.24802621
## 4    0.13778052   -0.58239640    0.2394136    0.07208933
## 5    0.67977925   -0.26697215    0.3199919   -0.89925496
## 6   -0.66507316    0.07428788   -1.1553028   -0.41198694
##   sector_educacion sector_clubes sector_entretenimiento
## 1   -0.6730529710   -0.63187684   -1.46337206
## 2    0.0050167092    0.10406531    0.24722983
## 3    0.0505777336    0.05697767    0.14442957
## 4   -0.0008614832    0.04901315    0.21419983
## 5    0.0671777024    0.08353895    0.06170583
## 6    0.1862892514    0.12250945    0.26440126

# Calculamos la suma total de cuadrados
wss <- (nrow(Data_Consumo_Final[,5:15])-1)*sum(apply(Data_Consumo_Final[,5:15],2,var))

# Lo calculamos por clusters
for (i in 2:15) wss[i] <- sum(kmeans(Data_Consumo_Final[,5:15],
                                         centers=i)$withinss)

# Graficamos el Codo de Jambu
plot(1:15, wss, type="b", xlab="Número de Clusters",
      ylab="Suma de cuadrados within")
```



Índice Calinski Harabasz

Este índice está basado en una F de anova.

El gráfico sugiere que el número óptimo de segmentos es 12 (punto más alto), pero podríamos considerar también 2, 8, 9, 10 y 11. La idea es obtener un número de clúster moderado, y éste podría ser 8. Veamos con los siguientes índices.

```
# Traemos a la librería fpc
library(fpc)

# Normalizamos la base
data_escale = scale(Data_Consumo_Final[,5:15])

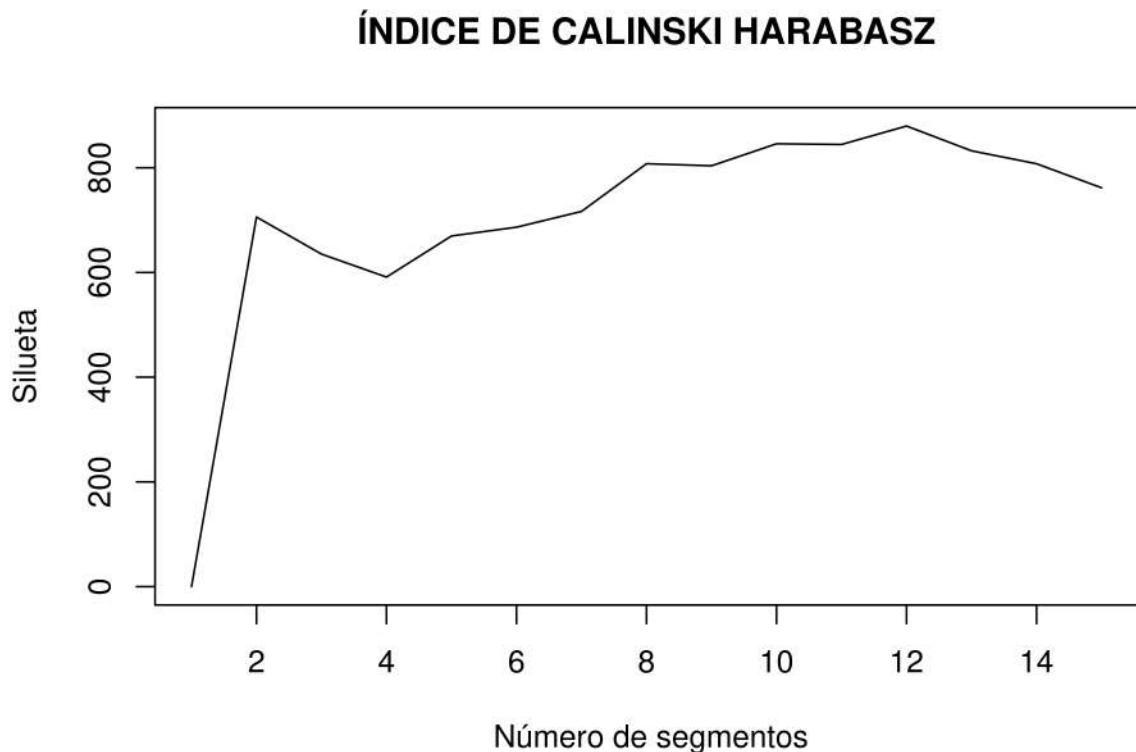
# Inicializamos la silueta, para que se almacen los valores calculados por el número de clústers
silueta <- rep (0 , times =15)

# Aplicamos un bucle para obtener los clústers y la silueta, mediante el criterio de Calinski Harabasz
for (k in 2:15) {
  clus <- kmeansruns(data_escale, krange = 2:15 , criterion = "ch", iter.max = 5, runs = 5, scaledata =
  silueta[k] <- clus$crit[k] }

# Mostramos el número óptimo de segmentos según este indicador: 12
clus$bestk

## [1] 12
```

```
# Graficamos la silueta
plot(1:15 ,silueta ,type = "l",xlab =" Número de segmentos" , ylab ="Silueta", main = "ÍNDICE DE CALINSKI HARABASZ")
```



Método de Silueta Media

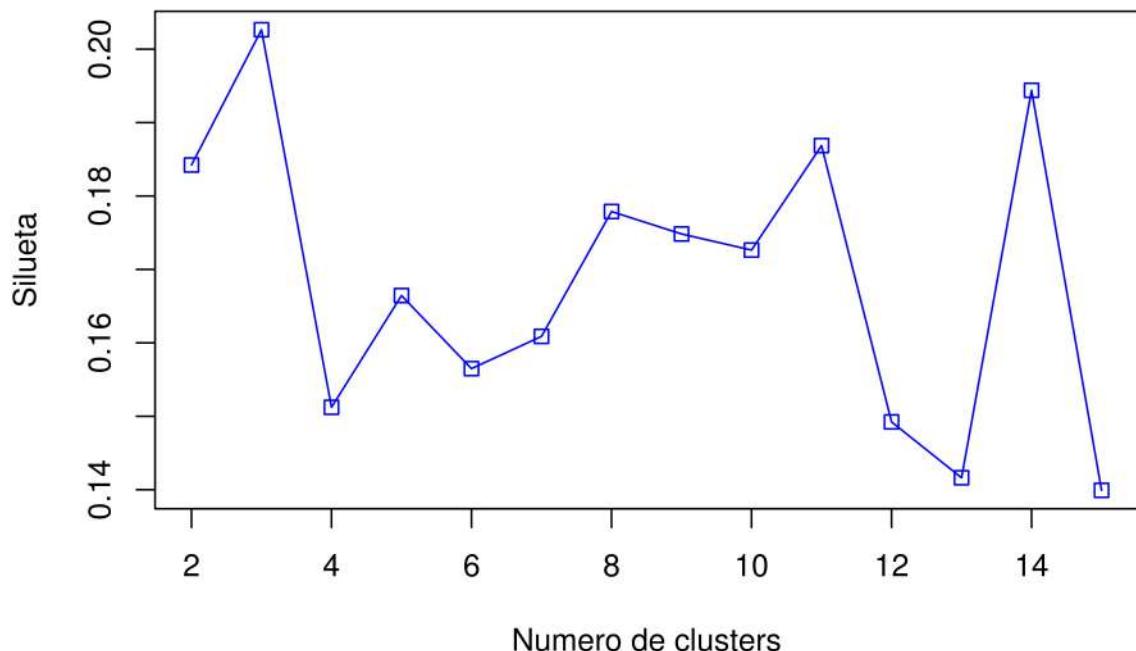
Según este método, vemos que el número óptimo de clústers es 2, aunque otro valor a considerar en segundo orden puede ser 9.

```
# Usamos la data_escala que construimos en el paso anterior
d <- daisy(data_escala)

# Inicializamos nuestra variable a graficar
resultados <- rep(0, 15)

# Aplicamos el bucle con el método de la media de los valores de la silueta
for (i in c(2:15))
{
  fit           <- kmeans(data_escala, i)
  y_cluster     <- fit$cluster
  sk            <- silhouette(y_cluster, d)
  resultados[i] <- mean(sk[,3])
}
# Graficamos
plot(2:15,resultados[2:15],type="o",col="blue",pch=0,xlab="Número de clusters",ylab="Silueta", main= "MÉTODO DE SILUETA MEDIA")
```

Método de la silueta media



Método AWS

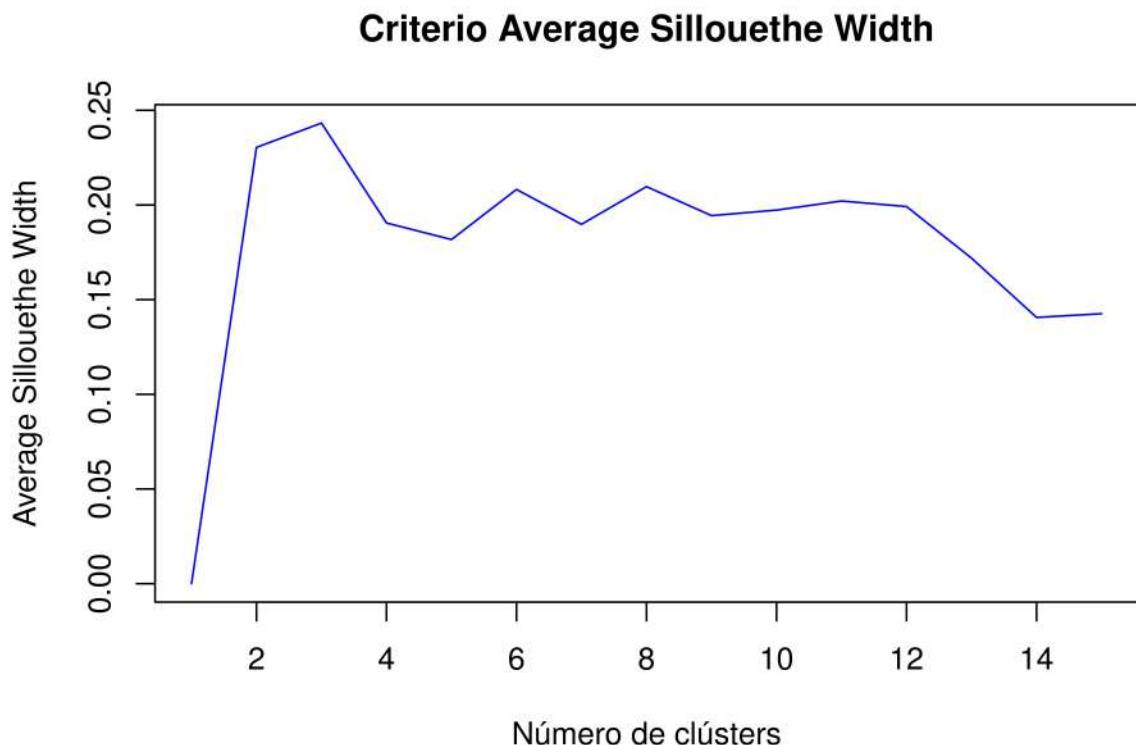
Según este método, vemos que el número óptimo de clústers es 3, aunque otros valores a considerar en segundo orden podrían ser 3, 4, 5 y 8.

```
# Evaluar usando el criterio ASW (average sillouetthe width)
set.seed(2) #Para evitar aleatoriedad en los resultados
clustering_asw <- kmeansruns(data_escale,krange=2:15,criterion="asw",
                               iter.max=100, runs= 100,critout=TRUE)
```

```
## 2  clusters  0.2304167
## 3  clusters  0.2432379
## 4  clusters  0.1904862
## 5  clusters  0.1817358
## 6  clusters  0.2081952
## 7  clusters  0.1897893
## 8  clusters  0.2096906
## 9  clusters  0.1943741
## 10 clusters  0.1972877
## 11 clusters  0.2020901
## 12 clusters  0.1991646
## 13 clusters  0.1719056
## 14 clusters  0.1406084
## 15 clusters  0.1425744

# El número óptimo de segmentos, según el criterio ASW es
clustering_asw$bestk
```

```
## [1] 3
# Graficamos
plot(1:15, clustering_asw$crit, type="l", col="blue", pch=0, xlab="Número de clústers", ylab="Average Sillouette Width")
```



Método GAP STATISTIC

En el blog de rpubs, encontramos la siguiente definición de este método: “Este estadístico compara, para diferentes valores de k , la varianza total intra-cluster observada frente al valor esperado acorde a una distribución uniforme de referencia. La estimación del número óptimo de clusters es el valor k con el que se consigue maximizar el estadístico gap”.¹

Vemos que, según este criterio, el número óptimo de clusters es 15, pudiendo tomar también 11 y 13.

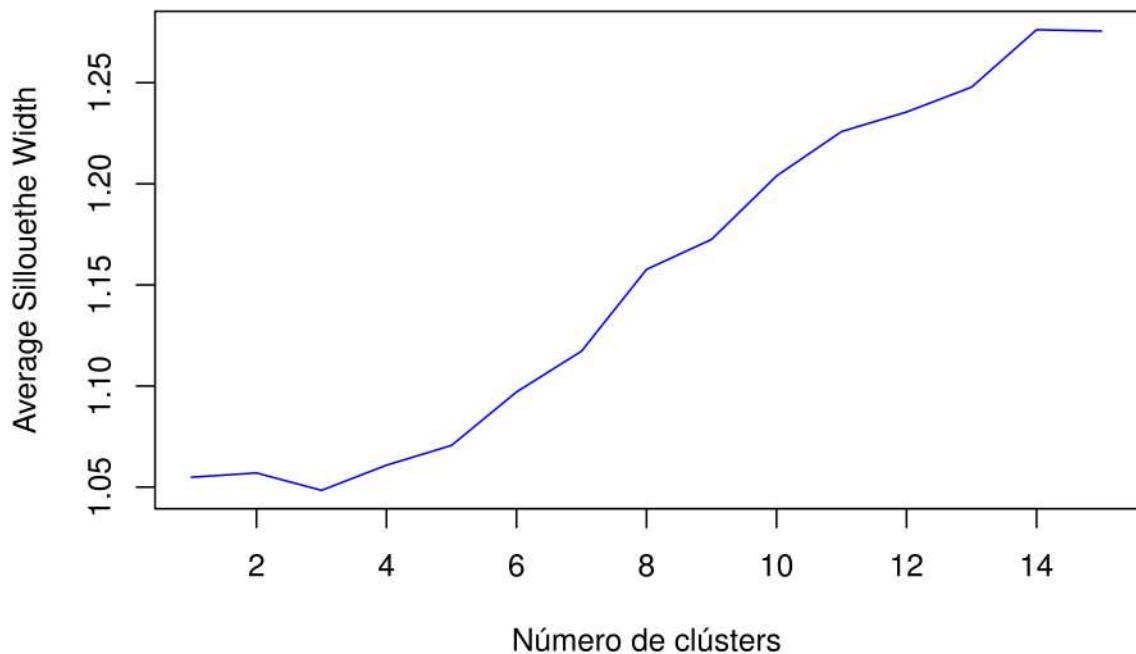
```
# Evaluamos con gap statistic
# Mira el minimo k tal que el gap sea mayor que el gap de k+1 restado de su desviacion
gscar<-clusGap(data_escala,FUN=kmeans,K.max=15,B=10)

# Sacamos a un lado el indicador, para graficarlo
gscar_gap= gscar$Tab[,3]

# Graficamos
plot(1:15,gscar_gap,type="l", col="blue", pch=0, xlab="Número de clústers", ylab="Average Sillouette Width")
```

¹ Fuente: https://rpubs.com/Joaquin_AR/310338

Criterio GAP Statistics



Evaluación de segmentos Bootstrap

Esta evaluación consiste en un muestreo bootstrap para evaluar cuán estable es un segmento dado. La idea es quedarnos con el de mayor índice. Una regla que nos permite medirlo es considerar muy bueno a aquellos mayores que 0.85.

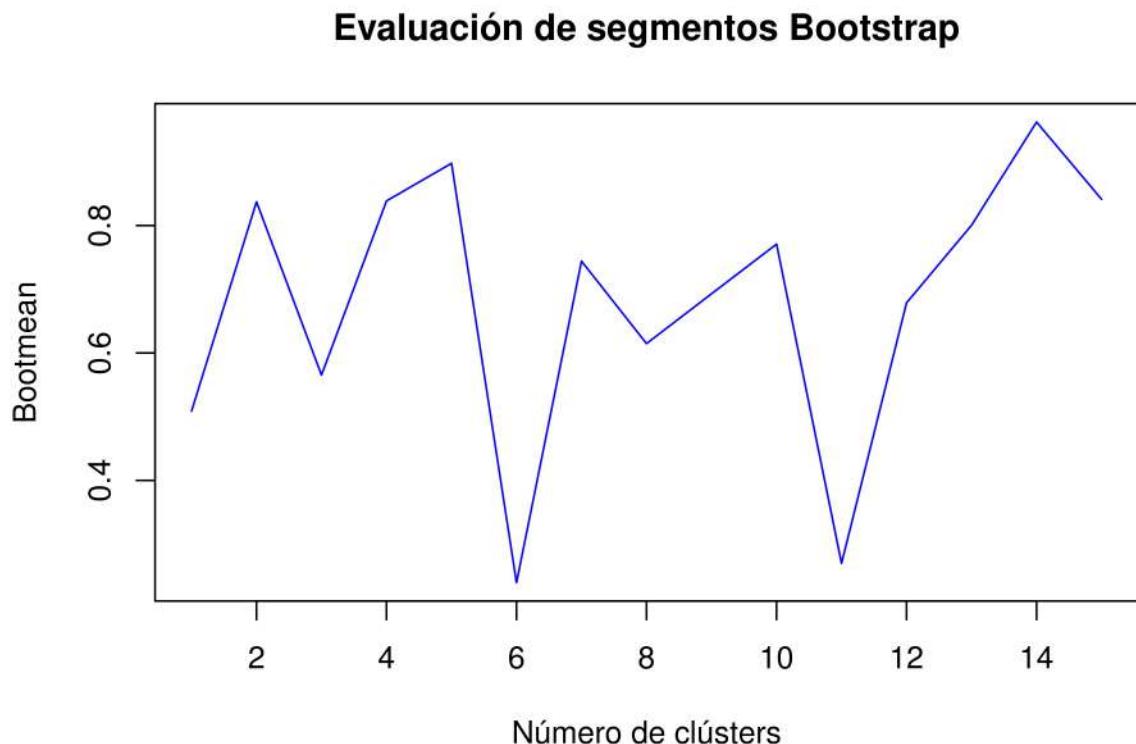
Vemos que, con 8 segmentos conseguimos una alta estabilidad, 0.95.

```
#validar resultados- consistencia
kclusters <- clusterboot(data_escale,B=10,clustermethod=kmeansCBI,k=15,seed=5)
```

```
## boot 1
## boot 2
## boot 3
## boot 4
## boot 5
## boot 6
## boot 7
## boot 8
## boot 9
## boot 10
#la validacion del resultado. <0.75 o .85 muy bueno; <.6 malo
kclusters$bootmean

## [1] 0.5084914 0.8372238 0.5653921 0.8385919 0.8976437 0.2396129 0.7441053
## [8] 0.6147435 0.6928679 0.7708740 0.2698877 0.6786916 0.8004800 0.9624133
## [15] 0.8411915
```

```
# Graficamos:
plot(1:15,kclusters$bootmean,type="l",col="blue",pch=0,xlab="Número de clústers",ylab="Bootmean", main = "Evaluación de segmentos Bootstrap")
```



Decisión

Entonces, después de evaluar los indicadores para tener el número óptimo de clústers, decidimos usar 4 segmentos, porque en todas las pruebas ha salido como el óptimo en un primer o segundo orden. Además, que tiene mucho sentido en la caracterización de lo que buscamos. A priori, las preferencias de consumo de las personas, considerando el abanico de rubros que tenemos, difícilmente se agrupen en 2 segmentos, por ello, 4 se hace un buen número.

Aplicación k-means

En este paso, aplicaremos la segmentación kmeans, con un número de segmentos igual a 4, pero veremos qué pasa si usamos 2 y 8 (que fueron los otros k que salieron en las pruebas).

En los gráficos vemos que con 2 y 4 segmentos obtenemos grupos marcados, mientras que con 8 no tanto. Usaremos 4 segmentos.

```
# Aplicamos kmeans
consumo_clusters <- kmeans(data_escala, 4)

#Vemos el número de iteraciones realizadas, los centroides y tamaños de cada segmento
consumo_clusters$iter # Ver numero de iteraciones

## [1] 4
consumo_clusters$cluster

## [1] 1 3 3 3 2 3 4 3 3 3 3 1 4 2 3 3 2 1 2 3 3 3 1 3 2 4 1 3 1 3 3 3 1
```

```

## [35] 4 3 1 3 2 1 3 2 1 3 3 2 4 3 3 3 3 1 3 2 1 3 3 3 3 4 3 1 3 3 4 2 3 3
## [69] 3 2 1 3 1 2 3 2 3 3 2 4 3 3 4 3 3 3 3 4 3 4 4 4 4 2 3 3 3 4 3 3 4 4 4
## [103] 4 3 3 3 3 2 3 4 3 3 4 1 3 3 3 1 4 3 1 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 4 2
## [137] 2 2 3 3 3 3 3 2 2 4 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 4 3 3 4 4 3 3
## [171] 3 3 4 3 3 4 3 3 3 3 4 3 2 3 3 3 4 1 3 3 3 3 1 4 3 4 3 3 3 2 3 3 3 3 3 3 4 3
## [205] 3 1 3 3 4 3 3 2 3 3 2 3 3 1 1 3 3 2 3 3 3 3 3 3 3 3 3 3 4 4 3 3 3 4 2 3
## [239] 3 3 4 3 3 2 3 2 2 3 2 2 2 1 2 3 3 2 3 3 2 3 1 3 3 3 1 2 1 3 3 3 3 3
## [273] 3 3 1 2 3 1 2 2 2 3 3 3 3 4 4 4 4 3 1 3 3 3 3 4 3 1 2 3 3 3 1 2 3 3 4 3
## [307] 1 3 2 1 3 3 4 1 3 2 1 3 3 4 3 4 3 3 3 4 3 3 2 3 4 3 2 3 1 2 3 2 3 3
## [341] 3 3 2 1 1 4 1 4 3 3 3 3 1 3 3 3 3 2 4 3 3 3 1 2 3 2 3 2 4 3 4 3 1 1
## [375] 3 4 3 2 3 3 2 3 2 3 2 2 3 3 3 3 3 3 4 2 3 2 4 3 3 3 1 4 3 3 4 2 1 3 3
## [409] 1 3 3 4 3 3 1 3 1 3 3 1 3 3 3 3 3 2 1 3 3 1 2 2 4 3 4 3 3 3 4 3 3 4 3
## [443] 3 3 3 2 2 1 2 2 3 3 2 3 1 2 4 3 3 3 3 3 1 3 1 3 3 3 1 3 3 3 2 3 1 3 3
## [477] 2 3 1 3 4 3 4 2 4 3 3 3 4 2 3 4 3 1 3 3 4 4 3 3 3 3 4 3 3 2 1 2 3 4 3
## [511] 3 4 2 3 4 4 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 4 2 3 2 2 4 3 3 3
## [545] 3 3 3 3 1 4 4 1 1 4 2 3 3 2 4 3 3 3 3 1 3 2 3 3 3 4 3 3 4 2 4 3 3 2 3
## [579] 3 3 4 4 2 2 3 2 3 2 3 3 3 3 3 3 3 3 2 3 3 3 1 2 3 4 3 3 3 4 3 3 2 3 3
## [613] 4 3 2 3 3 3 3 1 3 4 3 2 3 3 3 4 2 3 3 3 3 2 3 3 3 3 3 2 4 3 2 3 4 3 3
## [647] 4 2 3 3 3 3 3 2 3 1 2 3 4 3 3 3 3 3 3 2 1 3 4 1 3 3 3 3 3 3 2 2 1 3 3
## [681] 2 3 3 3 3 2 3 3 3 1 4 3 2 3 3 3 3 3 4 3 3 3 4 1 3 3 3 2 3 3 3 3 3 2 4
## [715] 3 4 3 1 3 3 3 2 1 4 4 4 4 2 4 2 2 1 2 2 3 3 3 3 1 4 1 1 2 3 2 3 2 2 3
## [749] 3 2 4 3 1 3 3 3 3 1 3 3 1 4 2 3 2 2 3 3 3 3 3 3 3 2 3 3 4 3 3 3 3 2 4
## [783] 3 3 3 3 1 2 1 3 2 4 3 2 3 3 3 3 2 3 3 2 3 4 3 4 3 3 3 3 3 3 4 1 3 2
## [817] 1 3 3 2 3 3 3 1 3 4 3 3 1 3 2 3 1 3 1 3 2 3 2 3 3 3 1 3 3 2 3 3 1 1 3
## [851] 1 4 1 3 3 3 3 1 3 2 3 3 4 1 3 3 3 4 3 2 3 3 3 4 3 1 3 3 3 3 3 2 4 2 3
## [885] 3 4 3 2 2 3 4 3 3 4 3 2 1 3 3 2 3 1 3 3 4 3 3 3 2 3 1 1 1 3 2 4 3 3 2
## [919] 3 3 3 1 2 4 3 3 4 3 3 4 2 1 3 3 3 3 3 3 3 4 2 3 3 3 2 3 3 3 3 3 3 2 1
## [953] 1 3 3 3 3 3 1 3 3 4 3 2 3 3 3 3 1 3 3 3 4 3 3 4 3 2 3 3 2 2 3 3 4 3 1 3
## [987] 3 1 3 3 3 3 3 3 2 3 3 2 3 3 2 2 3 3 3 3 3 2 3 4 3 3 4 3 1 3 3 3 3 3
## [1021] 1 3 3 4 3 3 3 1 3 3 3 3 3 1 4 3 3 2 3 3 3 3 1 4 2 4 2 3 1 2 1 4 1
## [1055] 4 4 3 3 3 3 1 1 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 2 3 1 3 3 3 4 3 3 3 3
## [1089] 3 3 4 3 2 2 3 3 3 1 3 3 1 3 4 3 1 3 4 2 3 3 4 3 3 3 3 2 3 3 1 3 2 3 3
## [1123] 3 3 2 1 1 1 2 3 4 4 4 4 3 3 3 2 3 4 1 2 2 3 2 3 3 1 1 3 3 3 2 3 3 1
## [1157] 1 3 3 4 3 4 3 3 1 3 3 4 4 3 3 3 2 3 1 3 3 1 3 3 4 3 2 3 4 3 3 3 4 3 2
## [1191] 3 3 3 3 1 3 2 1 2 3 3 2 2 2 1 2 3 3 3 3 3 4 3 2 1 4 2 3 3 2 3 3 3 2
## [1225] 3 3 2 4 3 3 3 3 1 1 1 3 1 3 3 1 3 4 1 1 3 2 1 4 3 4 3 3 4 3 2 3 4
## [1259] 2 3 3 2 1 1 3 1 3 4 3 3 3 3 3 2 2 3 3 3 4 4 1 3 2 3 3 3 4 3 3 3 3 3
## [1293] 3 2 3 2 3 3 1 4 2 3 3 3 1 1 1 4 1 3 3 3 2 4 2 2 3 3 3 3 1 3 3 3 1 2
## [1327] 3 4 2 3 3 3 1 3 2 4 1 3 3 4 1 1 2 3 3 4 3 3 3 3 2 2 3 3 3 4 3 3 3 1 3
## [1361] 3 3 3 1 3 2 3 4 3 3 3 3 4 1 3 3 3 3 3 3 3 3 1 1 1 2 3 3 3 4 3 4 4
## [1395] 4 2 3 3 1 3 3 3 2 3 1 1 3 3 3 2 3 3 3 3 4 3 1 2 3 3 3 3 2 2 3 3 2 3
## [1429] 3 4 3 3 1 3 2 3 4 1 2 4 4 1 3 4 2 2 3 3 3 4 1 1 2 4 4 3 3 3 3 1 1 3
## [1463] 1 3 3 3 3 4 1 4 3 3 2 3 3 3 3 2 4 1 1 3 3 1 2 1 3 2 1 3 3 1 3 2 3 3
## [1497] 1 3 3 4 3 3 3 4 3 2 1 2 3 3 3 3 3 3 4 3 4 3 3 3 3 3 3 3 3 1 3 2
## [1531] 3 4 2 1 2 3 3 3 3 2 1 3 4 3 1 3 2 4 3 3 3 3 4 4 4 2 3 3 3 3 3 3 2 3 3
## [1565] 4 3 3 3 1 3 2 3 2 3 2 4 3 3 2 2 3 3 3 3 4 3 3 1 3 3 1 3 2 1 3 2 4 3 3
## [1599] 3 4 4 3 3 4 3 1 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 1 4 4 4 4 2 2 3 3 2 2
## [1633] 3 2 3 3 3 1 3 2 1 1 2 3 3 3 4 2 3 4 3 3 2 4 3 2 3 4 1 3 3 4 1 3 3 3 2
## [1667] 3 3 3 2 2 3 3 3 2 3 3 3 4 2 3 3 3 4 3 1 2 1 3 3 3 3 2 3 4 4 1 3 1 4 3
## [1701] 1 3 3 1 3 1 2 4 3 3 3 3 4 2 3 3 4 2 4 3 4 4 3 1 3 3 3 2 3 3 3 3 3 3 3
## [1735] 3 3 3 3 3 3 3 4 4 3 2 1 3 2 3 3 4 4 2 3 3 3 2 3 3 3 2 3 2 3 3 3 3 3 3
## [1769] 1 3 3 2 3 3 4 3 3 4 3 1 3 3 4 3 2 3 2 4 2 4 1 4 3 4 3 1 3 4 1 1 3 3
## [1803] 3 2 3 2 3 1 2 2 3 4 3 2 1 3 1 4 3 4 2 3 2 3 3 4 3 3 3 3 1 3 2 3 2 4
## [1837] 2 2 2 3 1 3 3 3 2 3 3 2 1 3 3 3 3 2 3 3 4 3 3 3 1 3 4 3 2 2 3 3 3 2 1 3

```

```

## [1871] 1 4 3 3 3 4 1 3 3 3 3 3 3 4 3 2 3 3 2 3 1 1 3 2 1 4 3 3 2 3 3 3 3 3 2
## [1905] 3 3 3 1 3 4 3 4 2 3 3 3 3 3 1 1 3 2 3 2 4 3 2 3 2 3 3 3 3 3 3 3 4 3
## [1939] 3 3 3 2 3 3 3 3 2 3 3 1 2 3 3 4 3 3 3 3 3 4 1 4 3 3 3 3 3 2 3 4 2 2
## [1973] 1 3 3 3 3 3 3 3 3 2 3 3 3 3 3 4 3 3 3 4 2 2 3 2 3 1 2 2 3 3 2 2 3 3
## [2007] 3 1 4 3 3 3 3 3 3 1 3 4 3 3 3 1 3 2 1 3 2 2 1 4 4 3 3 3 3 3 4 3 4 1
## [2041] 4 4 3 3 2 3 3 1 3 3 3 3 3 3 3 2 3 3 1 3 4 2 3 3 3 2 4 3 3 3 3 3 3 3
## [2075] 3 3 3 3 2 3 3 2 3 2 3 1 3 3 1 2 3 3 3 3 1 2 4 1 3 3 3 4 3 3 3 3 4 4 3
## [2109] 1 3 3 2 3 3 2 1 3 2 3 1 3 3 3 3 4 3 3 3 4 3 3 3 3 4 3 1 3 3 2 3 2 1
## [2143] 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 2 1 4 3 3 3 3 1 3 2 3 3 4 3 4 1 3
## [2177] 3 3 4 4 3 4 3 4 3 3 3 3 4 3 3 2 3 3 4 1 3 4 2 3 2 4 2 3 2 3 4 3 3 3
## [2211] 3 3 3 3 2 3 2 1 3 3 3 3 1 3 2 4 4 3 3 3 4 3 3 3 1 3 4 4 3 3 4 3 2 3
## [2245] 2 1 3 3 1 3 3 3 3 3 3 1 3 1 1 4 3 3 2 4 1 2 3 3 1 4 4 3 4 3 2 3 3 3
## [2279] 3 2 3 2 1 3 2 3 1 3 4 3 3 4 3 4 3 3 4 3 3 4 3 3 3 4 1 3 3 1 3 4 3 2
## [2313] 1 3 1 3 3 1 3 3 2 3 3 1 1 3 4 2 2 3 3 3 3 3 3 2 3 2 3 3 3 2 3 1 3 4
## [2347] 2 3 2 2 1 3 3 3 3 3 3 2 3 2 3 3 4 3 3 3 3 3 3 3 3 2 4 2 3 3 3 3 2 3
## [2381] 3 3 3 2 3 1 1 3 3 1 3 1 1 4 1 3 3 3 4 2 1 1 3 3 3 3 4 4 2 4 3 3 1 3
## [2415] 4 3 3 2 3 2 4 4 4 4 3 4 3 3 3 4 3 3 3 3 1 3 4 3 4 3 1 2 2 3 4 4 3
## [2449] 3 3 2 3 3 3 3 3 4 3 3 3 3 2 4 2 3 4 3 3 3 3 1 1 3 4 2 3 4 3 1 3 3 4
## [2483] 3 3 1 3 3 3 3 3 3 2 3 3 2 3 3 4 3 1 3 3 3 3 2 1 3 3 3 2 1 4 3 4 2
## [2517] 4 3 2 3 4 3 3 2 3 4 2 3 2 4 3 3 3 4 4 3 3 3 3 1 3 3 2 3 3 3 4 4 4
## [2551] 3 3 4 3 3 3 3 2 2 3 3 3 3 1 3 4 3 3 3 1 3 3 3 3 3 4 4 3 3 3 2 3 3 3 3
## [2585] 1 3 4 3 1 3 3 3 2 3 3 3 1 2 3 3 4 3 3 3 3 2 2 3 3 3 4 4 3 3 3 3 4 3 4 3
## [2619] 3 4 1 4 3 2 4 1 3 2 3 1 2 3 3 4 3 3 3 3 1 4 3 3 3 1 3 3 4 4 4 3 3 3 1 3
## [2653] 3 3 3 3 3 1 3 1 3 3 3 3 3 3 2 1 3 4 2 3 3 3 3 1 2 3 3 1 4 4 3 3 3 3
## [2687] 2 3 3 4 3 4 3 4 2 3 3 3 2 3 1 3 3 3 3 1 3 1 1 2 4 3 3 3 1 3 2 3 3 3 3
## [2721] 3 3 2 2 1 2 1 3 3 3 3 2 3 3 4 3 2 4 2 1 3 2 2 3 3 2 3 2 1 3 3 2 4 3
## [2755] 3 2 4 2 4 3 3 3 3 3 1 2 3 3 3 3 3 3 4 3 3 3 3 3 1 3 2 3 3 3 1 4
## [2789] 1 2 3 3 3 4 3 3 3 2 3 3 2 4 2 2 3 3 4 2 2 4 3 3 3 3 2 3 3 3 4 4 4 3 3
## [2823] 2 3 2 3 3 2 3 3 4 2 3 3 1 1 3 1 2 2 4 3 1 4 3 1 4 3 4 1 3 4 3 4 4 3
## [2857] 3 3 2 3 2 3 4 3 3 3 3 2 4 3 1 3 3 3 1 3 3 3 2 3 1 3 3 3 3 3 3
## [2891] 3 3 3 3 3 1 4 3 3 3 3 3 4 1 3 3 3 3 3 4 4 3 3 3 3 4 3 2 3 3 3 4 4 4 2
## [2925] 3 3 3 3 3 3 3 3 2 3 3 4 3 3 3 3 2 1 3 3 2 3 3 3 1 3 1 3 3 3 3 3 2 2
## [2959] 3 3 3 4 2 1 3 1 2 2 4 3 1 3 1 2 2 3 3 1 2 1 3 3 1 3 4 3 3 3 3 3 4 3 4
## [2993] 4 2 1 3 2 2 3 4 3 3 1 1 3 4 3 1 3 1 3 1 3 2 1 4 3 4 3 4 3 3 3 3 3 3
## [3027] 3 3 3 4 3 3 3 3 3 3 1 1 3 1 3 3 3 3 3 4 1 3 4 3 3 2 3 3 3 1 3 2 3
## [3061] 3 1 3 3 3 4 3 3 2 1 3 3 3 4 2 3 3 1 1 3 4 4 4 4 1 2 3 3 3 2 3 3 1 3
## [3095] 3 1 3 3 3 2 3 3 3 4 3 3 3 2 4 3 3 3 3 3 1 3 4 4 4 4 3 3 3 3 3 2 3 3 4
## [3129] 3 3 1 3 3 3 4 2 3 3 4 2 1 2 3 3 3 3 3 3 3 3 3 3 4 3 3 3 4 3 4 3
## [3163] 3 2 3 3 2 3 3 1 1 3 3 4 1 2 3 3 3 2 4 3 3 3 3 3 2 2 3 3 3 3 2 4 3
## [3197] 3 1 4 2 3 3 3 3 3 3 4 3 3 3 3 2 3 2 1 3 2 3 3 3 2 3 3 3 3 4 3 3 3 3
## [3231] 4 4 3 1 3 3 3 3 3 3 1 2 3 3 3 3 3 3 4 3 2 3 2 3 3 3 4 3 1 1 4 1 4 3
## [3265] 3 3 3 3 3 3 3 3 4 3 2 2 3 2 3 3 3 4 4 3 3 3 3 3 2 2 1 4 3 3 3 3 4 3
## [3299] 3 3 3 3 3 2 2 3 3 4 3 3 3 3 3 1 1 3 3 3 3 4 3 2 3 3 3 2 3 3 2 4 3 3
## [3333] 3 3 3 3 3 2 2 3 2 3 4 2 1 4 3 4 3 3 3 3 3 4 3 3 3 2 3 2 3 3 3 3 1 3
## [3367] 1 4 3 3 2 4 2 3 3 1 3 4 3 3 3 3 3 3 2 2 3 3 3 3 3 4 3 2 1 3 2 2 2 3
## [3401] 3 4 3 3 4 3 4 3 3 3 3 3 3 4 3 1 3 4 3 2 4 3 3 3 3 2 3 3 1 3 3 3 3
## [3435] 4 3 2 3 4 4 3 3 3 3 1 3 2 3 4 3 3 3 1 3 1 3 3 3 3 3 2 3 3 1 4 3
## [3469] 3 2 1 3 1 2 2 3 3 3 3 3 3 3 2 4 1 3 2 3 3 2 1 4 3 3 3 3 2 3 3 4 1 3
## [3503] 3 4 1 3 3 1 2 3 3 3 3 1 3 2 2 3 3 2 4 3 3 3 4 2 4 3 4 2 2 1 3 3 3 3
## [3537] 3 4 3 1 1 3 4 4 4 3 4 3 3 3 3 4 3 2 2 3 3 2 3 2 3 1 3 3 3 3 3 3 3 3 1
## [3571] 4 2 3 1 3 4 3 2 2 3 3 3 3 2 3 3 1 3 3 3 2 3 3 3 3 2 3 1 3 3 4 3 1 3
## [3605] 2 3 3 3 3 4 2 3 3 3 4 3 3 3 3 3 3 3 2 3 3 3 3 4 2 3 1 3 1 3 3 3 3 4 3
## [3639] 4 4 3 3 2 2 2 3 3 3 3 3 1 3 2 3 4 4 4 3 4 4 3 3 3 3 3 1 1 3 3 2 3 3
## [3673] 3 3 1 4 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 3 2 3 2 3 3 3 3 3 1

```



```

## [5543] 2 2 3 4 3 3 3 1 3 3 3 3 1 2 2 4 3 1 3 3 3 3 1 4 4 4 3 1 3 3 4 3 3
## [5577] 3 3 3 4 4 3 3 3 3 1 1 3 1 3 3 2 3 3 3 2 4 3 1 3 3 3 3 4 3 3 2 1 2 4
## [5611] 3 3 4 1 3 2 3 3 2 3 3 3 2 4 3 3 2 3 1 2 3 3 2 3 3 1 3 1 3 4 3 4 2 4
## [5645] 2 3 3 3 1 3 3 3 3 3 2 3 3 3 3 2 2 3 3 3 4 2 3 2 3 2 3 3 3 3 3 1 3 3
## [5679] 1 3 1 2 3 3 4 3 4 2 3 3 4 3 2 4 1 3 2 3 1 3 1 3 3 3 3 3 2 3 3 2 3
## [5713] 3 3 2 2 4 3 3 3 3 2 1 1 3 3 3 2 3 3 3 3 3 3 1 3 3 3 3 4 3 3 1 1 3
## [5747] 1 2 3 3 3 4 3 3 3 3 3 1 3 3 2 3 1 1 3 3 4 3 3 2 3 1 3 1 3 3 4 3 3
## [5781] 3 3 3 3 3 2 1 3 1 3 4 3 3 3 3 3 3 3 1 3 3 3 3 3 4 3 3 3 3 3 1 4 1
## [5815] 3 3 3 4 2 3 3 3 3 3 3 3 1 2 2 3 3 3 3 3 3 3 4 3 2 3 3 3 3 3 2 3 2
## [5849] 3 3 3 3 3 3 3 3 2 3 2 2 3 1 1 2 4 3 2 4 2 2 2 3 3 3 2 2 3 3 3 3 4
## [5883] 3 3 1 3 3 1 3 2 3 3 2 4 2 3 3 3 3 1 3 4 3 4 4 3 3 3 2 3 2 3 3 3 1 4
## [5917] 3 3 3 1 3 1 3 3 4 2 3 1 3 3 3 1 1 3 3 3 3 3 3 3 4 3 3 3 4 3 3 2 3 3
## [5951] 3 3 1 1 4 4 4 3 3 3 3 4 3 3 3 3 3 1 3 3 3 3 3 2 3 1 3 3 1 2 2 3 3
## [5985] 2 1 2 2 3 1 3 3 1 3 3 2 1 1 1 2 1 3 3 2 4 3 4 3 2 4 3 1 3 1 3 3 1 4
## [6019] 4 3 3 3 3 3 4 2 2 4 3 3 4 3 3 3 3 3 3 3 4 3 3 3 3 1 3 3 3 3 4 2 3 3
## [6053] 4 1 2 3 2 3 3 3 4 3 3 3 3 3 3 3 4 3 3 3 3 3 3 4 3 3 3 3 4 1 3 3 3 1
## [6087] 3 3 3 3 2 3 3 1 4 3 2 3 1 3 3 1 3 3 3 3 3 2 4 3 3 3 3 1 4 3 3 3 3 3
## [6121] 4 3 3 3 3 4 3 3 2 3 1 3 3 4 3 3 2 3 4 2 3 4 4 3 1 3 4 3 3 3 3 2 4 3
## [6155] 4 3 3 1 3 3 1 3 3 3 1 3 4 3 3 2 3 4 1 4 3 3 3 1 3 3 3 3 2 3 3 3 4 3 3
## [6189] 3 3 3 3 1 3 1 3 3 3 3 3 3 3 3 3 2 3 3 3 3 1 3 3 3 3 3 3 2 3 3 3 2 3
## [6223] 2 3 4 3 2 1 3 2 3 1 3 3 2 3 3 1 3 3 3 3 3 2 3 4 3 2 3 3 3 3 2 3 3 3 2
## [6257] 1 2 3 2 2 1 3 3 1 3 2 3 3 3 4 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3
## [6291] 4 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 1 3 2 3 3 3 3 3 1 2 3 3 3 4 4 3 1 3 3
## [6325] 2 3 4 3 3 3 1 3 3 2 2 3 3 3 3 2 3 2 3 2 3 4 3 3 3 3 3 1 3 3 4 1 2 3 3 3
## [6359] 3 1 3 3 3 3 3 4 2 2 3 3 3 2 3 3 4 3 3 3 3 4 3 3 3 2 1 4 4 3 3 3 3 1 3 3 2
## [6393] 2 2 2 3 4 3 3 3 3 2 3 3 3 4 3 4 4 3 3 3 2 2 3 3 4 3 3 2 3 3 2 3 1 3 3
## [6427] 3 3 2 3 3 4 3 4 3 2 1 3 3 3 2 4 4 1 3 1 2 4 3 1 3 3 3 3 3 2 3 3 3 3
## [6461] 3 1 3 2 2 3 3 3 3 1 2 3 4 4 1 3 2 3 2 2 3 3 3 3 3 3 3 4 1 3 3 3 3
## [6495] 1 1 2 3 3 3 2 3 4 1 3 1 3 2 3 3 1 3 3 3 3 3 3 3 1 1 3 3 3 2 3 3 3 3
## [6529] 3 4 4 1 1 3 2 3 2 3 2 4 3 3 3 3 3 2 2 1 3 3 3 2 1 3 1 3 3 2 4 3 1 3
## [6563] 3 3 2 3 2 3 2 3 3 1 4 3 3 3 3 3 3 2 3 3 3 3 3 1 3 3 4 1 3 3 1 2
## [6597] 2 4 3 3 4 3 1 3 3 1 2 3 3 3 1 3 3 3 3 3 3 1 2 3 3 1 2 2 3 3 3 4 2 1
## [6631] 4 4 3 3 4 3 3 3 3 3 4 3 1 3 3 1 3 3 3 3 2 3 2 3 3 1 2 3 3 1 4 3 1 3
## [6665] 3 3 3 3 3 3 3 3 3 2 1 4 3 2 3 4 1 3 4 4 4 3 3 3 3 3 3 2 1 4 3 1
## [6699] 4 3 3 4 3 4 2 3 2 3 3 3 3 3 3 4 3 3 3 3 4 2 3 1 3 3 3 3 2 4 2 2 4
## [6733] 3 1 3 3 3 1 2 3 2 3 2 2 1 1 4 3 3 3 3 3 3 1 3 1 1 3 3 3 1 3 3 3 3 3
## [6767] 4 3 3 3 3 4 3 3 3 3 2 2 3 1 1 3 2 2 2 3 3 2 3 3 3 3 3 1 3 3 3 2 2 2
## [6801] 3 3 1 3 2 2 3 2 3 2 3 3 3 3 3 2 3 1 3 3 4 2 3 2 2 2 3 1 1 3 3 3 3 3
## [6835] 4 1 3 3 3 4 3 2 1 3 3 3 4 3 3 1 4 1 3 2 3 3 2 3 3 2 3 3 3 3 4 4 4 2 1
## [6869] 3 2 4 3 3 1 4 3 3 3 2 2 3 3 3 3 3 2 3 3 3 4 4 4 3 4 3 3 3 3 3 1 3 1
## [6903] 3 3 1 3 2 3 3 3 3 2 3 3 4 2 3 2 3 3 3 1 3 3 1 3 3 1 2 1 1 3 3 3 3 3
## [6937] 3 3 3 3 3 1 3 3 1 3 3 1 1 2 3 3 3 3 3 2 4 3 4 3 3 3 4 3 3 3 2 3 3
## [6971] 3 1 3 4 3 3 2 2 3 3 3 3 3 3 3 3 3 4 3 3 3 1 3 1 2 4 4 1 3 3 4 4 3 3
## [7005] 3 4 3 3 3 3 3 3 4 3 3 2 3 4 2 2 3 3 1 3 3 4 3 3 3 1 3 3 3 1 1 3 3 2 3 3
## [7039] 3 3 4 3 2 2 1 3 1 1 1 3 2 3 1 4 3 3 2 4 3 3 3 4 3 3 3 3 1 3 3 3 3 1
## [7073] 3 3 3 3 2 2 4 3 2 3 3 4 4 3 3 2 3 4 2 3 3 3 3 3 1 2 3 3 3 3 3 3 2 2 3
## [7107] 3 1 1 2 2 3 3 3 3 1 3 3 3 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 1 3
## [7141] 2 3 3 3 3 3 3 3 3 3 2 3 3 3 2 3 3 3 3 4 2 4 3 1 1 3 4 3 3 3 3 3 3 4
## [7175] 3 3 3 3 1 3 3 3 2 3 1 3 3 3 3 2 2 2 3 1 3 3 1 3 3 1 3 3 3 3 3 3 1 3 4
## [7209] 2 3 1 1 4 1 3 3 4 3 3 3 3 3 1 3 3 1 1 2 3 2 3 3 3 2 3 2 2 2 3 4 3 3
## [7243] 3 4 3 3 1 1 4 3 3 4 3 3 3 3 3 3 3 1 3 2 3 4 3 3 3 2 3 1 3 3 3 3 3 3
## [7277] 3 3 3 3 3 3 3 4 4 3 1 3 3 3 3 3 4 3 3 3 3 1 3 3 4 3 3 3 2 2 1 3 2 3 4
## [7311] 3 3 3 1 3 3 3 3 3 3 3 1 4 2 4 3 3 3 3 3 4 1 3 3 2 3 3 3 1 1 3 3 3 3 3
## [7345] 2 3 3 4 3 3 3 3 1 3 3 3 3 3 4 1 1 3 1 3 3 2 2 3 3 3 3 2 3 3 3 2 3 3 3 1

```

```

## [7379] 3 4 1 1 3 3 3 3 4 3 3 3 3 4 1 3 3 1 1 1 4 1 1 3 3 3 3 3 3 1 1 3 2 3
## [7413] 3 3 2 3 3 1 3 1 3 3 3 3 2 3 2 4 3 3 3 2 4 3 1 2 4 2 3 3 3 3 3 3 3 3
## [7447] 3 3 2 3 1 3 3 3 3 3 2 2 1 3 3 1 1 1 3 2 4 1 3 3 3 3 2 1 3 1 2 2 4 3
## [7481] 3 2 2 2 2 3 4 3 2 3 3 3 3 1 3 3 1 3 3 4 2 3 1 3 3 3 2 3 3 3 3 3 3 3 4
## [7515] 3 4 1 4 2 4 3 3 3 3 1 3 3 3 3 2 2 3 3 3 4 3 3 2 3 3 3 3 1 3 3 2 3 3
## [7549] 3 3 3 3 3 3 3 1 3 4 3 3 3 3 2 3 4 3 3 3 3 2 2 3 2 3 2 3 3 3 1 3 1 4
## [7583] 1 3 2 3 3 4 1 1 3 3 3 3 1 3 3 4 1 2 3 3 3 2 1 3 3 3 3 4 3 1 1 3 4 4
## [7617] 3 3 2 3 3 2 1 4 3 2 3 3 2 3 3 2 4 1 3 3 3 3 4 3 3 3 2 2 3 3 3 1 1 3 3
## [7651] 1 3 3 2 3 3 3 3 4 4 3 3 3 3 3 3 1 3 3 2 3 3 3 2 3 3 3 3 2 3 3 1 3 2
## [7685] 1 1 3 3 3 4 1 4 3 4 3 3 3 2 2 3 3 1 3 4 4 3 3 2 3 2 3 2 3 2 3 3 3
## [7719] 2 4 3 4 3 3 3 1 1 2 3 3 3 3 3 3 3 4 3 3 1 3 3 3 3 3 2 1 3 3 3 3
## [7753] 3 3 1 4 3 3 2 2 2 2 3 3 3 3 4 3 2 2 1 4 3 2 2 3 3 4 3 3 2 3 1 1 3 3
## [7787] 3 3 3 3 3 3 2 3 2 3 3 3 3 3 4 3 3 3 3 4 3 3 3 3 3 3 3 1 3 1 3 1
## [7821] 3 3 3 2 3 3 3 3 4 3 3 3 2 3 3 3 3 3 1 3 4 3 1 3 3 3 3 3 3 3 3 3 3 3
## [7855] 3 4 3 3 3 2 3 1 3 3 2 3 3 3 4 3 1 1 2 3 3 3 3 3 3 2 3 3 3 3 1 4 3
## [7889] 4 3 2 1 3 3 4 4 3 3 3 3 4 3 4 3 1 3 4 3 3 3 3 4 2 3 3 3 2 3 3 1 2
## [7923] 3 3 3 3 3 3 3 3 2 2 1 3 3 2 3 2 2 3 3 3 3 3 3 1 3 3 3 2 1 3 4 3
## [7957] 3 1 2 3 3 3 2

consumo_clusters$centers

##   sector_viaje sector_rest_bar sector_salud sector_supermcd
## 1   -0.4420956      0.3088986  -0.04585913   -0.352620887
## 2   -0.9970553      -0.1702628     0.77784999    0.613090453
## 3    0.3637802      0.1784416     0.19417443   -0.003774304
## 4   -0.2645054      -0.9655032   -1.79213323   -0.354371098
##   sector_electronico sector_tienda_depart sector_vehiculo sector_finanzas
## 1    -0.14293164      0.1711458   -0.1086413     0.31103685
## 2     0.28874070      -1.1882088     0.4854009   -0.30914054
## 3    -0.05154080      0.2664812   -0.1160825     0.06232457
## 4     0.05705199      -0.1249438     0.1216266   -0.23924216
##   sector_eduacion sector_clubes sector_entretenimiento
## 1    -0.73788859     -0.74769298     -1.8841245
## 2     0.09623506      0.16823695     0.1718154
## 3     0.08579642      0.06493823     0.2524317
## 4     0.13701128      0.16724910     0.2691178

consumo_clusters$size

## [1] 906 1130 4926 1001

#####
# Validación
#####

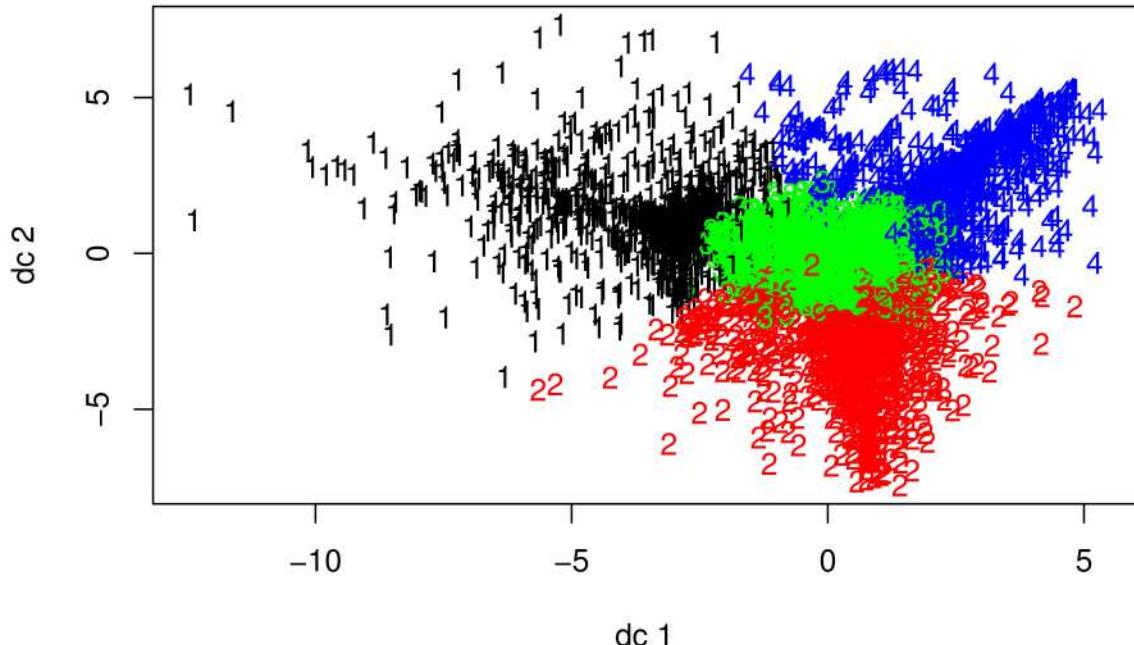
# Veamos la medio de los coeficientes de silueta
sil_4 <- silhouette(consumo_clusters$cluster, dist(data_escale))

# Si existen clústers con media menor o igual que cero, la clasificación es mala, cc, si es positivo, e
aggregate(sil_4[,3], list(sil_4[,1]), mean)

##   Group.1      x
## 1      1 0.02747813
## 2      2 0.04869697
## 3      3 0.17776486
## 4      4 0.17755005

```

```
# Graficamos
plotcluster(Data_Consumo_Final[,5:15], consumo_clusters$cluster)
```



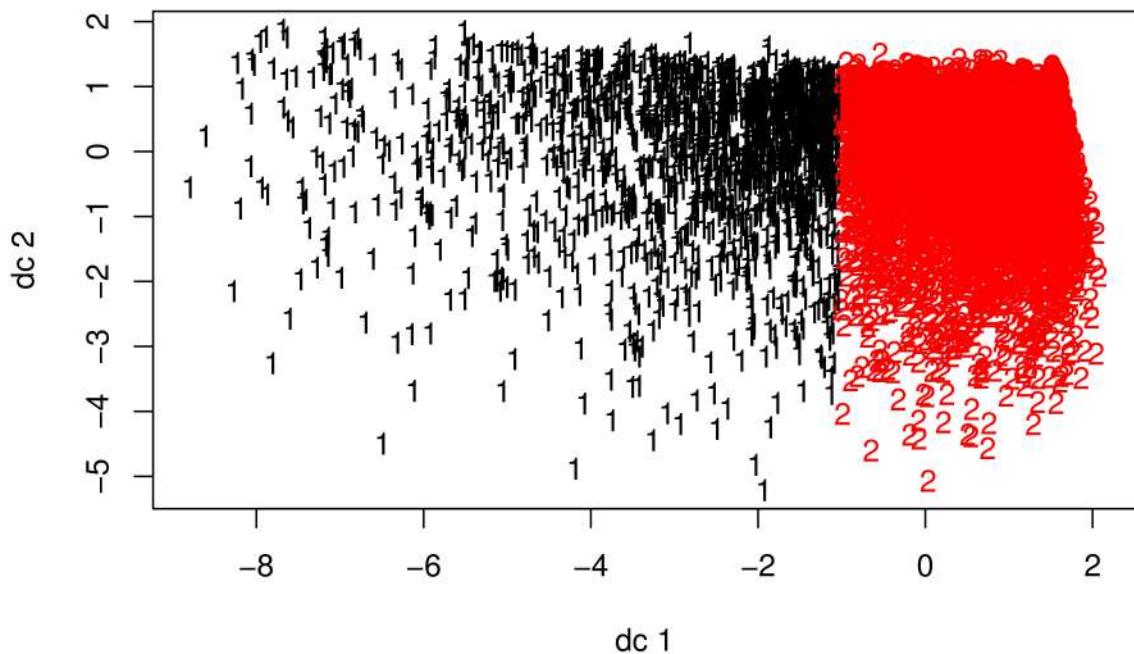
```
# Probamos con 2 segmentos
consumo_clusters_2 <- kmeans(data_escala, 2)
consumo_clusters_2$iter # Ver numero de iteraciones
```

```
## [1] 1
#consumo_clusters_2$cluster
consumo_clusters_2$centers

##   sector_viaje sector_rest_bar sector_salud sector_supermcd
## 1   -0.8157001    1.1679609   -0.22631684    -0.6515624
## 2    0.2186024   -0.3130061    0.06065147    0.1746146
##   sector_electronico sector_tienda_depart sector_vehiculo sector_finanzas
## 1    -0.011932291     0.07650914    0.07580329   -0.009174856
## 2     0.003197778    -0.02050396   -0.02031480    0.002458803
##   sector_eduacion sector_clubes sector_entretenimiento
## 1     0.09119252    0.031771193     0.06382100
## 2    -0.02443902   -0.008514477    -0.01710362

consumo_clusters_2$size

## [1] 1683 6280
plotcluster(Data_Consumo_Final[,5:15], consumo_clusters_2$cluster)
```



```

#Validación
sil_2 <- silhouette(consumo_clusters_2$cluster, dist(data_escale))
aggregate(sil_2[,3], list(sil_2[,1]), mean)

##  Group.1      x
## 1      1 0.007569558
## 2      2 0.190506194

# Probamos con 8 segmentos
consumo_clusters_8 <- kmeans(data_escale, 8)
consumo_clusters_8$iter # Ver numero de iteraciones

## [1] 5

#consumo_clusters_8$cluster
consumo_clusters_8$centers

##   sector_viaje sector_rest_bar sector_salud sector_supermcd
## 1 -0.49966531    1.53426958  -0.1318712   -1.01675744
## 2 -0.21839205   -1.06180801  -1.9962159   -0.37818734
## 3 -0.34175727    0.12118348   0.1749488   -0.45998279
## 4 -1.25088736   -0.90498882   0.9078867   -0.09074017
## 5  0.04160436    0.80799570  -0.5304597   1.87794580
## 6 -0.34919337   -0.28172335   0.2403790   0.13752614
## 7 -0.54330306   -0.40313497   0.5303760  -0.07808660
## 8  0.60388603   -0.04991507   0.2752384  -0.06020317
##   sector_electronico sector_tienda_depart sector_vehiculo sector_finanzas
## 1      -0.22434819        -0.06488715      0.314023243     -0.19773079

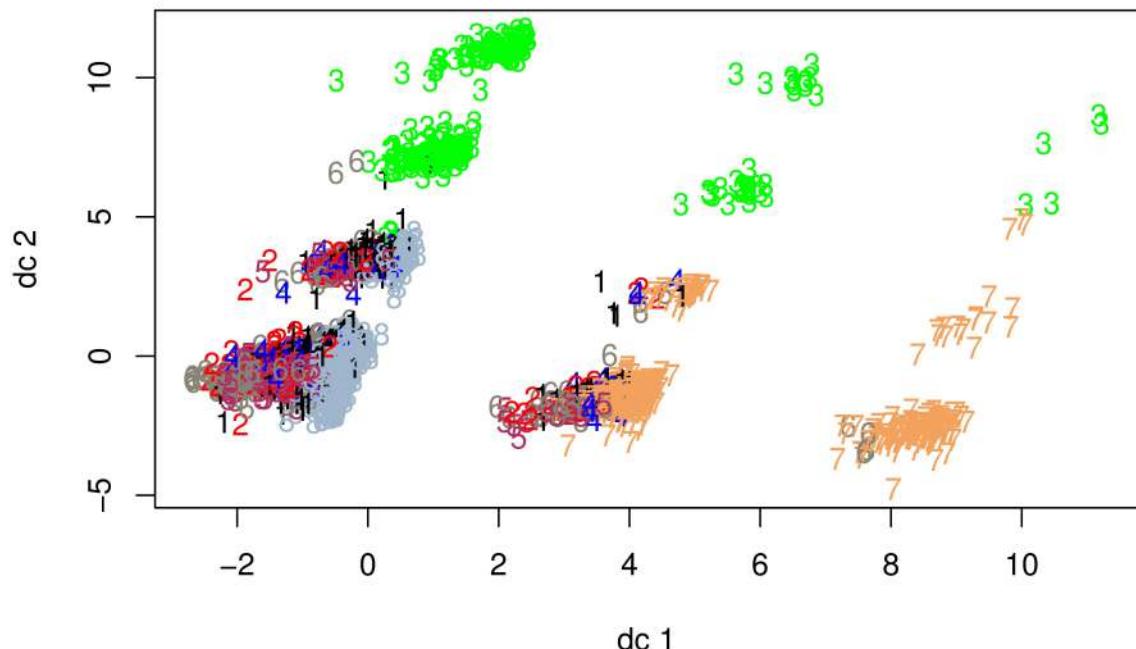
```

```

## 2      -0.01783655      -0.37764468      0.153581606      -0.31874698
## 3      -0.05942601      0.24157459      0.557101732      1.06081484
## 4      -1.35299378      0.57696129      -0.635365727      -0.31574845
## 5      -0.21158097      0.39697672      -0.365196466      -0.07031122
## 6      0.23868655      -0.54837512      0.196470639      2.61297585
## 7      2.22202305      1.13188401      0.090748463      -0.45793376
## 8      -0.13333859      -0.28735265      0.006873713      -0.16326887
##   sector_educacion sector_clubes sector_entretenimiento
## 1      0.385459544     0.02577782      0.221661723
## 2      0.108510599     0.07786014      0.052206138
## 3      -3.905539369     2.04596851      0.583153917
## 4      0.162960513     -0.07523031      0.107362191
## 5      -0.051372900     -0.03394107      0.031807596
## 6      0.933813313     -0.01351401      0.006476695
## 7      0.094365260     -0.10861596      -0.090167583
## 8      -0.005980771     -0.12968145      -0.122180814
consumo_clusters_8$size

## [1] 861 767 254 658 817 496 728 3382
plotcluster(Data_Consumo_Final[,5:15], consumo_clusters_8$cluster)

```



```

#Validación
sil_8 <- silhouette(consumo_clusters_8$cluster, dist(data_escala))
aggregate(sil_8[,3], list(sil_8[,1]), mean)

##   Group.1      x

```

```

## 1      1 0.08503155
## 2      2 0.19520311
## 3      3 0.25503057
## 4      4 0.12624931
## 5      5 0.17789263
## 6      6 0.09233779
## 7      7 0.11711333
## 8      8 0.19259311

```

Clustering JERÁRQUICO

En el blog Rpubs, un post de Miguel Jiménez describe esta técnica de manera muy precisa:

“La agrupación es una técnica para agrupar puntos de datos similares en un grupo y separar las diferentes observaciones en diferentes grupos o grupos. En Hierarchical Clustering, los clusters se crean de manera que tengan un orden predeterminado, es decir, una jerarquía.”

Para mayor alcance de esta técnica, remito la fuente: <http://rpubs.com/mjimcua/clustering-jerarquico-en-r>

Usando la base normalizada y sin outliers (data_escale), aplicaremos la segmentación jerárquica. Debido al coste computacional alto obviamos el paso de las comparaciones entre métodos de aglomeración jerárquica (complete, average, ward.D2, single, etc.), este paso ya se realizó, y el mejor método resultó ward.D. El método que se usó para la comparación se deja comentado.

Vemos en el dendograma, que podemos cortar la gráfica en 4 y 5 grupos. Decidimos usar 4 grupos, siguiendo la validación en el ejercicio anterior.

```

library(factoextra)

#####
# Comparación de métodos
#####
#library(purrr)

#m <- c( "average", "single", "complete", "ward")
#names(m) <- c( "average", "single", "complete", "ward")

## `Función de coeficiente de aglomeración (el más alto indica el mejor método)
#ac <- function(x) {
#  agnes(df, method = x)$ac
#}
#map_dbl(m, ac)
#####

# Creamos una copia de la bd donde tenemos las columnas estandarizadas
df = data.frame(data_escale)

# Creamos dos funciones:
### Aplicamos clúster jerárquico, bajo el método ward.D
hclustfunc <- function(x) hclust(x, method="ward.D")
### Creamos una función de distancia para la matriz de disimilaridad
distfunc <- function(x) as.dist((1-cor(t(x)))/2)

# Aplicamos las funciones creadas
d <- distfunc(df)

```

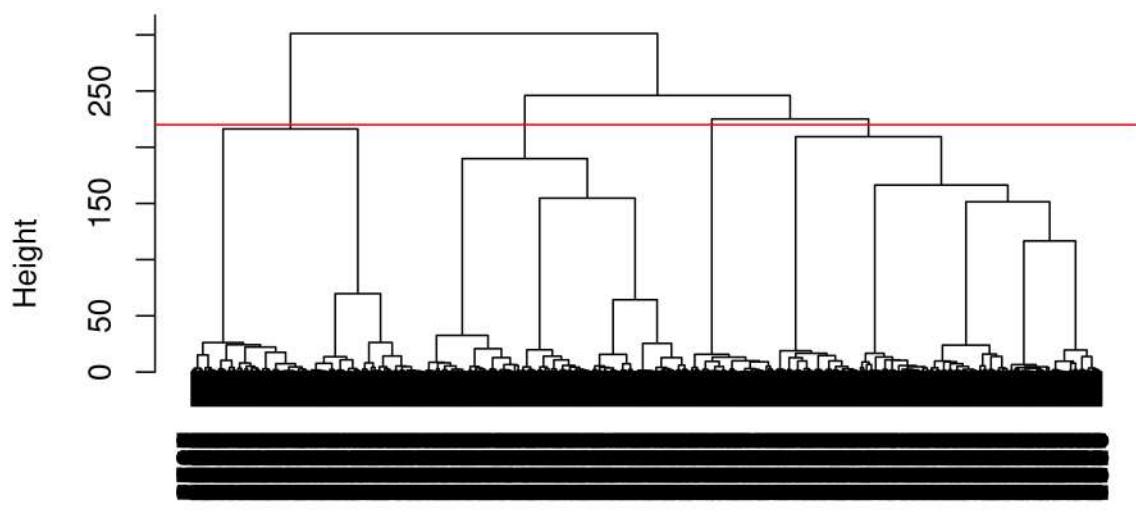
```

fit <- hclustfunc(d)

# Graficamos el dendograma y la intersección que refleja las 4 aglomeraciones
plot(fit) +
  abline(h=220, col="red")

```

Cluster Dendrogram



X
hclust (*, "ward.D")

```
## integer(0)
```

Probemos con 5 grupos. Veremos que existen pocas diferencias con 4 segmentos

```

# Formamos los 5 grupos
clusteres_consumo <- cutree(tree = fit, k = 5)

#####
# Validación
#####
# Mediante el mismo coeficiente usado en kmeans (silueta)
sil_jerar <- silhouette(clusteres_consumo, dist(data_escala))
aggregate(sil_jerar[,3], list(sil_jerar[,1]), mean)

```

```

##   Group.1      x
## 1      1  0.21816884
## 2      2 -0.18030333
## 3      3  0.27766073
## 4      4  0.04238602
## 5      5  0.18919680

```

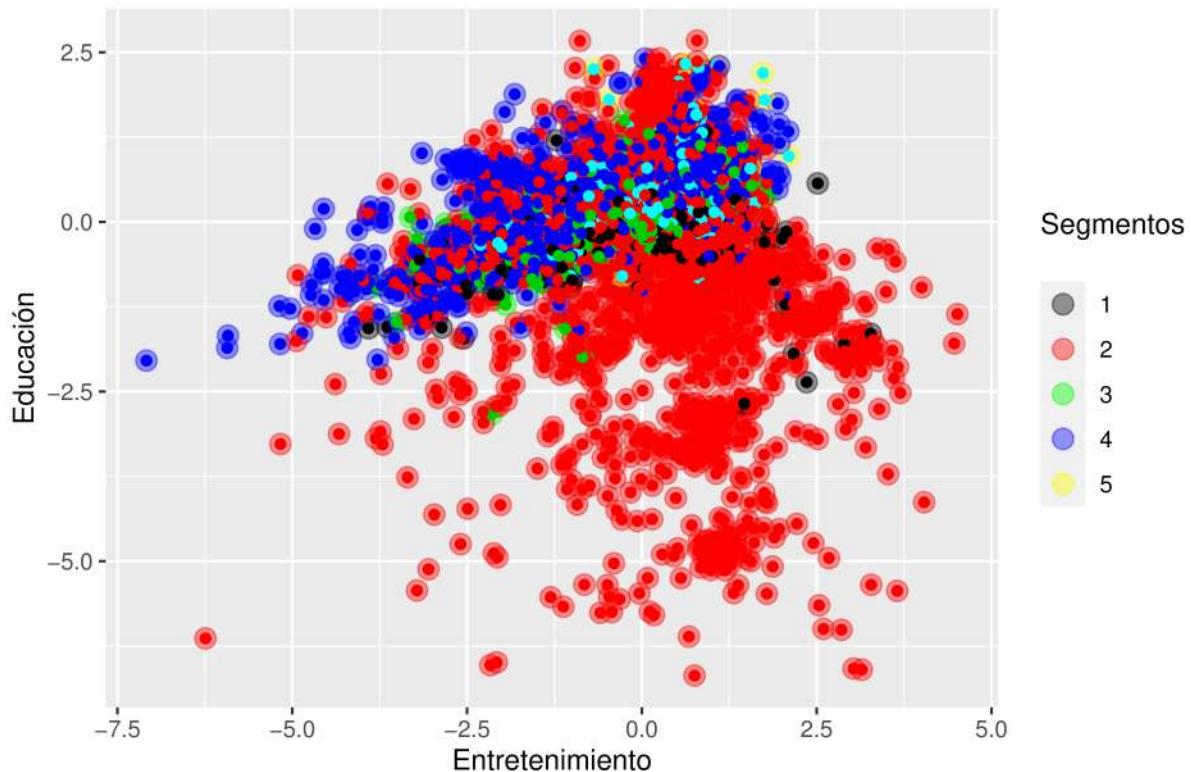
```

# Traemos los grupos al dataframe df, que es con el que realizamos el análisis
df=cbind(data_escala,data.frame(clusteres_consumo))

# Graficamos algunas variables y las coloreamos según los segmentos obtenidos
ggplot(df, aes(df$sector_entretenimiento, df$sector_eduacion, color = as.factor(df$clusteres_consumo)))
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +
  scale_color_manual(values = c('black', 'red', 'green', "blue", "yellow"))+
  ggtitle("Entretenimiento vs Educación") +
  xlab("Entretenimiento") + ylab("Educación")+
  labs(color = "Segmentos\n")

```

Entretenimiento vs Educación

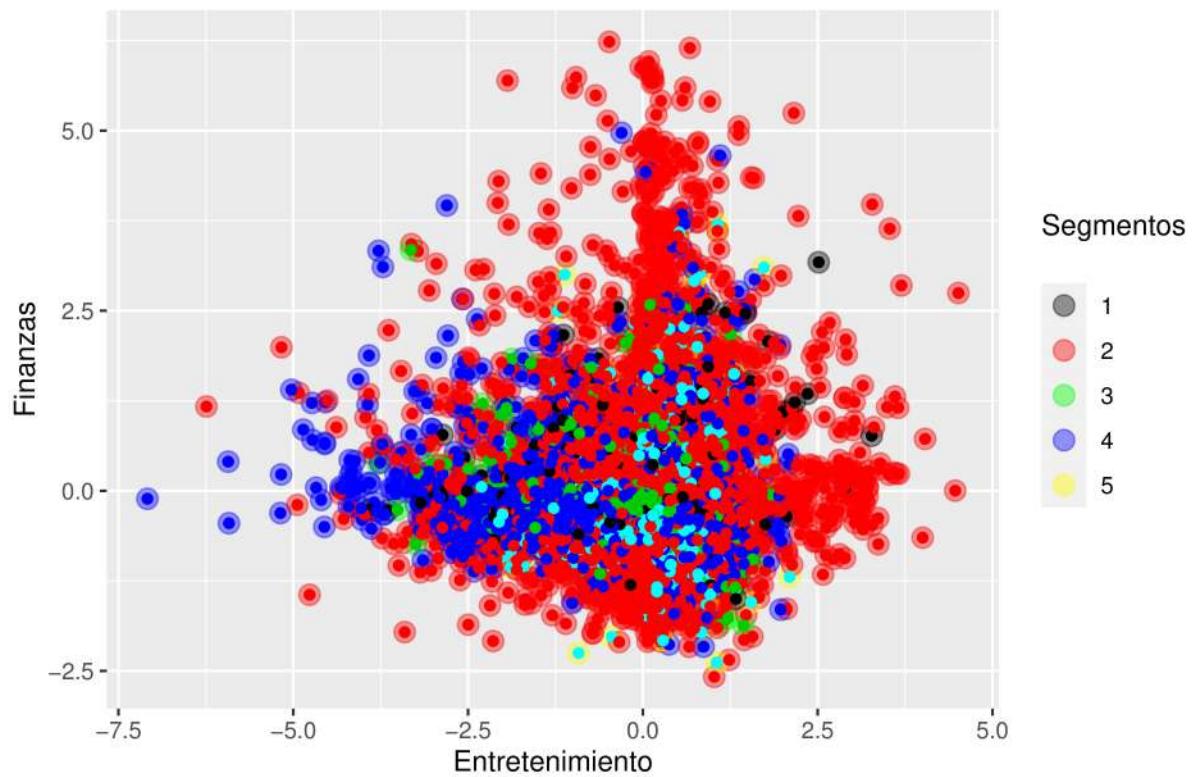


```

ggplot(df, aes(df$sector_entretenimiento, df$sector_finanzas, color = as.factor(df$clusteres_consumo)))
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +
  scale_color_manual(values = c('black', 'red', 'green', "blue", "yellow"))+
  ggtitle("Entretenimiento vs Finanzas") +
  xlab("Entretenimiento") + ylab("Finanzas")+
  labs(color = "Segmentos\n")

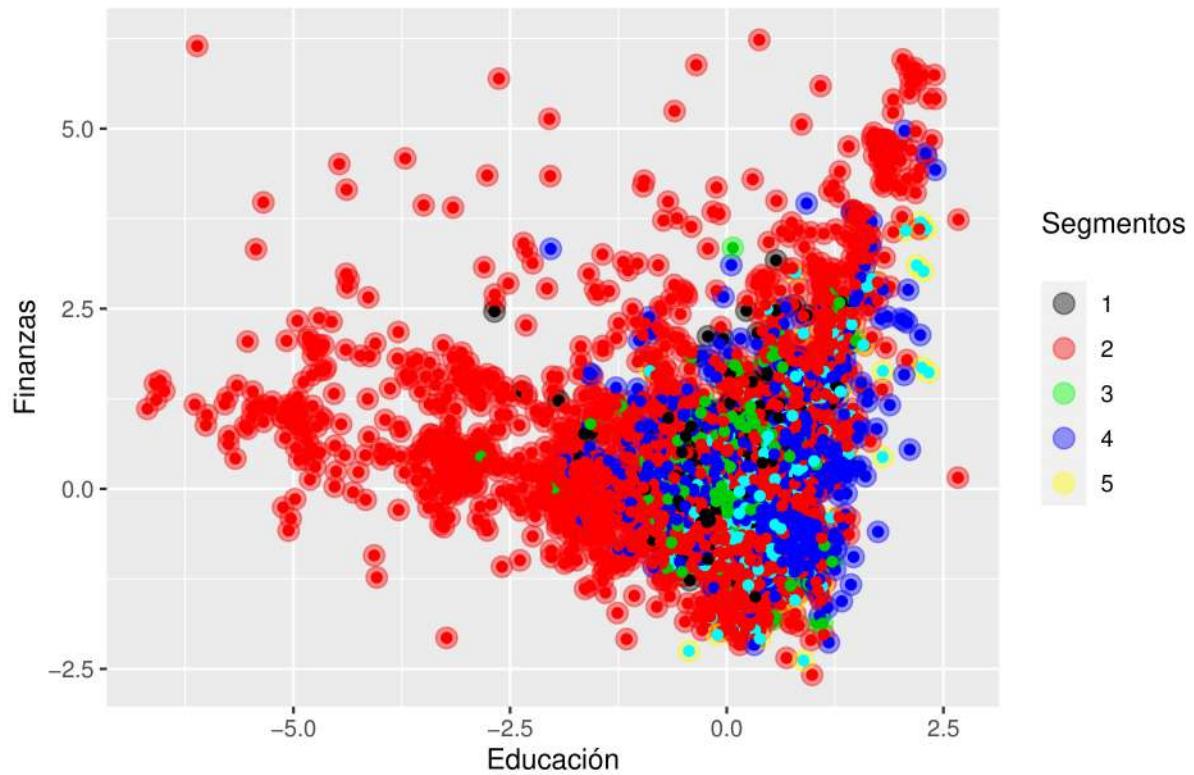
```

Entretenimiento vs Finanzas



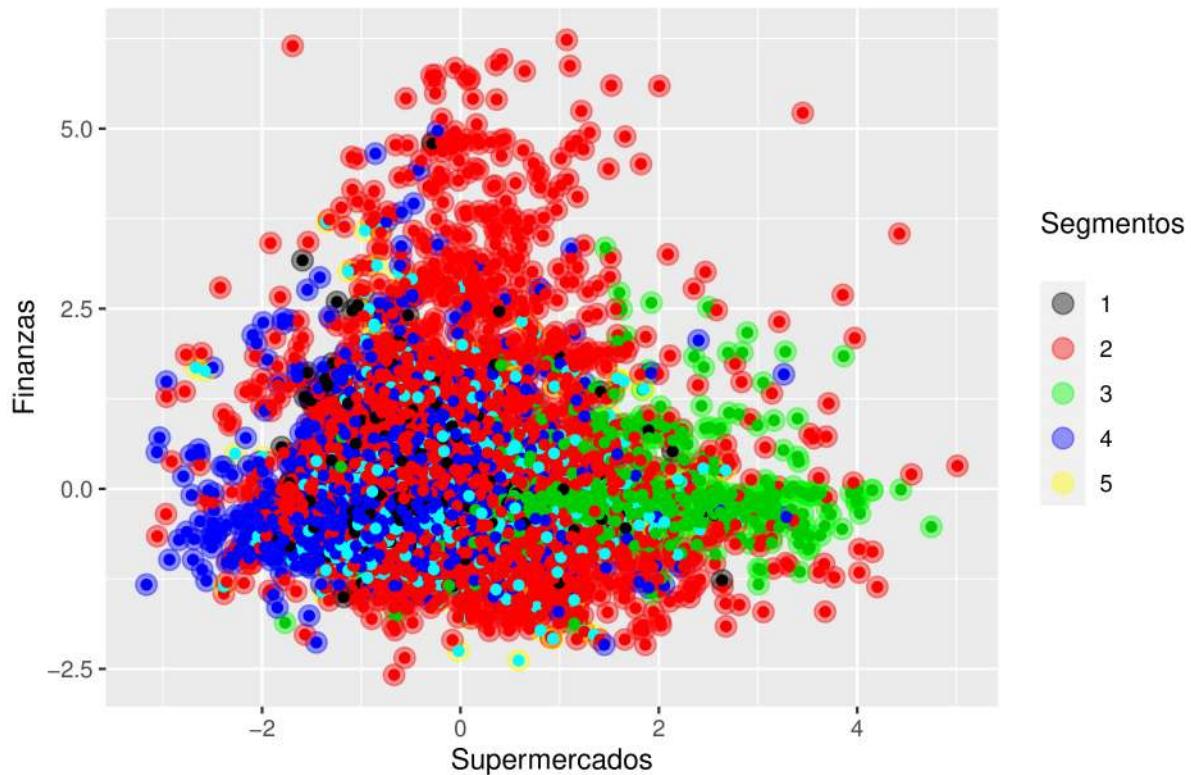
```
ggplot(df, aes(df$sector_eduacion, df$sector_finanzas, color = as.factor(df$clusteres_consumo))) +  
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +  
  scale_color_manual(values = c('black', 'red', 'green', "blue", "yellow"))+  
  ggtitle("Educación vs Finanzas") +  
  xlab("Educación") + ylab("Finanzas") +  
  labs(color = "Segmentos\\n")
```

Educación vs Finanzas



```
ggplot(df, aes(df$sector_supermcd, df$sector_finanzas, color = as.factor(df$clusteres_consumo))) +  
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +  
  scale_color_manual(values = c('black', 'red', 'green', "blue", "yellow"))+  
  ggtitle("Supermercados vs Finanzas") +  
  xlab("Supermercados") + ylab("Finanzas") +  
  labs(color = "Segmentos\n")
```

Supermercados vs Finanzas



Ahora, cortamos las aglomeraciones en 4 y graficamos el comportamiento de estos grupos en planos bidimensionales de los sectores con mayor cantidad de rubros absorbidos en el paso del pca (educación, finanzas, supermercado y entretenimiento).

Si bien no hay grupos muy marcados, en todos los gráficos vemos que el segmento dos y 4 son predominantes.

```
# Formamos los 4 grupos
clusteres_consumo <- cutree(tree = fit, k = 4)

#####
# Validación
#####
# Mediante el mismo coeficiente usado en kmeans (silueta)
sil_jerar <- silhouette(clusteres_consumo, dist(data_escala))
aggregate(sil_jerar[,3], list(sil_jerar[,1]), mean)

##   Group.1      x
## 1      1  0.24657669
## 2      2 -0.15701444
## 3      3  0.14204118
## 4      4  0.08694901

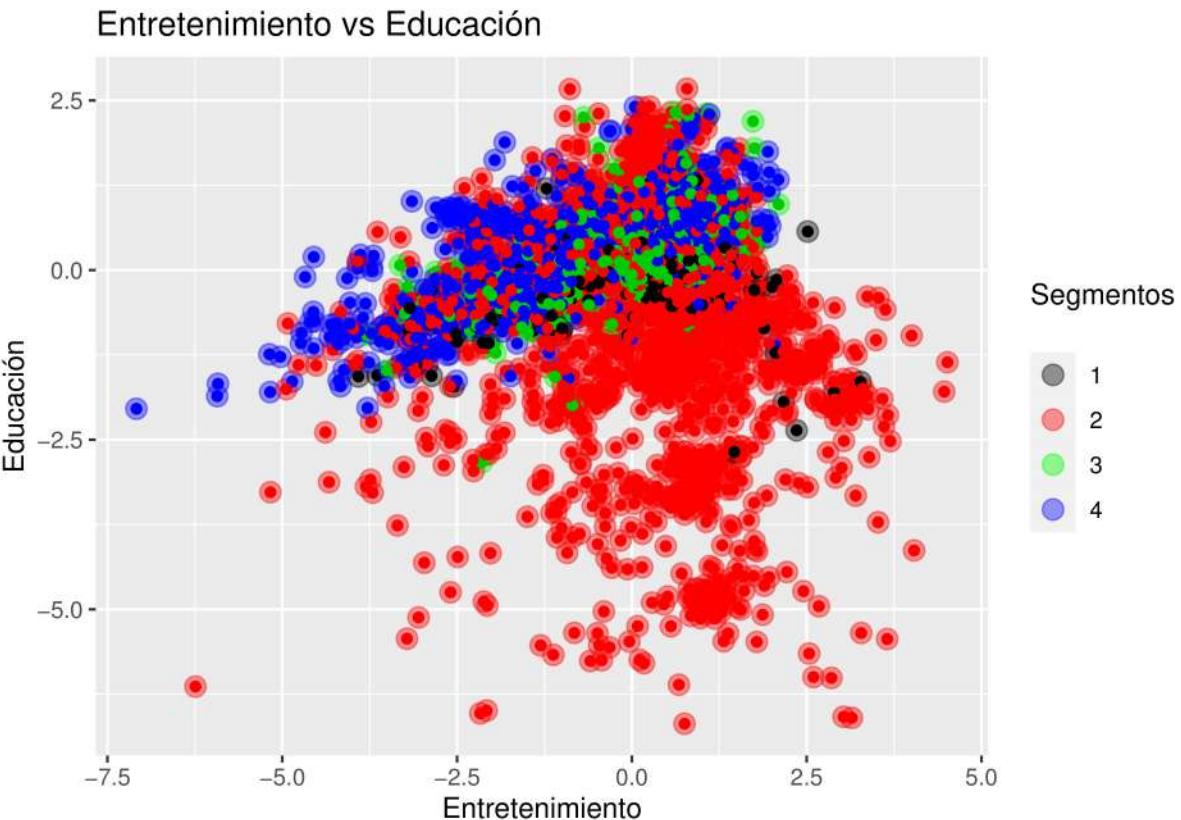
# Traemos los grupos al dataframe df, que es con el que realizamos el análisis
df=cbind(data_escala,data.frame(clusteres_consumo))

# Graficamos algunas variables y las coloreamos según los segmentos obtenidos
ggplot(df, aes(df$sector_entretenimiento, df$sector_eduacion, color = as.factor(df$clusteres_consumo)))
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +
```

```

scale_color_manual(values = c('black', 'red', 'green', "blue"))+
ggtitle("Entretenimiento vs Educación") +
xlab("Entretenimiento") + ylab("Educación")+
labs(color = "Segmentos\n")

```

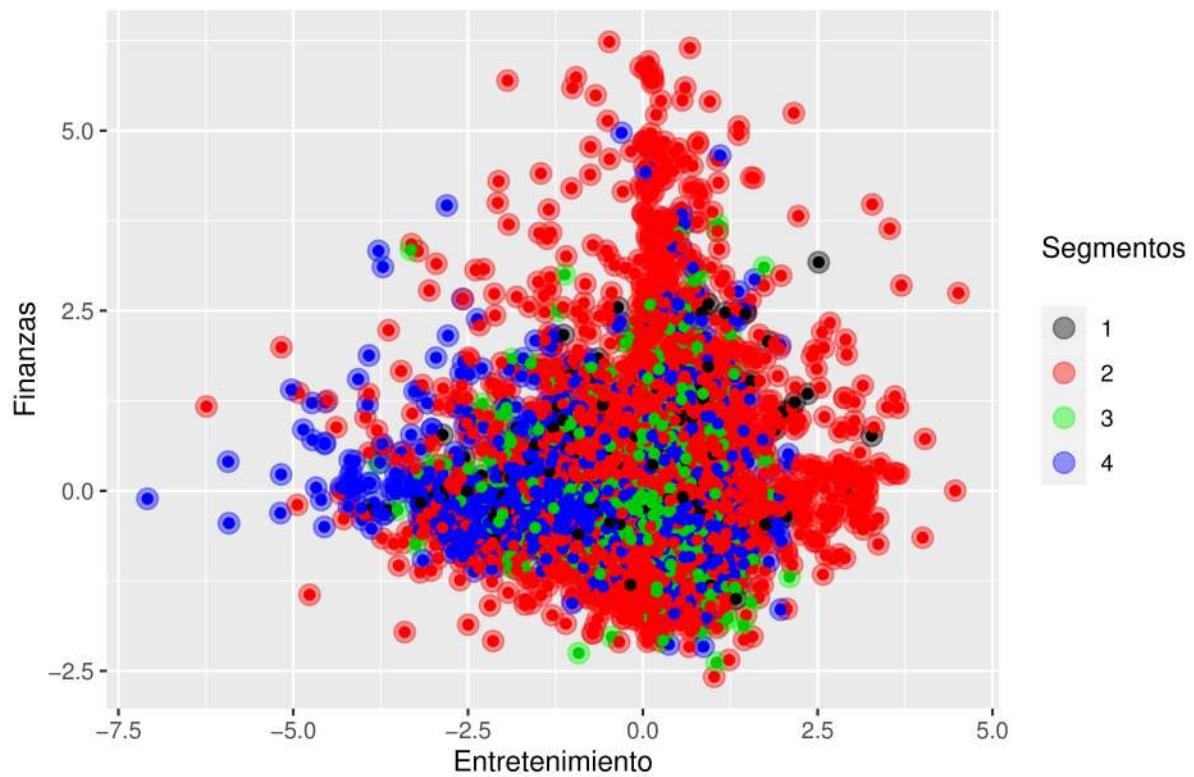


```

ggplot(df, aes(df$sector_entretenimiento, df$sector_finanzas, color = as.factor(df$clusteres_consumo)))
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +
  scale_color_manual(values = c('black', 'red', 'green', "blue"))+
  ggtitle("Entretenimiento vs Finanzas") +
  xlab("Entretenimiento") + ylab("Finanzas")+
  labs(color = "Segmentos\n")

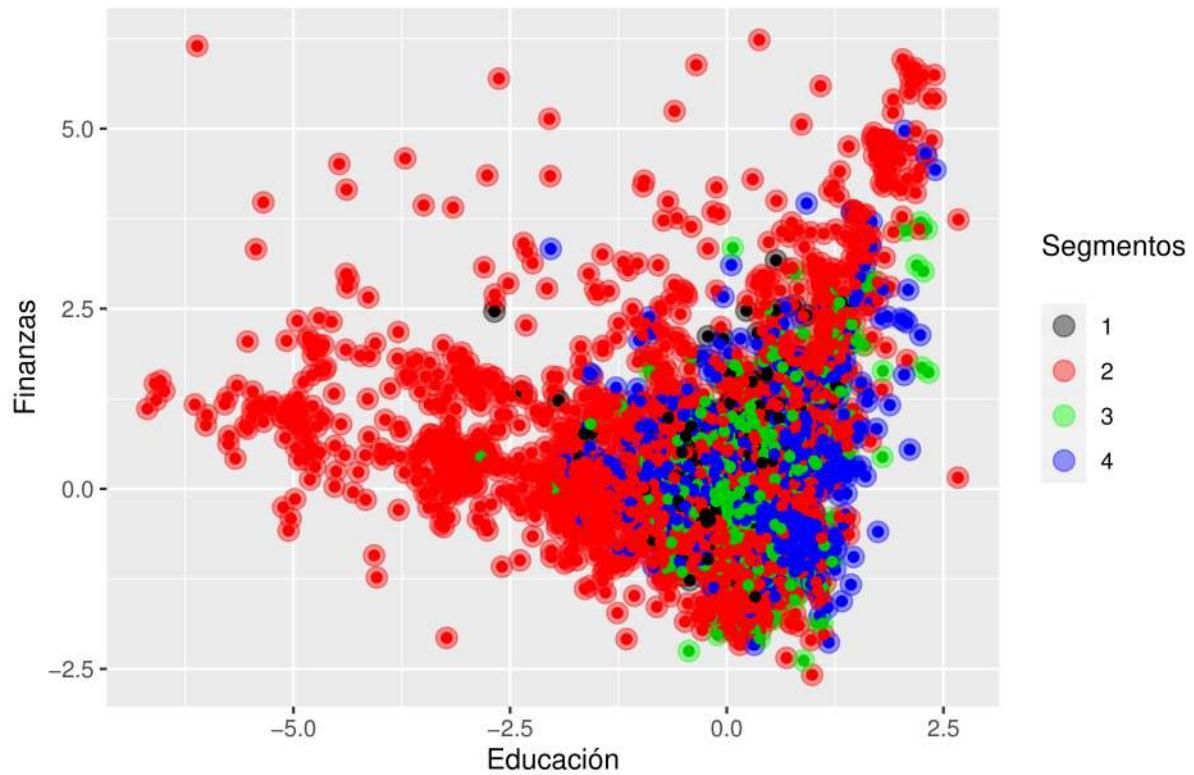
```

Entretenimiento vs Finanzas



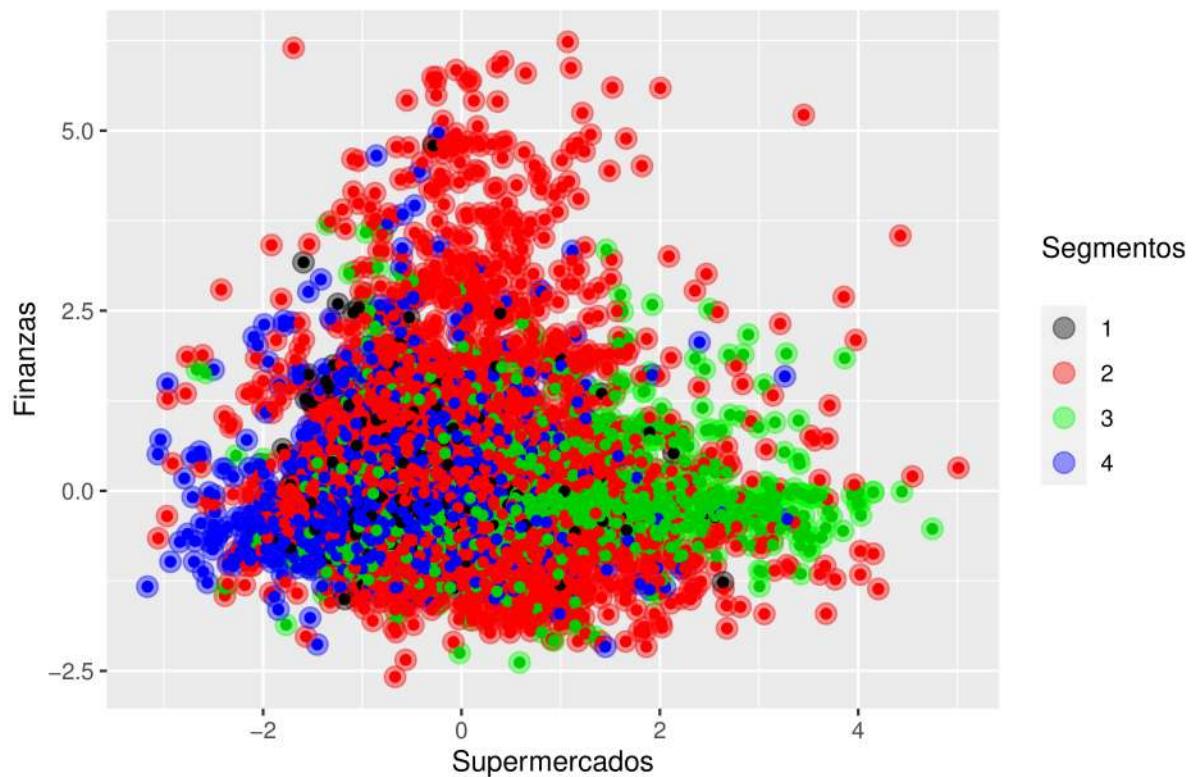
```
ggplot(df, aes(df$sector_eduacion, df$sector_finanzas, color = as.factor(df$clusteres_consumo))) +  
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +  
  scale_color_manual(values = c('black', 'red', 'green', "blue")) +  
  ggtitle("Educación vs Finanzas") +  
  xlab("Educación") + ylab("Finanzas") +  
  labs(color = "Segmentos\\n")
```

Educación vs Finanzas



```
ggplot(df, aes(df$sector_supermcd, df$sector_finanzas, color = as.factor(df$clusteres_consumo))) +  
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +  
  scale_color_manual(values = c('black', 'red', 'green', "blue"))+  
  ggtitle("Supermercados vs Finanzas") +  
  xlab("Supermercados") + ylab("Finanzas") +  
  labs(color = "Segmentos\\n")
```

Supermercados vs Finanzas



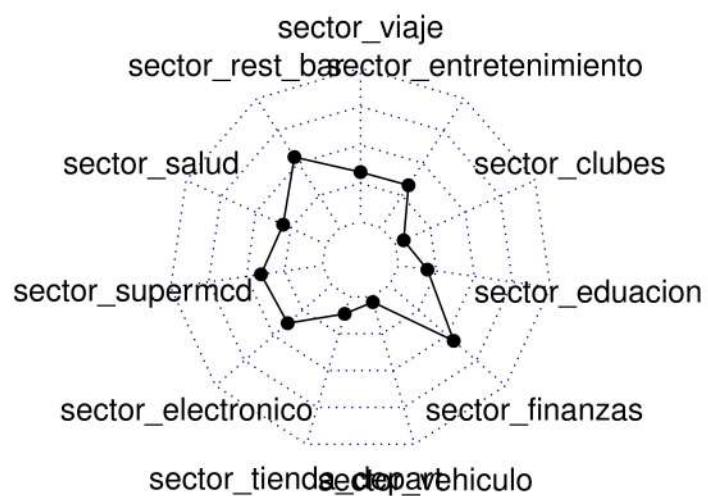
Obtenemos la influencia de cada sector por cada segmento:

```
Data_Segmentada = cbind(Data_Consumo_Final, data.frame(clusteres_consumo))

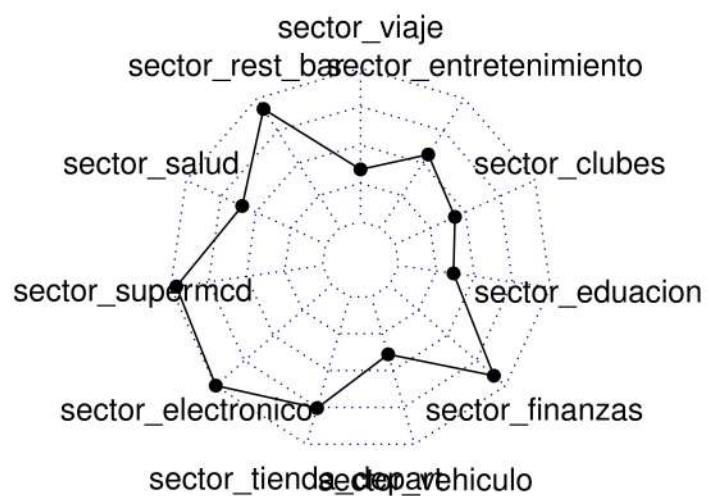
# le damos forma para trabajarla
Data_Segmentada_Unida=tidyr::gather(data=Data_Segmentada, key = desc_segmento, value = valor_rubro, 5:15)

#install.packages("fmsb")
library(fmsb)

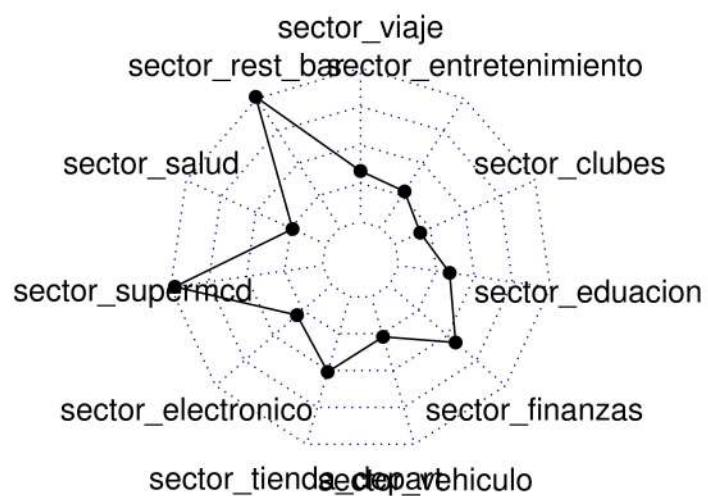
# Usamos el gráfico de radar para ver las preferencias de cada segmento.
radar_data = aggregate(Data_Segmentada[, 5:15], list(Data_Segmentada$cluster), max)
radar_data= data.frame(radar_data)
radar_data_1 = data.frame(radar_data[1,2:12])
data_1 = rbind(rep(4,11) , rep(0,11) , radar_data_1)
radarchart(data_1)
```



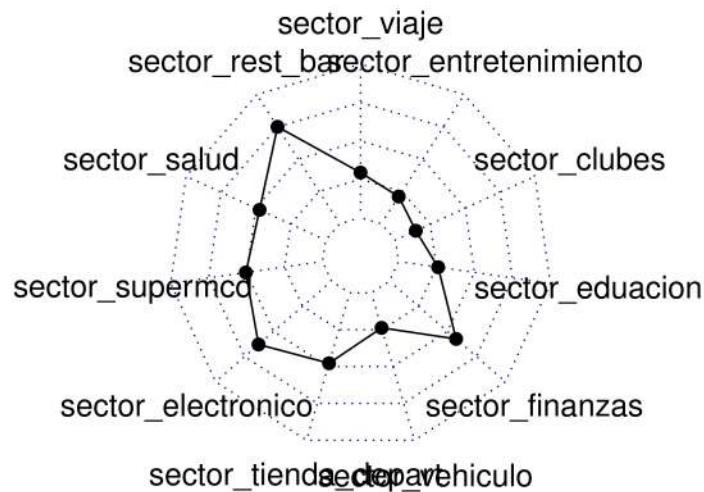
```
radar_data_2 = data.frame(radar_data[2,2:12])
data_2 = rbind(rep(3.2,11) , rep(0,11) , radar_data_2)
radarchart(data_2)
```



```
radar_data_3 = data.frame(radar_data[3,2:12])
data_3 = rbind(rep(3,11) , rep(0,11) , radar_data_3)
radarchart(data_3)
```



```
radar_data_4 = data.frame(radar_data[4,2:12])
data_4 = rbind(rep(4,11) , rep(0,11) , radar_data_4)
radarchart(data_4)
```



Ahora, veremos los descriptivos.

```
#install.packages("viridis")
library(viridis)

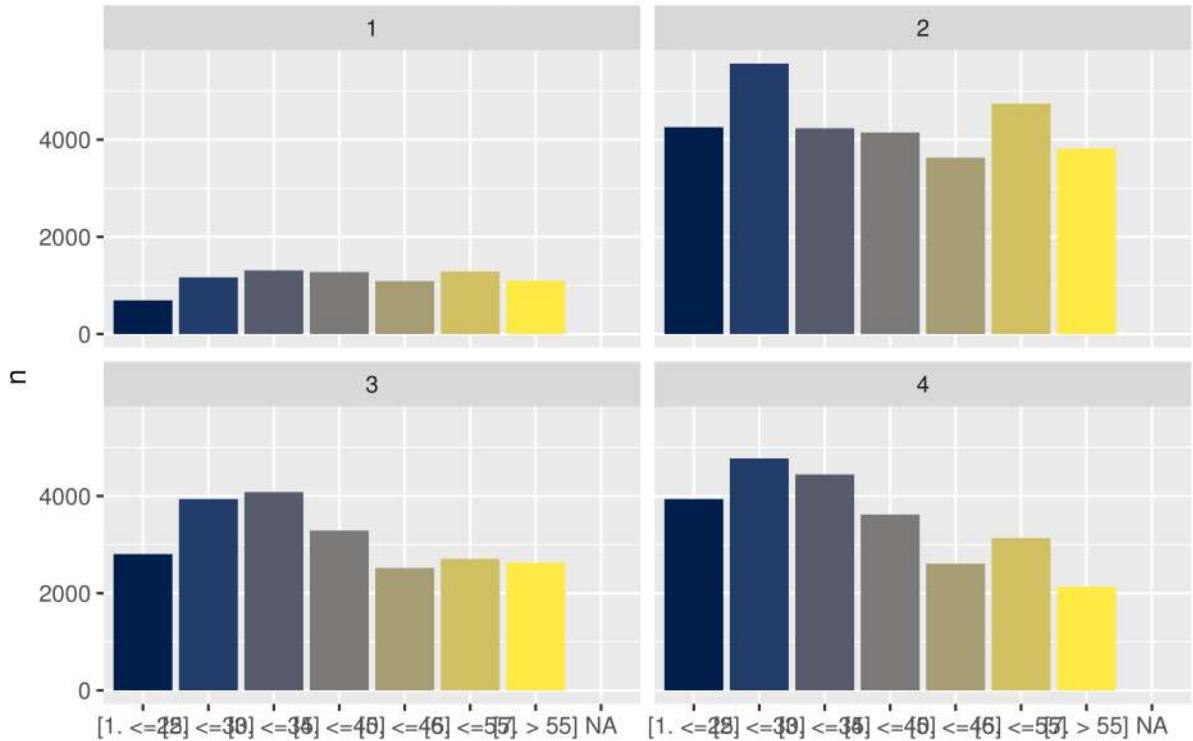
## Warning: package 'viridis' was built under R version 3.6.3
## Loading required package: viridisLite
#install.packages("hrbrthemes")
library(hrbrthemes)

## Warning: package 'hrbrthemes' was built under R version 3.6.3
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.
##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and
##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
edad_cluster = Data_Segmentada_Unida[,c(2,5)] %>%
  dplyr:::group_by(edad, clusteres_consumo) %>%
  dplyr:::summarise(n = n())

## Warning: Factor `edad` contains implicit NA, consider using
## `forcats::fct_explicit_na`
ggplot(edad_cluster, aes(fill=edad, y=n, x=edad)) +
  geom_bar(position="dodge", stat="identity") +
  scale_fill_viridis(discrete = T, option = "E") +
  ggtitle("Distribución de las edades por cluster") +
```

```
facet_wrap(~clusteres_consumo) +
#theme_ipsum() +
theme(legend.position="none") +
xlab("")
```

Distribución de las edades por cluster

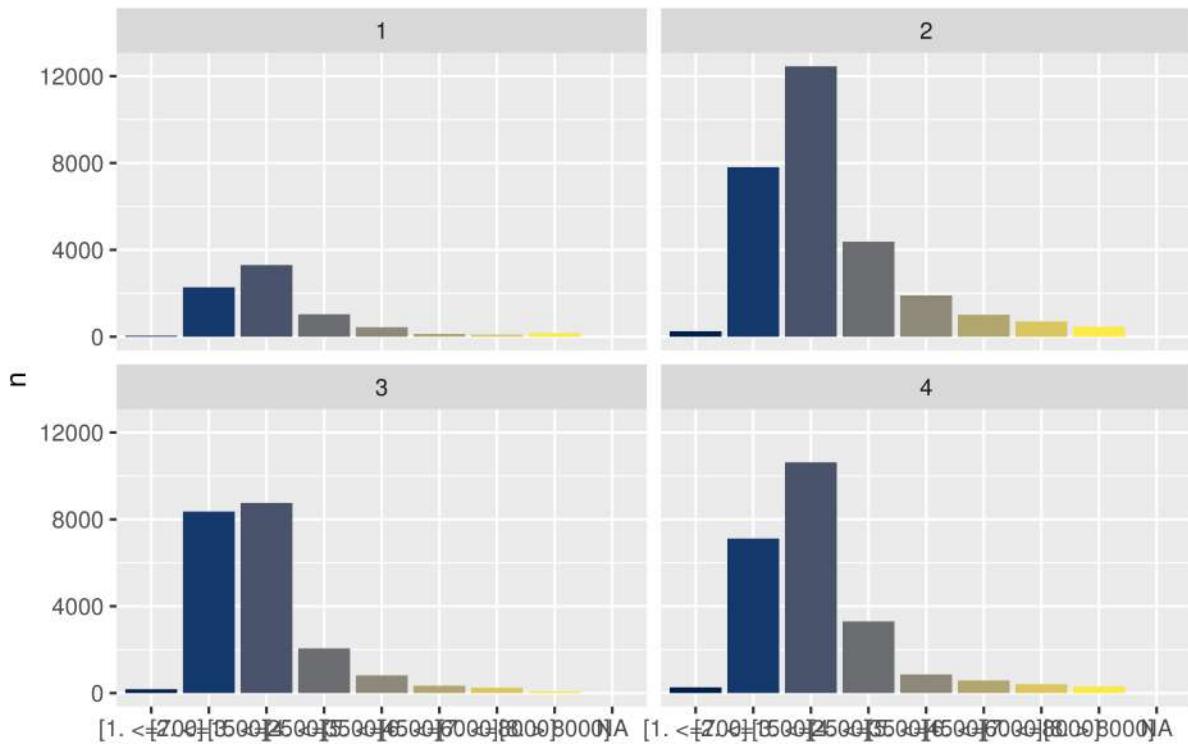


```
ingreso_cluster = Data_Segmentada_Unida[,c(3,5)] %>%
  group_by(ingreso, clusteres_consumo) %>%
  summarise(n = n())

## Warning: Factor `ingreso` contains implicit NA, consider using
## `forcats::fct_explicit_na`

ggplot(ingreso_cluster, aes(fill=ingreso, y=n, x=ingreso)) +
  geom_bar(position="dodge", stat="identity") +
  scale_fill_viridis(discrete = T, option = "E") +
  ggttitle("Distribución de los ingresos por segmento") +
  facet_wrap(~clusteres_consumo) +
#theme_ipsum() +
  theme(legend.position="none") +
  xlab("")
```

Distribución de los ingresos por segmento



```

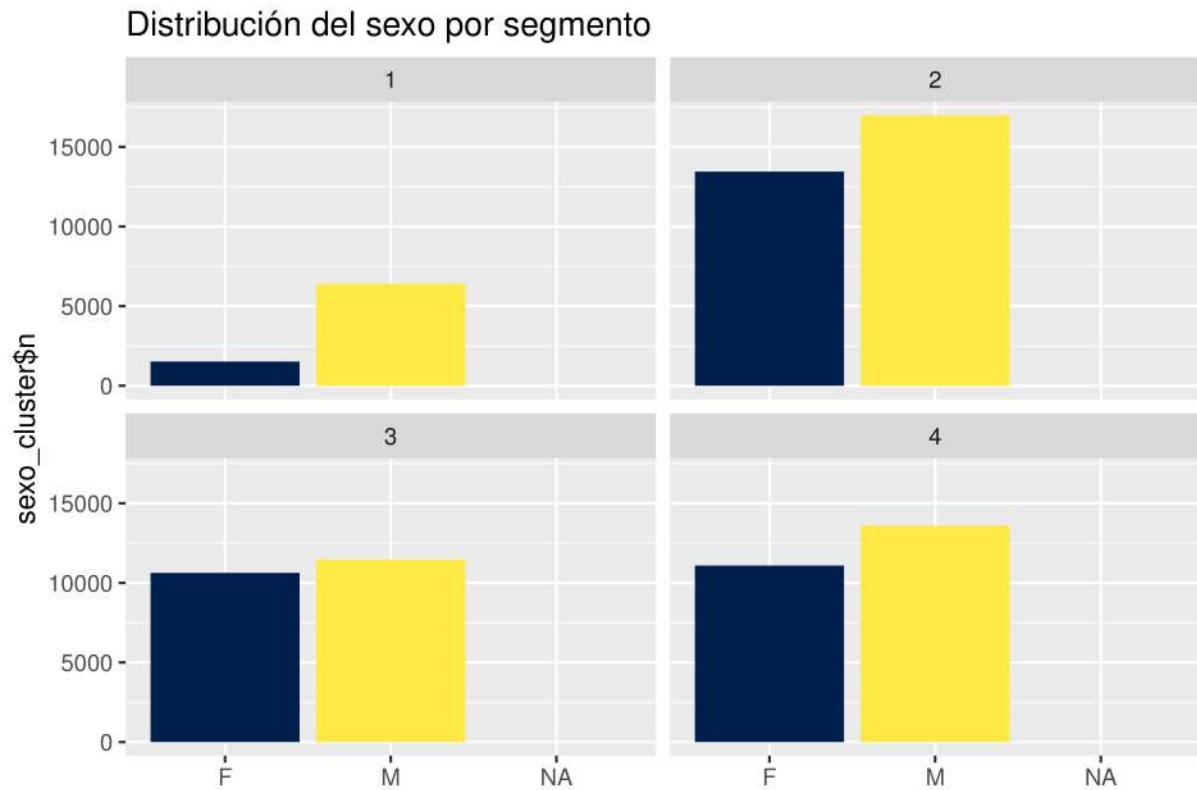
sexo_cluster = Data_Segmentada_Unida[,c(4,5)] %>%
  group_by(sexo, clusteres_consumo) %>%
  summarise(n = n())

## Warning: Factor `sexo` contains implicit NA, consider using
## `forcats::fct_explicit_na`

ggplot(sexo_cluster, aes(fill=sexo_cluster$sexo, y=sexo_cluster$n, x=sexo_cluster$sexo)) +
  geom_bar(position="dodge", stat="identity") +
  scale_fill_viridis(discrete = T, option = "E") +
  ggttitle("Distribución del sexo por segmento") +
  facet_wrap(~clusteres_consumo) +
  #theme_ipsum() +
  theme(legend.position="none") +
  xlab("")

## Warning: Factor `sexo` contains implicit NA, consider using
## `forcats::fct_explicit_na`

```



Clustering de Partición entorno a Centroides (PAM)

Una definición que podemos encontrar en wikipedia de este método, es la siguiente: “Tanto el k-medoids como el k-means son algoritmos que trabajan con particiones (dividiendo el conjunto de datos en grupos) y ambos intentan minimizar la distancia entre puntos que se añadirían a un grupo y otro punto designado como el centro de ese grupo. En contraste con el algoritmo k-means, k-medoids escoge datapoints como centros y trabaja con una métrica arbitraria de distancias entre datapoints en vez de usar la norma **l2**. En 1987 se propuso este método para el trabajo con la norma **l1** y otras distancias.”

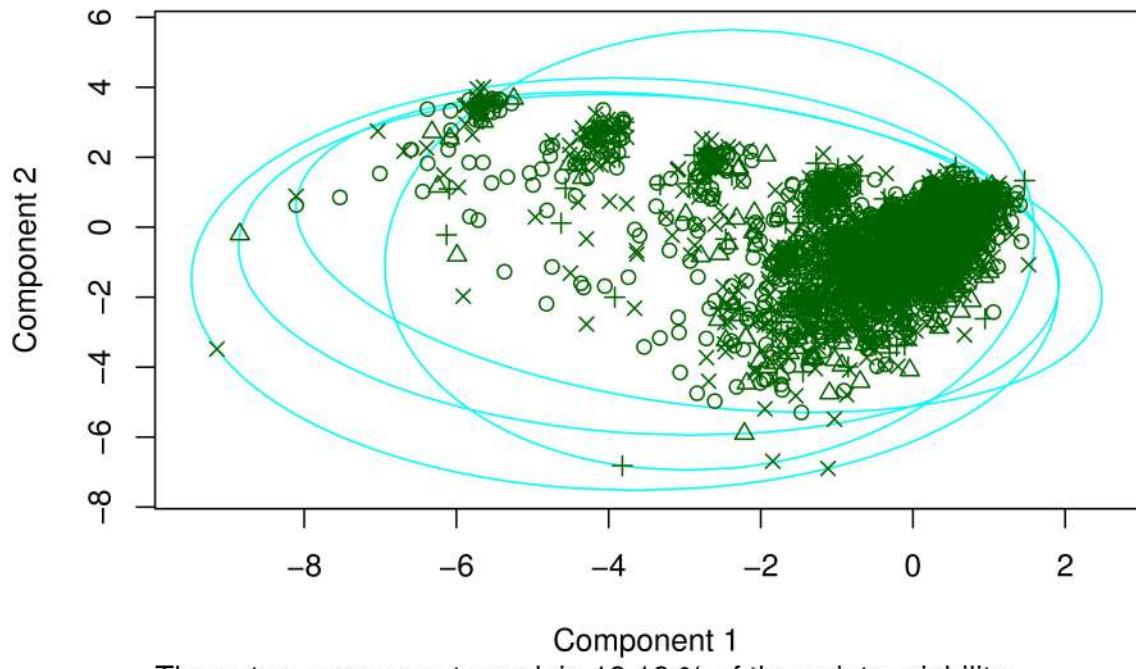
```
# Usamos la librería purrr
library(purrr)

# Traemos la data trabajada en el profiling
datos = data_escale

# Generamos los segmentos usando el método basado en medoides PAM. Usamos el número de segmentos 4, obt
set.seed(123)
pam_clusters <- pam(x = datos, k = 4, metric = "manhattan")

# Graficamos
clusplot(pam_clusters)
```

```
clusplot(pam(x = datos, k = 4, metric = "manhattan"))
```



Component 1

These two components explain 18.18 % of the point variability.

```
#####
# Validación
#####

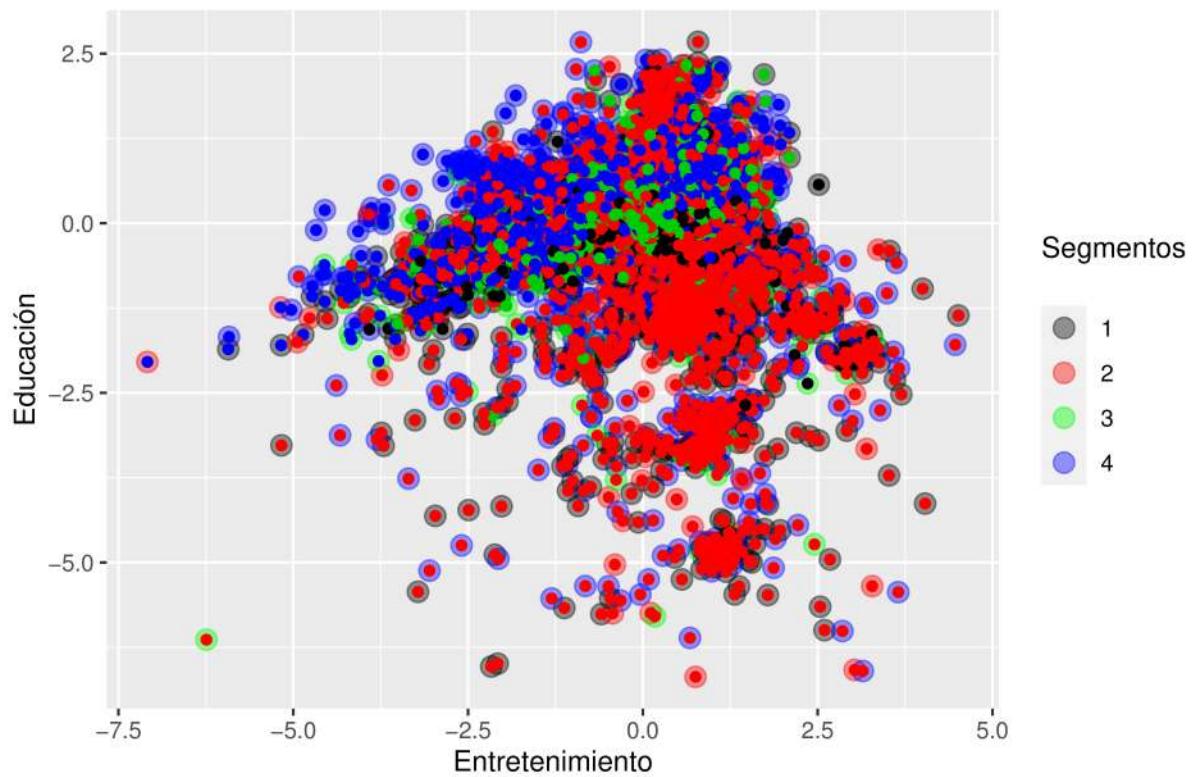
# Usamos la media de los coeficientes de silueta para validar si el agrupamiento es bueno
sil_pam <- silhouette(pam_clusters$clustering, dist(data_escala))
aggregate(sil_pam[,3], list(sil_pam[,1]), mean)

##   Group.1      x
## 1      1 -0.009606754
## 2      2  0.047369759
## 3      3  0.242406914
## 4      4  0.056602857

# Traemos la data y la unimos a los clústeres obtenidos
df=cbind(data_escala,pam_clusters$clustering)
df=data.frame(df)
colnames(df)[12] = "clusteres_consumo"

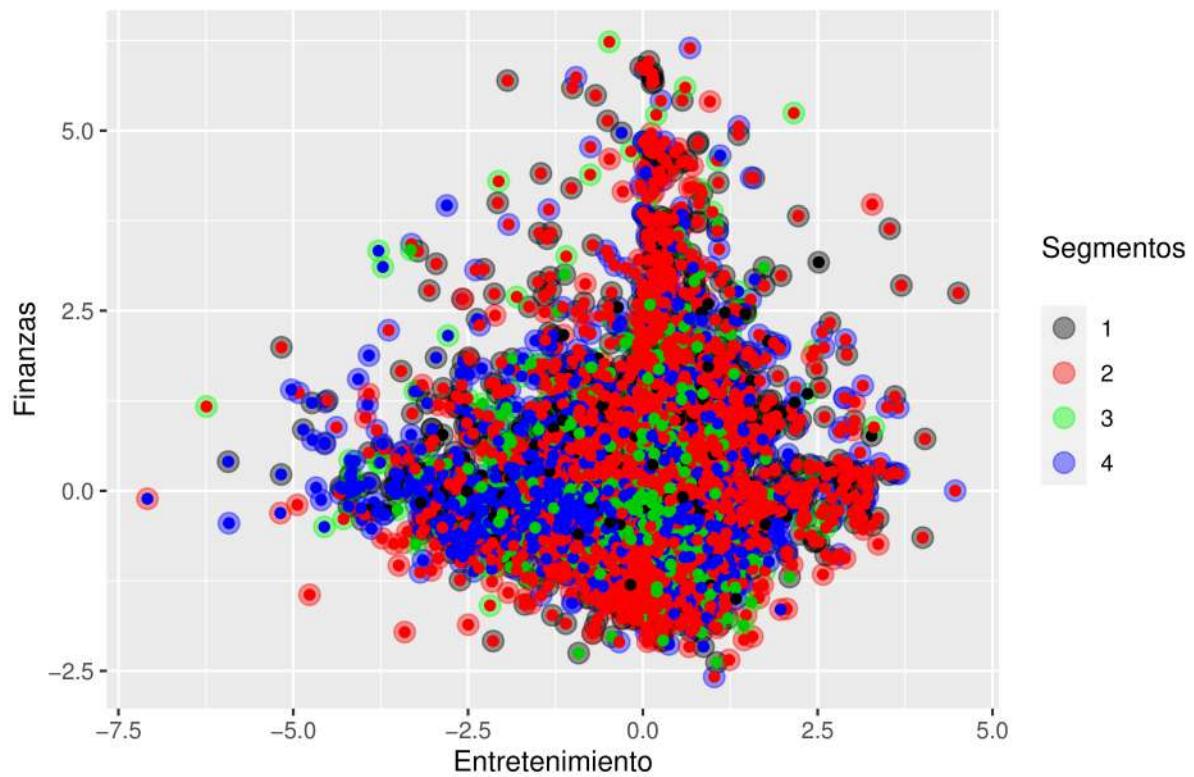
ggplot(df, aes(df$sector_entretenimiento, df$sector_eduacion, color = as.factor(df$clusteres_consumo)))
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +
  scale_color_manual(values = c('black', 'red', 'green', "blue"))+
  ggtitle("Entretenimiento vs Educación") +
  xlab("Entretenimiento") + ylab("Educación")+
  labs(color = "Segmentos\n")
```

Entretenimiento vs Educación



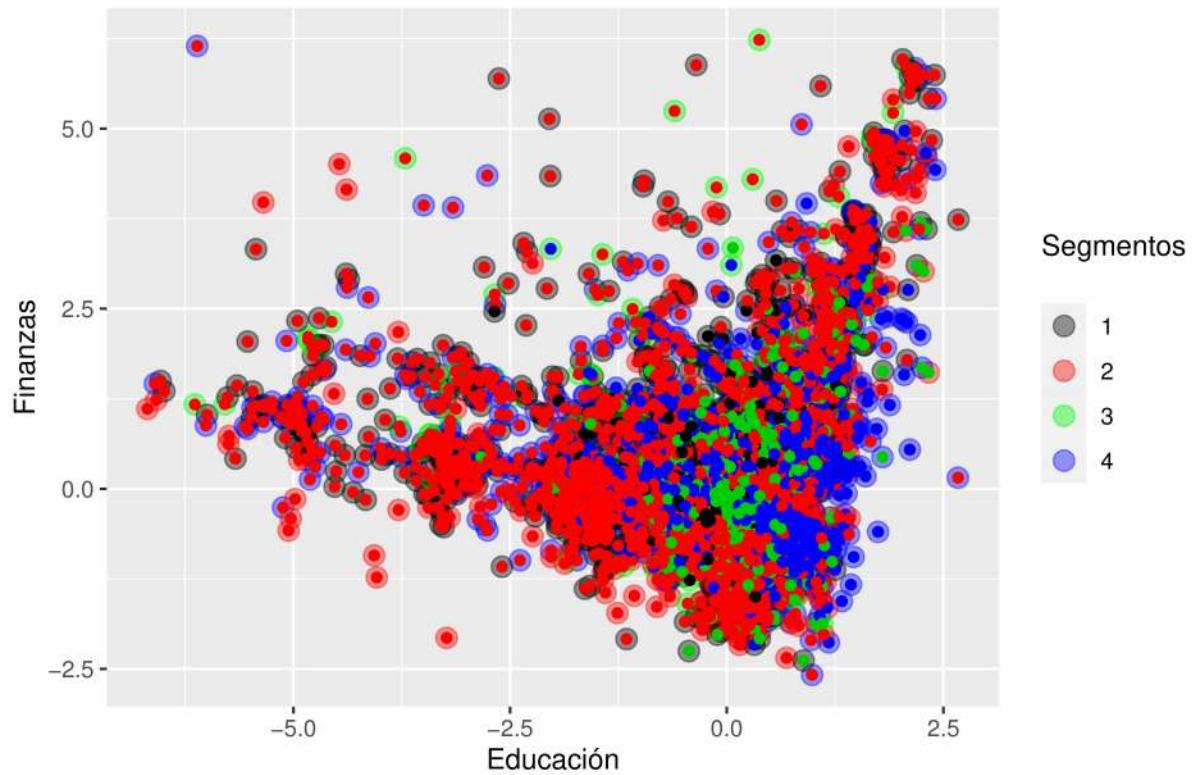
```
ggplot(df, aes(df$sector_entretenimiento, df$sector_finanzas, color = as.factor(df$clusteres_consumo))) +  
  geom_point(alpha = 0.4, size = 3.5) +  
  geom_point(col = as.factor(clusteres_consumo)) +  
  scale_color_manual(values = c('black', 'red', 'green', "blue")) +  
  ggtitle("Entretenimiento vs Finanzas") +  
  xlab("Entretenimiento") + ylab("Finanzas") +  
  labs(color = "Segmentos\\n")
```

Entretenimiento vs Finanzas



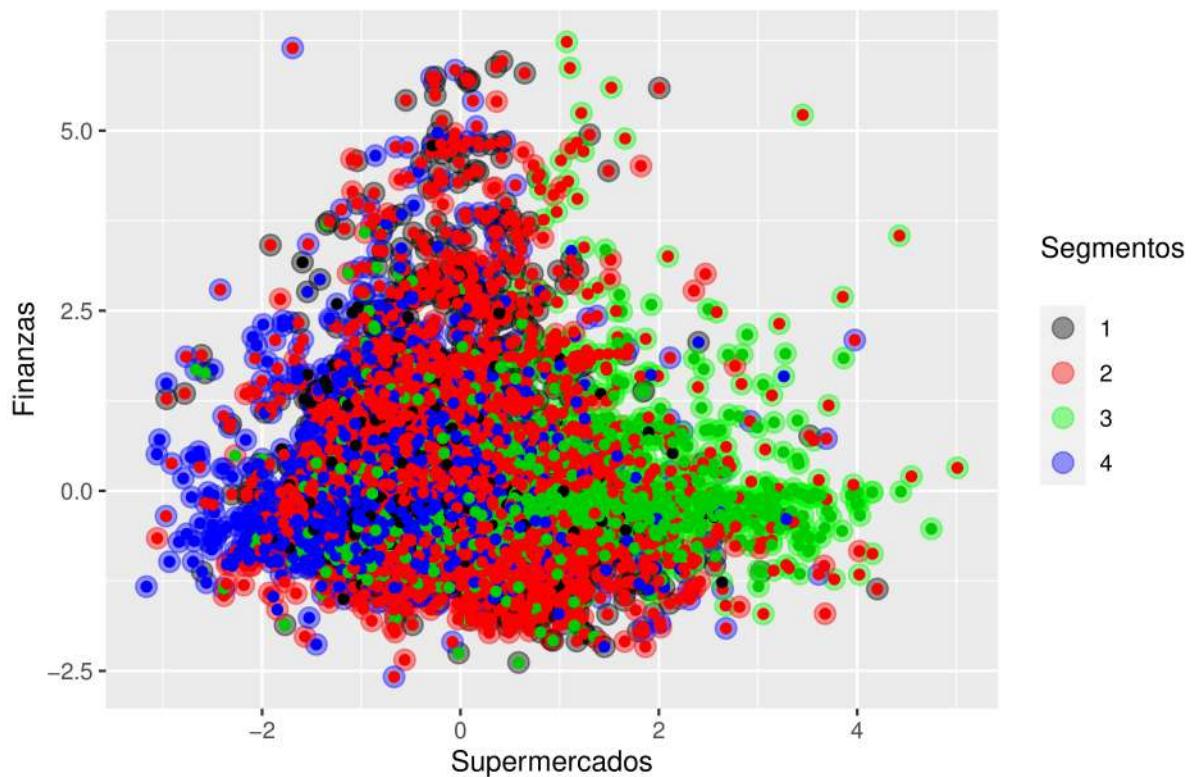
```
ggplot(df, aes(df$sector_eduacion, df$sector_finanzas, color = as.factor(df$clusteres_consumo))) +  
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +  
  scale_color_manual(values = c('black', 'red', 'green', "blue")) +  
  ggtitle("Educación vs Finanzas") +  
  xlab("Educación") + ylab("Finanzas") +  
  labs(color = "Segmentos\\n")
```

Educación vs Finanzas



```
ggplot(df, aes(df$sector_supermcd, df$sector_finanzas, color = as.factor(df$clusteres_consumo))) +  
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = as.factor(clusteres_consumo)) +  
  scale_color_manual(values = c('black', 'red', 'green', "blue")) +  
  ggtitle("Supermercados vs Finanzas") +  
  xlab("Supermercados") + ylab("Finanzas") +  
  labs(color = "Segmentos\\n")
```

Supermercados vs Finanzas



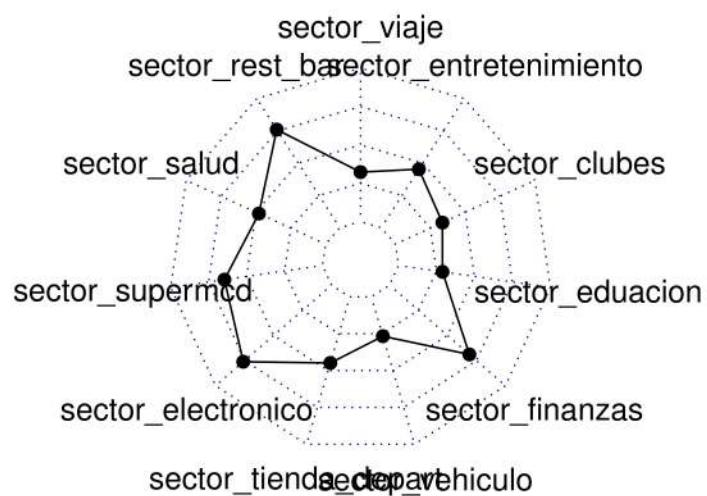
Graficamos la influencia de los sectores por cada segmento calculado.

```
# Traemos la data original y le agregamos el clustering
Data_Segmentada = cbind(Data_Consumo_Final, data.frame(pam_clusters$clustering))
colnames(Data_Segmentada)[16] = "cluster"

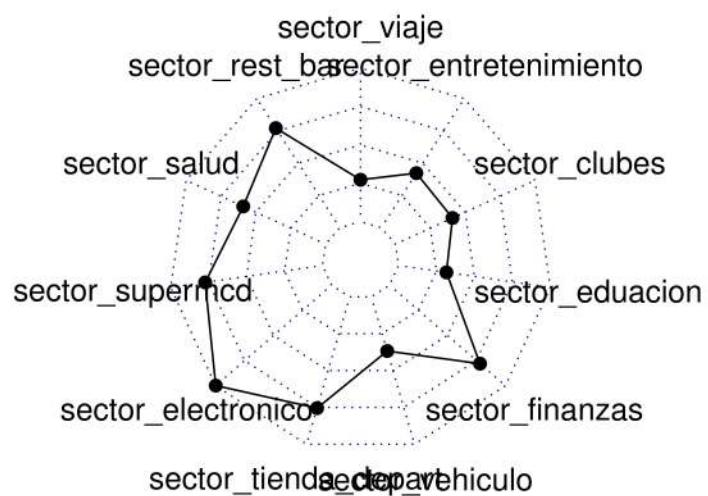
# le damos forma para trabajarla
Data_Segmentada_Unida=tidyr::gather(data=Data_Segmentada, key = desc_segmento, value = valor_rubro, 5:11)

#install.packages("fmsb")
library(fmsb)

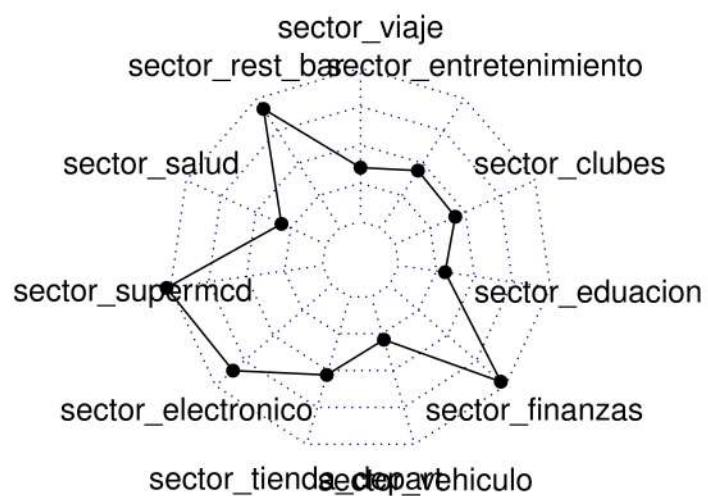
# Usamos el gráfico de radar para ver las preferencias de cada segmento.
radar_data = aggregate(Data_Segmentada[, 5:15], list(Data_Segmentada$cluster), max)
radar_data= data.frame(radar_data)
radar_data_1 = data.frame(radar_data[1,2:12])
data_1 = rbind(rep(4,11) , rep(0,11) , radar_data_1)
radarchart(data_1)
```



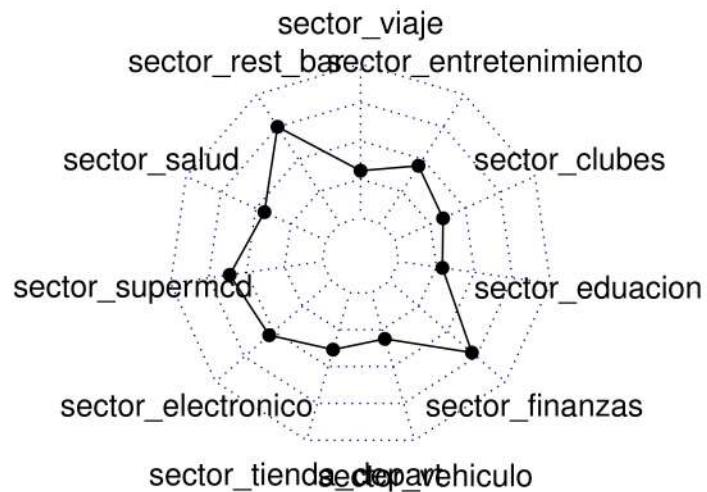
```
radar_data_2 = data.frame(radar_data[2,2:12])
data_2 = rbind(rep(3.2,11) , rep(0,11) , radar_data_2)
radarchart(data_2)
```



```
radar_data_3 = data.frame(radar_data[3,2:12])
data_3 = rbind(rep(3,11) , rep(0,11) , radar_data_3)
radarchart(data_3)
```



```
radar_data_4 = data.frame(radar_data[4,2:12])
data_4 = rbind(rep(4,11) , rep(0,11) , radar_data_4)
radarchart(data_4)
```

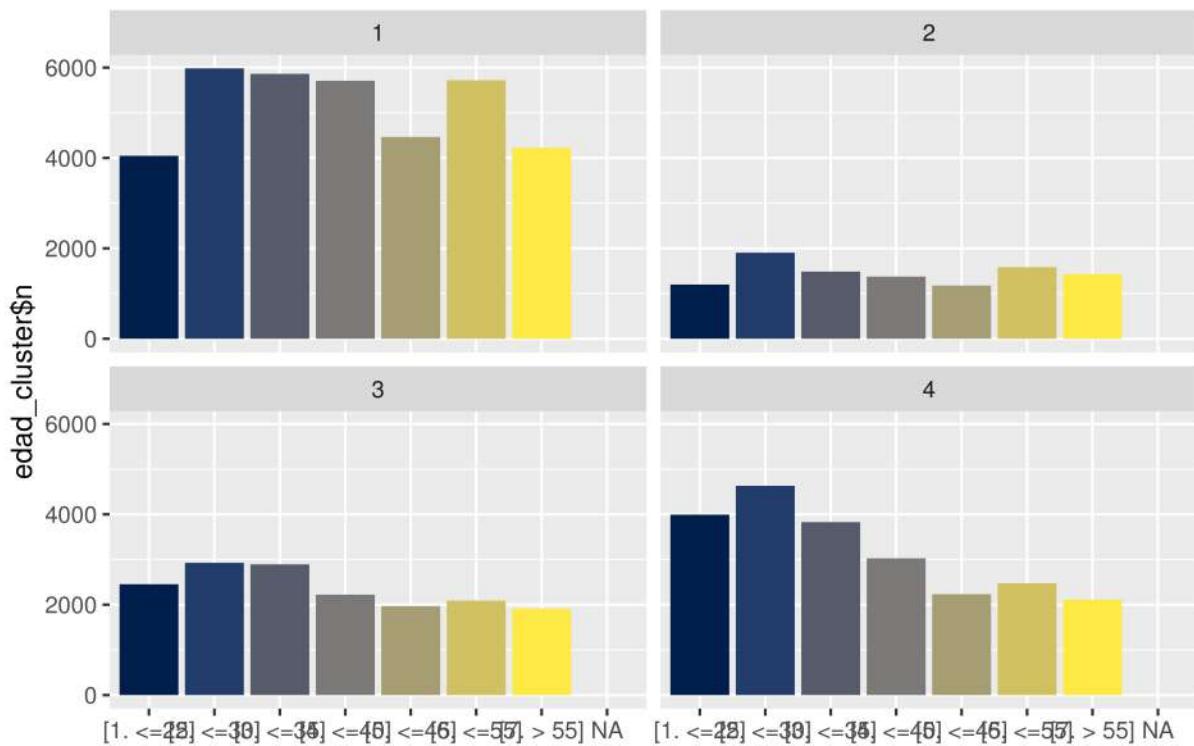


Ahora, mostramos los descriptivos:

```
edad_cluster = Data_Segmentada_Unida[,c(2,5)] %>%
  group_by(edad, cluster) %>%
  summarise(n = n())

library(viridis)
library(hrbrthemes)
ggplot(edad_cluster, aes(fill=edad_cluster$edad, y=edad_cluster$n, x=edad_cluster$edad)) +
  geom_bar(position="dodge", stat="identity") +
  scale_fill_viridis(discrete = T, option = "E") +
  ggtitle("Distribución de las edades por cluster") +
  facet_wrap(~cluster) +
  #theme_ipsum() +
  theme(legend.position="none") +
  xlab("")
```

Distribución de las edades por cluster



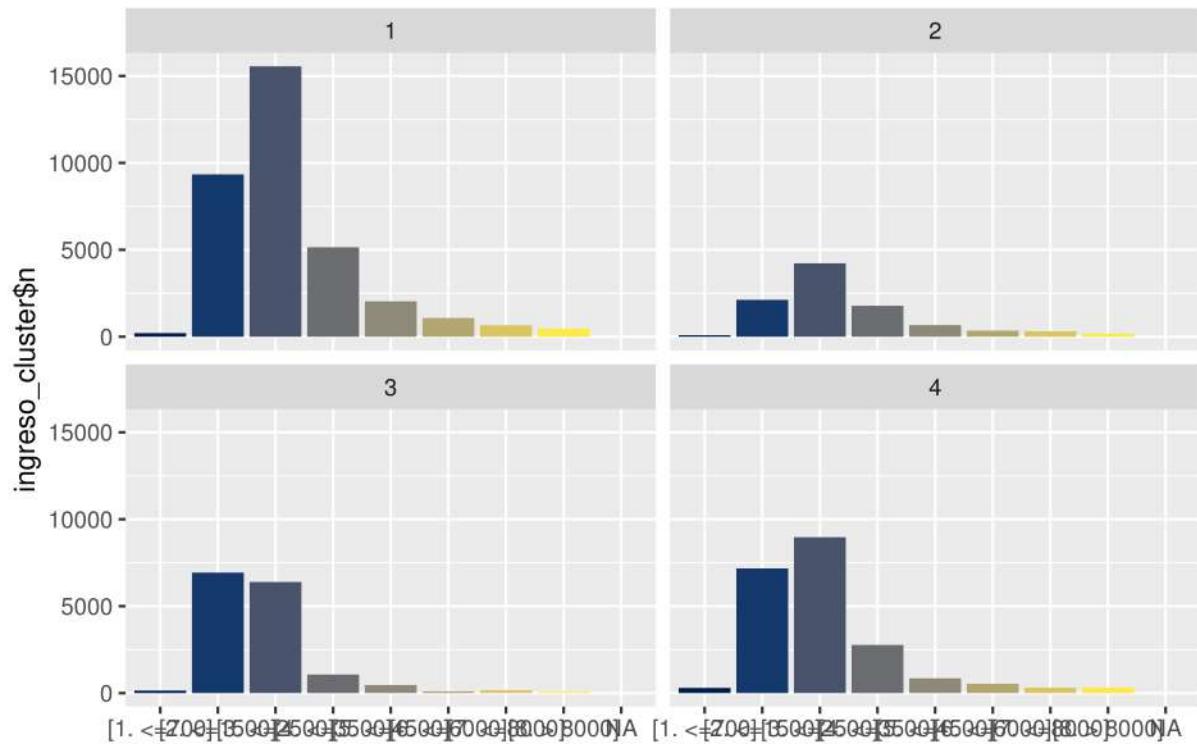
```

ingreso_cluster = Data_Segmentada_Unida[,c(3,5)] %>%
  group_by(ingreso, cluster) %>%
  summarise(n = n())

ggplot(ingreso_cluster, aes(fill=ingreso_cluster$ingreso, y=ingreso_cluster$n, x=ingreso_cluster$ingreso,
  geom_bar(position="dodge", stat="identity") +
  scale_fill_viridis(discrete = T, option = "E") +
  gtitle("Distribución de los ingresos por segmento") +
  facet_wrap(~cluster) +
  #theme_ipsum() +
  theme(legend.position="none") +
  xlab(""))


```

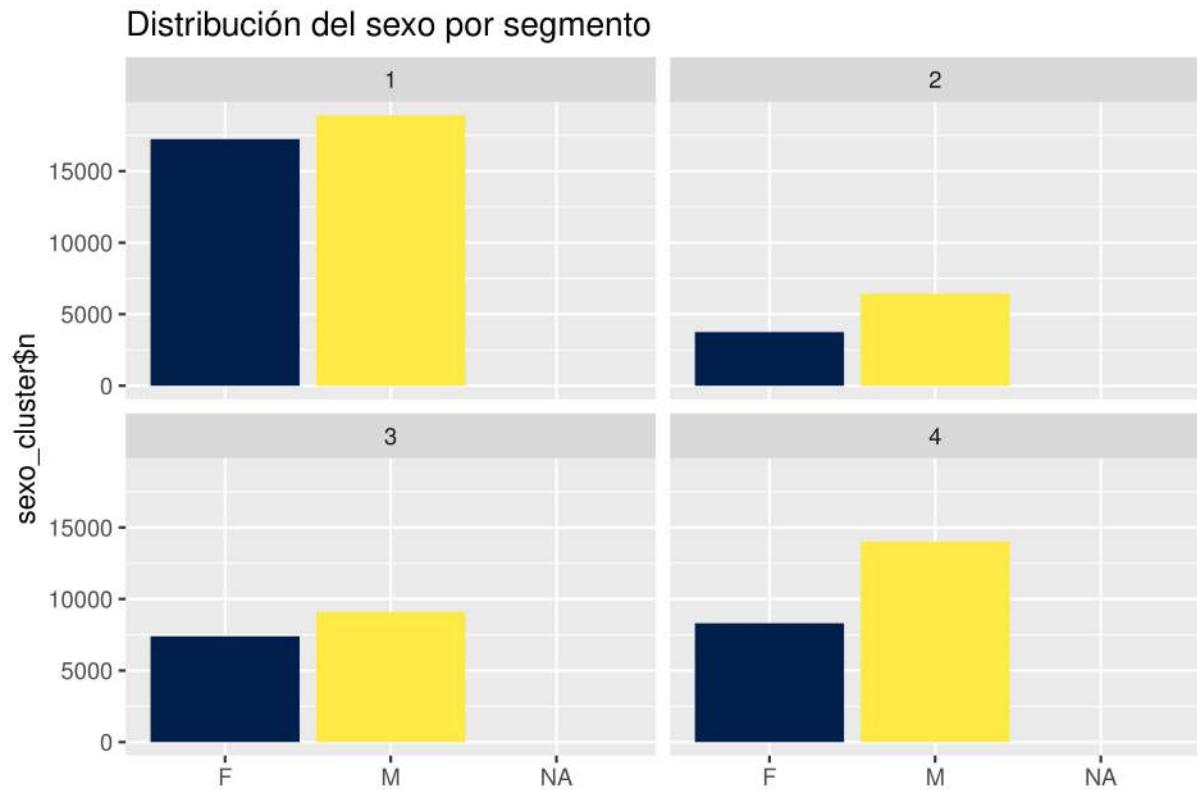
Distribución de los ingresos por segmento



```

sexo_cluster = Data_Segmentada_Unida[,c(4,5)] %>%
  group_by(sexo, cluster) %>%
  summarise(n = n())

ggplot(sexo_cluster, aes(fill=sexo_cluster$sexo, y=sexo_cluster$n, x=sexo_cluster$sexo)) +
  geom_bar(position="dodge", stat="identity") +
  scale_fill_viridis(discrete = T, option = "E") +
  ggtile("Distribución del sexo por segmento") +
  facet_wrap(~cluster) +
  #theme_ipsum() +
  theme(legend.position="none") +
  xlab("")
  
```



COMPARACIONES

Para todos los casos (PAM, K-MEANS y jerárquico) hemos usado 4 segmentos, los cuales fueron seleccionados después de inspeccionar varias técnicas. La validación de cuán bien agrupan los clústeres la realizamos mediante el coeficiente de silueta. En el siguiente link, se encuentra la explicación de este coeficiente como validación clustering: <https://towardsdatascience.com/clustering-analysis-in-r-using-k-means-73eca4fb7967>

Vemos que el mejor es el kmeans, pues todos sus segmentos tienen una media positiva. Esto también lo podemos ver en el gráfico.

```
# Coeficiente clustering

# Kmeans
aggregate(sil_4[,3], list(sil_4[,1]), mean)

##   Group.1      x
## 1      1 0.02747813
## 2      2 0.04869697
## 3      3 0.17776486
## 4      4 0.17755005

# Jerárquico
aggregate(sil_jerar[,3], list(sil_jerar[,1]), mean)

##   Group.1      x
## 1      1 0.24657669
## 2      2 -0.15701444
## 3      3 0.14204118
## 4      4 0.08694901
```

```
# PAM
aggregate(sil_pam[,3], list(sil_pam[,1]), mean)

##   Group.1      x
## 1      1 -0.009606754
## 2      2  0.047369759
## 3      3  0.242406914
## 4      4  0.056602857
```

Resultados

Análisis visual del conjunto de datos

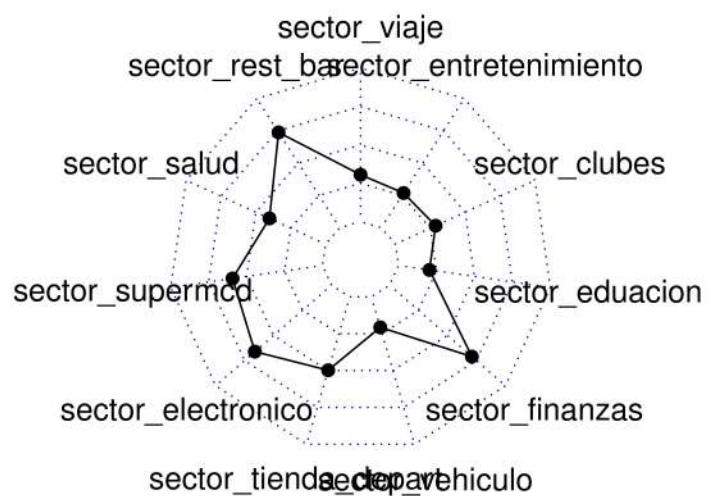
En la primera parte echamos mano de algunos gráficos para revisar algunos indicadores, como cantidad de transacciones, respecto a algunas variables. Finalmente, veremos algunos descriptivos de cada segmento, como la edad, el ingreso y el sexo. Asimismo, presentamos los gráficos de radar que son muy útiles para ver las preferencias por segmentos.

```
# Traemos a la base, los segmentos hallados
Data_Segmentada = cbind(Data_Consumo_Final, cluster=consumo_clusters$cluster )

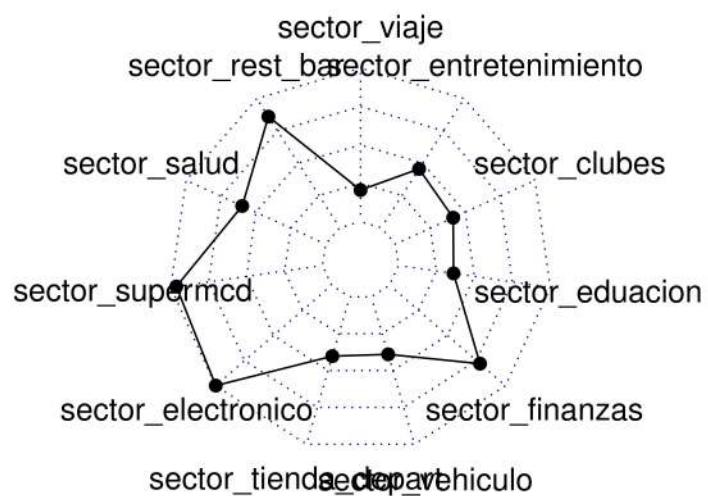
# le damos forma para traerla
Data_Segmentada_Unida=tidyr::gather(data=Data_Segmentada, key = desc_segmento, value = valor_rubro, 5:15)

#install.packages("fmsb")
library(fmsb)

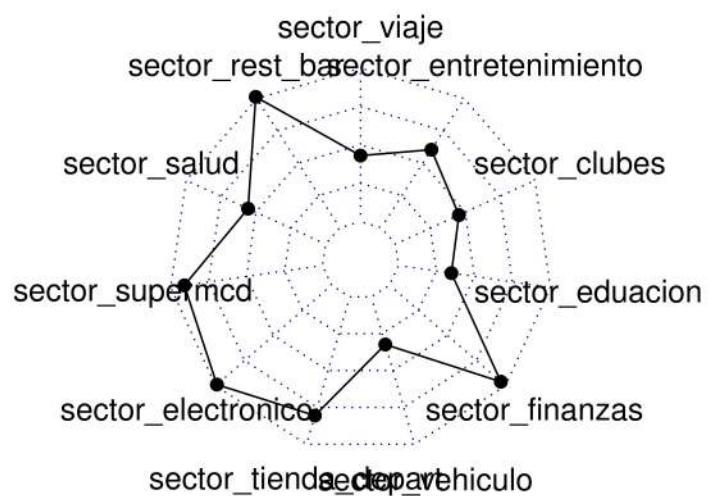
# Usamos el gráfico de radar para ver las preferencias de cada segmento.
radar_data = aggregate(Data_Segmentada[, 5:15], list(Data_Segmentada$cluster), max)
radar_data= data.frame(radar_data)
radar_data_1 = data.frame(radar_data[1,2:12])
data_1 = rbind(rep(4,11) , rep(0,11) , radar_data_1)
radarchart(data_1)
```



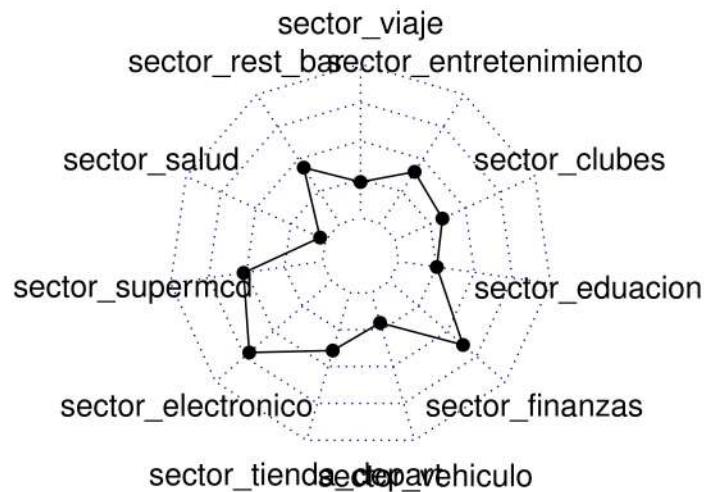
```
radar_data_2 = data.frame(radar_data[2,2:12])
data_2 = rbind(rep(3.2,11) , rep(0,11) , radar_data_2)
radarchart(data_2)
```



```
radar_data_3 = data.frame(radar_data[3,2:12])
data_3 = rbind(rep(3,11) , rep(0,11) , radar_data_3)
radarchart(data_3)
```



```
radar_data_4 = data.frame(radar_data[4,2:12])
data_4 = rbind(rep(4,11) , rep(0,11) , radar_data_4)
radarchart(data_4)
```



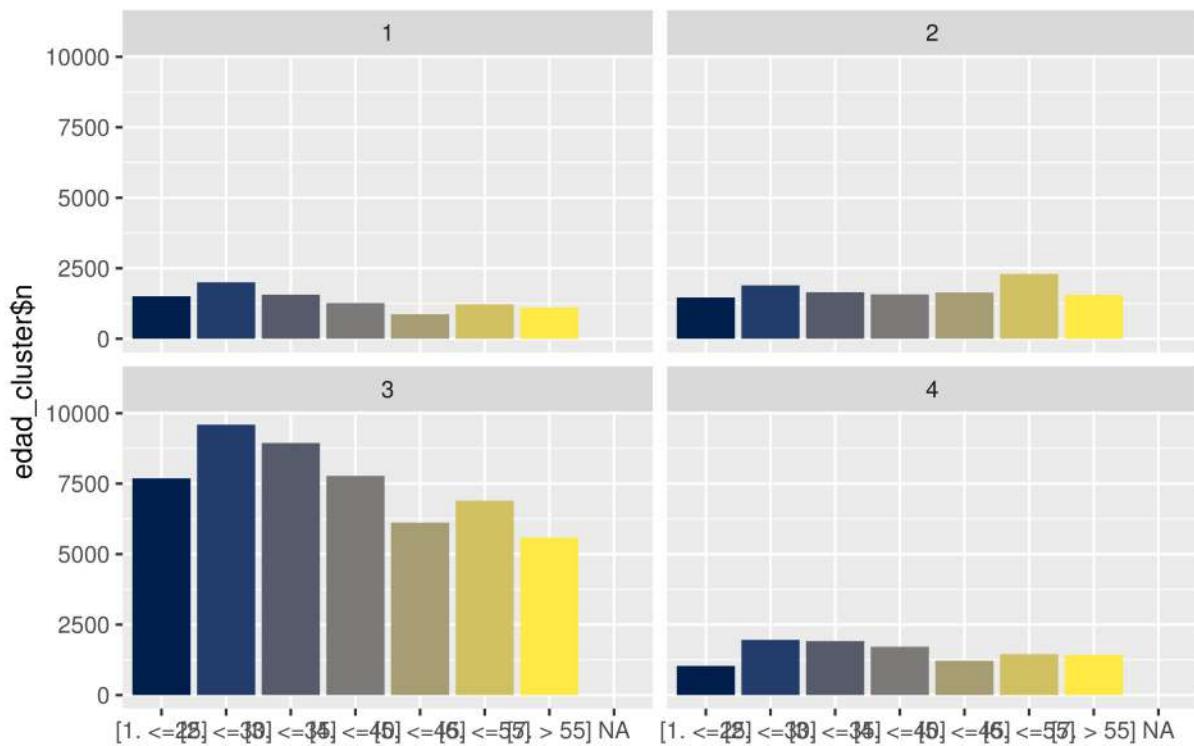
Seguimos con los demás descriptivos, en los que vemos las distribuciones de edad, sexo e ingreso por segmentos.

```
library(viridis)
#install.packages("hrbrthemes")
library(hrbrthemes)

edad_cluster = Data_Segmentada_Unida[,c(2,5)] %>%
  group_by(edad, cluster) %>%
  summarise(n = n())

ggplot(edad_cluster, aes(fill=edad_cluster$edad, y=edad_cluster$n, x=edad_cluster$edad)) +
  geom_bar(position="dodge", stat="identity") +
  scale_fill_viridis(discrete = T, option = "E") +
  ggtitle("Distribución de las edades por cluster") +
  facet_wrap(~cluster) +
  #theme_ipsum() +
  theme(legend.position="none") +
  xlab("")
```

Distribución de las edades por cluster

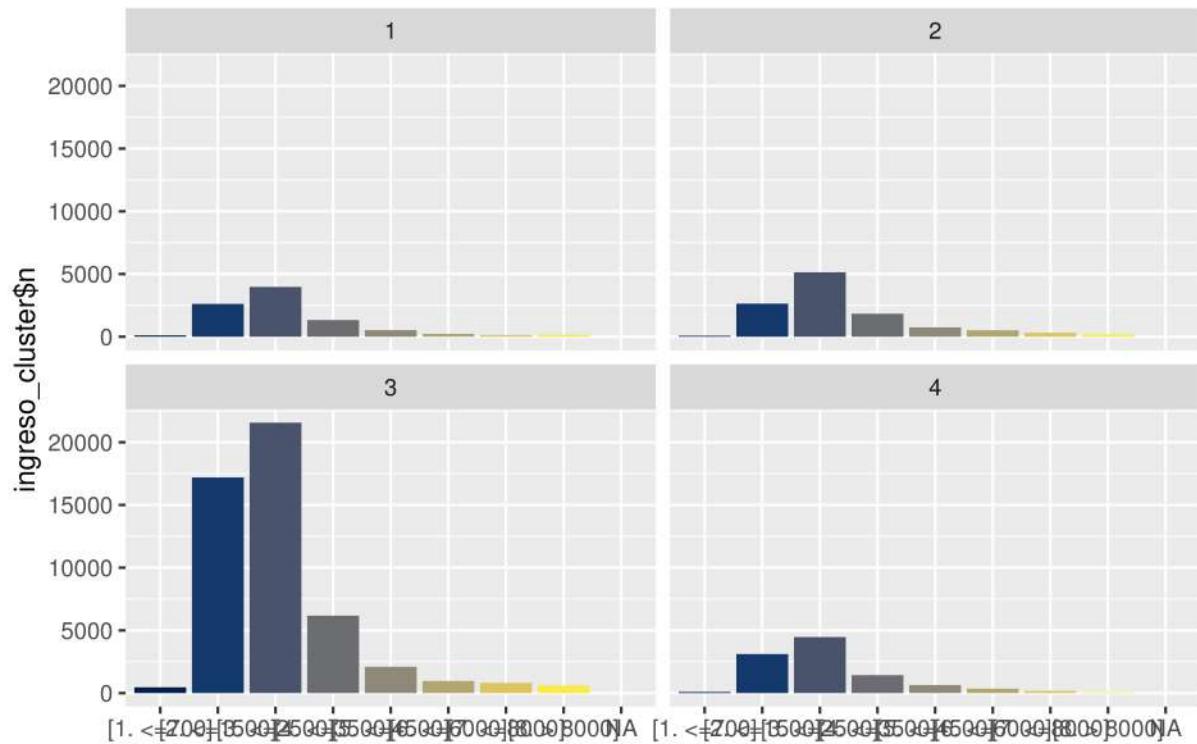


```
ingreso_cluster = Data_Segmentada_Unida[,c(3,5)] %>%
  group_by(ingreso, cluster) %>%
  summarise(n = n())

ggplot(ingreso_cluster, aes(fill=ingreso_cluster$ingreso, y=ingreso_cluster$n, x=ingreso_cluster$ingreso,
  geom_bar(position="dodge", stat="identity") +
  scale_fill_viridis(discrete = T, option = "E") +
  ggtile("Distribución de los ingresos por segmento") +
  facet_wrap(~cluster) +
  #theme_ipsum() +
  theme(legend.position="none") +
  xlab(""))


```

Distribución de los ingresos por segmento

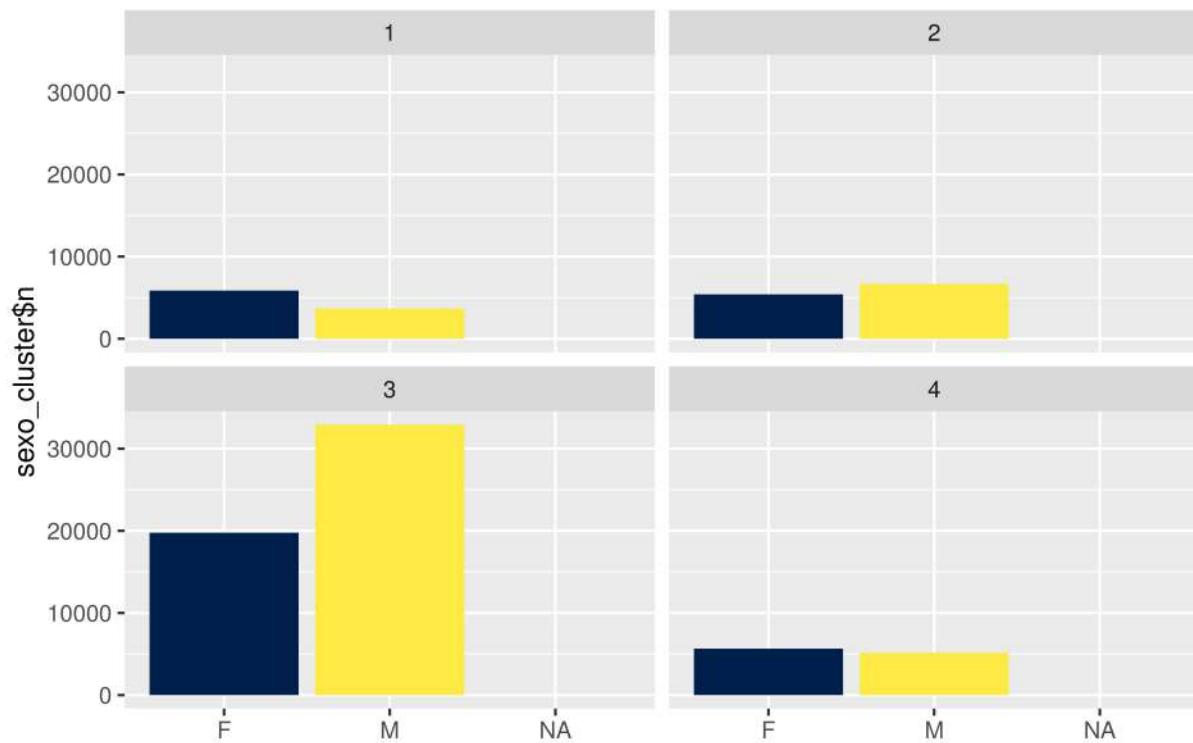


```

sexo_cluster = Data_Segmentada_Unida[,c(4,5)] %>%
  group_by(sexo, cluster) %>%
  summarise(n = n())

ggplot(sexo_cluster, aes(fill=sexo_cluster$sexo, y=sexo_cluster$n, x=sexo_cluster$sexo)) +
  geom_bar(position="dodge", stat="identity") +
  scale_fill_viridis(discrete = T, option = "E") +
  ggtile("Distribución del sexo por segmento") +
  facet_wrap(~cluster) +
  #theme_ipsum() +
  theme(legend.position="none") +
  xlab("")
  
```

Distribución del sexo por segmento



Resolución del problema

Una vez obtenido los segmentos, 4, es importante ver cuál es la característica asociada. Hemos visto que hay segmentos en los que los hombres son los de mayor proporción, o donde el sector entretenimiento y salud son los más consumidos. Esto podría darnos un perfilamiento acerca de las preferencias de consumo y edades. Si dispusiéramos de mayor información (si es un banco, de hecho que lo maneja), podríamos enriquecer ese perfilamiento con más variables y estos segmentos fácilmente podrían corresponder a estilos de vida, sobre un enfoque de consumo. La utilidad mayor sería mejorar las ofertas, por ejemplo ofreciendo descuentos a los segmentos donde prevale el consumo entretenimiento.