UNIVERSIDAD DE GRANADA

Reglas de asociación

Carlos Morales Aguilera

13/12/2020

Lectura de datos

El primer paso en el problema es leer correctamente los datos, para ello se utiliza la función `read_excel` de la librería **readxl**, a continuación se transforma el conjunto de datos obtenido en una estructura del tipo **dataframe** propia de *R*.

```
f <- "C:\\Users\\power\\Desktop\\TID\\Practica4\\Prestamo.xls"
# Read dataframe
my_description <- read_excel(f, sheet = "descripcion")

bd_prestamo <- as.data.frame(read_excel(f, sheet = "datos"))
```

Una de las buenas prácticas aprendidas de las prácticas anteriores es la de eliminar valores perdidos, por lo que previamente a tratar con los datos, nos aseguramos de eliminar los posibles valores perdidos si hubieran.

Por otro lado, otro de los errores comunes vistos anteriormente es que ciertos modelos no pueden trabajar con nombres de variables complejos, por lo que para ello se han analizado las columnas, y se ha decidido borrar los espacios de aquellas que poseyeran dicho carácter.

```
# Omit NAs
bd_prestamo <- na.omit(bd_prestamo)

# Get column names and erase ' ' character
cols <- colnames(bd_prestamo)
for(i in 1:length(cols)) cols[i] <- gsub(" ", "", cols[i])
# Set column names
names(bd_prestamo) <- cols
```

Preprocesamiento de los datos

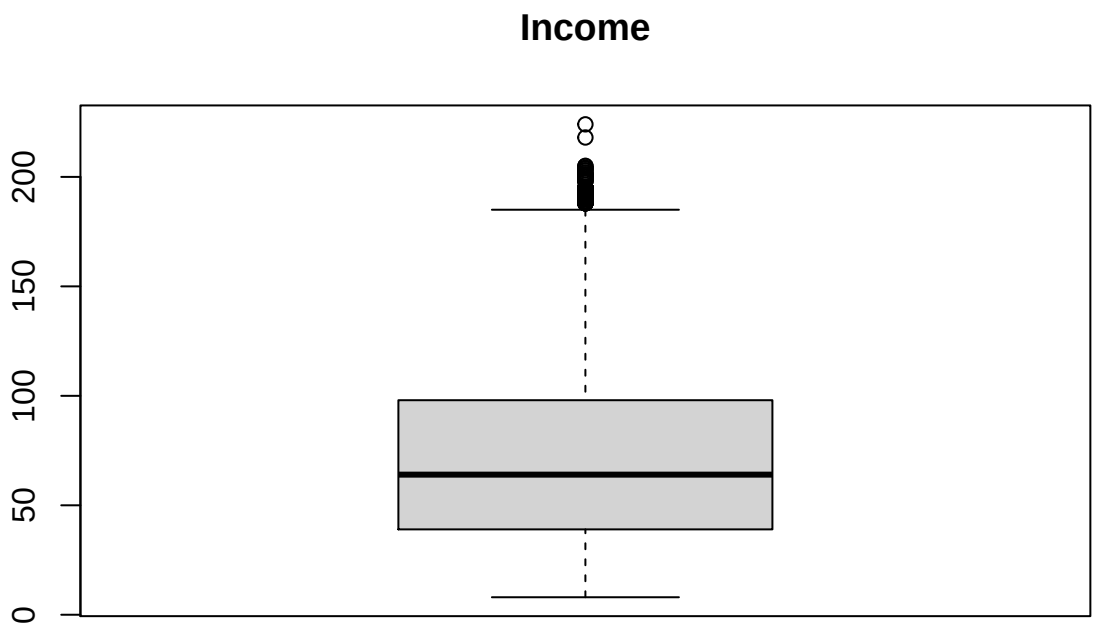
Para procesar los datos correctamente, lo primero es determinar la importancia de las distintas variables que se poseen en el conjunto de datos, tras ello, se ha llegado a la conclusión de que el código postal por sí solo no aporta suficiente información en este problema, por lo que queda descartado, ya que requeriría una discretización compleja basada en zonas geográficas.

Por otro lado, la información sobre la conversión de los clientes tras la última campaña, no llega a ser una información que aporte más información que la ya existente, ya que se desconoce en un principio el enfoque de la campaña, y el objetivo al final es observar que características poseen en común los clientes que han contratado varios servicios, es decir, sobre los que se ha realizado una *venta cruzada*, por lo que se descarta dicha información ya que el objetivo actual es distinto, pese a intentar alcanzar el mismo objetivo final.

```
# Remove unused columns
bd_prestamo[,"ZIPCode"] <- NULL
bd_prestamo[,"PersonalLoan"] <- NULL
```

Tras un primer análisis simple de los datos, se puede observar que los límites son coherentes, por lo que solo examinaremos posibles outliers en las variables **Income** y **Mortgage**.

```
# Boxplots Income and Mortgage
boxplot(bd_prestamo$Income, main = "Income")
```




```
boxplot(bd_prestamo$Mortgage, main = "Mortgage")
```

Tras analizar los *outliers* obtenidos, se puede observar que son casos donde se posee una nómina o una hipoteca muy elevada, por lo que aunque no son datos atípicos, se eliminarán del estudio ya que son clientes que se pueden considerar especiales, y el enfoque de la lógica del banco debería ser distinto para dichos casos especiales.

```
# Remove Income outliers
outliers <- boxplot.stats(bd_prestamo$Income)$out
for(i in 1:length(outliers))
  bd_prestamo <- bd_prestamo[!bd_prestamo$Income == outliers[i], ]
# Remove Mortgage outliers
outliers <- boxplot.stats(bd_prestamo$Mortgage)$out
for(i in 1:length(outliers))
  bd_prestamo <- bd_prestamo[!bd_prestamo$Mortgage == outliers[i], ]
```

Discretización de variables

Una de las técnicas principales de preprocesamiento de datos consisten en la discretización de valores continuos, o discretización de valores categóricos como agrupación de estas categorías en otras que abarquen más categoría en una misma. Para ello se ha utilizado la función cut, perteneciente al paquete *base* del propio *R*.

Para la realización de la discretización de las variables categóricas del problema, se han estudiado los distintos casos posibles en la información de *variables* del conjunto de datos, para ello se han tomado las siguientes decisiones:

- **Age**: Se divide en tres grupos según el rango.

  - *Joven* (0): 20-39 años.
  - *Medio* (1): 40-49 años.
  - *Mayor* (2): 50-70 años.

- **Experience**: Se divide en categorías según la experiencia profesional.
  - *Sin experiencia* (0): 0 años.
  - *Junior* (1): 1-9 años.
  - *Senior* (2): 10-24 años.
  - *Experto* (3): 25-45 años.

- **Income**: Se clasifica según rangos salariales, para distinguir los diferentes casos.
  - *Bajo* (0): 8-19 mil €.
  - *Medio-bajo* (1): 20-49 mil €.
  - *Medio-alto* (2): 50-99 mil €.
  - *Alto* (3): 100-185 mil €.

- **CCAvg**: Se divide de forma equitativa en tres grupos.
  - *Sin gastos* (0): 0.
  - *Gastos comunes* (1): 0.1-2.9 mil €.
  - *Gastos medios* (2): 3.0-5.9 mil €.
  - *Gastos altos* (3): 6.0-10.0 mil €.

- **Mortgage**: Se divide según rangos, para tener en cuenta las diferencias de hipotecas pendientes.
  - *Sin hipoteca* (0): 0 €.
  - *Baja* (1): 1-49 mil €.
  - *Media* (2): 50-99 mil €.
  - *Alta* (3): 100-250 mil €.

```
# Discretize Age by range (Young, Adult, Old)
bd_prestamo[,"Age"] <- cut(bd_prestamo[,"Age"],
                           breaks = c(20,40,50,70),
                           labels = c(0, 1, 2), right = FALSE)
# Discretize Experience by range (No experience, Junior, Senior, Expert)
bd_prestamo[,"Experience"] <- cut(bd_prestamo[,"Experience"],
                                  breaks = c(0, 1, 10, 25, 45),
                                  labels = c(0, 1, 2, 3), right = FALSE)
# Discretize Income by range (Low, Medium-low, Medium-high, High)
bd_prestamo[,"Income"] <- cut(bd_prestamo[,"Income"],
                              breaks = c(8, 20, 50, 100, 186),
                              labels = c(0, 1, 2, 3), right = FALSE)
# Discretize CCAvg by range (No spending, Low, Medium, High)
bd_prestamo[,"CCAvg"] <- cut(bd_prestamo[,"CCAvg"],
                             breaks = c(0, 0.1, 3.0, 6.0, 10.0),
                             labels = c(0, 1, 2, 3), right = FALSE)
# Discretize Mortgage by range (Low, Medium-low, Medium-high, High)
bd_prestamo[,"Mortgage"] <- cut(bd_prestamo[,"Mortgage"],
                                breaks = c(0, 1, 50, 100, 251),
                                labels = c(0, 1, 2, 3), right = FALSE)

# Omit NAs
bd_prestamo <- na.omit(bd_prestamo)
```

Por último, antes del tratamiento, se trasladan los valores numéricos a factores para su posterior procesamiento.

```r
# Convert into factors
bd_prestamo[, "Family"] <- as.factor(bd_prestamo[, "Family"])
bd_prestamo[, "Education"] <- as.factor(bd_prestamo[, "Education"])
```

Planteamiento del problema

Para poder examinar las posibles ventas cruzadas, se ha decidido evaluar todas las posibilidades de venta cruzada entre dos productos:

- Cuentas de valores (*Securities Accounts*) y Certificados de depósitos (*CD Accounts*).
- Cuentas de valores (*Securities Accounts*) y Servicios de banca en línea (*Online*).
- Cuentas de valores (*Securities Accounts*) y Tarjetas de crédito (*Credit Card*).
- Certificados de depósitos (*CD Accounts*) y Servicios de banca en línea (*Online*).
- Certificados de depósitos (*CD Accounts*) y Tarjetas de crédito (*Credit Card*).
- Servicios de banca en línea (*Online*) y Tarjetas de crédito (*Credit Card*).

Para ello se ha decidido unificar la información en las siguientes columnas:

- *Securities_CD* si *Securities Accounts* y *CD Accounts* son igual 1, entonces será 1, si una de las dos no lo es, entonces será 0.
- *Securities_Online* si *Securities Accounts* y *Online* son igual 1, entonces será 1, si una de las dos no lo es, entonces será 0.
- *Securities_CreditCard* si *Securities Accounts* y *Credit Card* son igual 1, entonces será 1, si una de las dos no lo es, entonces será 0.
- *CD_Online* si *CD Accounts* y *Online* son igual 1, entonces será 1, si una de las dos no lo es, entonces será 0.
- *CD_CreditCard* si *CD Accounts* y *Credit Card* son igual 1, entonces será 1, si una de las dos no lo es, entonces será 0.
- *Online_CreditCard* si *Online* y *Credit Card* son igual 1, entonces será 1, si una de las dos no lo es, entonces será 0.

Para realizar dicha funcionalidad sin repetir demasiado código, se ha decidido implementar la función **get_dataframe**, que dependiendo de la columna indicada añade determinada información al dataset.

```r
# Definition of min_max normalization function
get_dataframe <- function(x, data) {
  data_ready <- data

  if(x == 1){
    data_ready <- mutate(data_ready, Securities_CD = SecuritiesAccount + CDAccount)
    data_ready[, "Securities_CD"] <- ifelse(
      data_ready[, "Securities_CD"] == 2, 1, 0)
    data_ready[, "Securities_CD"] <- as.factor(
      data_ready[, "Securities_CD"])
  }else if(x == 2){
    data_ready <- mutate(data_ready, Securities_Online = SecuritiesAccount + Online)
    data_ready[, "Securities_Online"] <- ifelse(
      data_ready[, "Securities_Online"] == 2, 1, 0)
    data_ready[, "Securities_Online"] <- as.factor(
      data_ready[, "Securities_Online"])
  }else if(x == 3){
```

```
    data_ready <- mutate(data_ready, Securities_CreditCard = SecuritiesAccount + CreditCard)
    data_ready[, "Securities_CreditCard"] <- ifelse(
      data_ready[, "Securities_CreditCard"] == 2, 1, 0)
    data_ready[, "Securities_CreditCard"] <- as.factor(
      data_ready[, "Securities_CreditCard"])
  }else if(x == 4){
    data_ready <- mutate(data_ready, CD_Online = CDAccount + Online)
    data_ready[, "CD_Online"] <- ifelse(
      data_ready[, "CD_Online"] == 2, 1, 0)
    data_ready[, "CD_Online"] <- as.factor(
      data_ready[, "CD_Online"])
  }else if(x == 5){
    data_ready <- mutate(data_ready, CD_CreditCard = CDAccount + CreditCard)
    data_ready[, "CD_CreditCard"] <- ifelse(
      data_ready[, "CD_CreditCard"] == 2, 1, 0)
    data_ready[, "CD_CreditCard"] <- as.factor(
      data_ready[, "CD_CreditCard"])
  }else if(x == 6){
    data_ready <- mutate(data_ready, Online_CreditCard = Online + CreditCard)
    data_ready[, "Online_CreditCard"] <- ifelse(
      data_ready[, "Online_CreditCard"] == 2, 1, 0)
    data_ready[, "Online_CreditCard"] <- as.factor(
      data_ready[, "Online_CreditCard"])
  }

  data_ready[, "SecuritiesAccount"] <- NULL
  data_ready[, "CDAccount"] <- NULL
  data_ready[, "Online"] <- NULL
  data_ready[, "CreditCard"] <- NULL

  data_ready
}
```

Además, se han definido dos funciones para calcular el soporte y la confianza de los items que contengan las variables de venta cruzada anteriormente mencionadas, con el objetivo de ver la representatividad de estos casos en el conjunto total.

```
get_support <- function(x, data){
  if(x == 1){
    result <- ifelse(data[,"SecuritiesAccount"] == 1
                     | data[,"CDAccount"] == 1, 1, 0)
  }else if(x == 2){
    result <- ifelse(data[,"SecuritiesAccount"] == 1
                     | data[,"Online"] == 1, 1, 0)
  }else if(x == 3){
    result <- ifelse(data[,"SecuritiesAccount"] == 1
                     | data[,"CreditCard"] == 1, 1, 0)
  }else if(x == 4){
    result <- ifelse(data[,"CDAccount"] == 1
                     | data[,"Online"] == 1, 1, 0)
  }else if(x == 5){
    result <- ifelse(data[,"CDAccount"] == 1
                     | data[,"CreditCard"] == 1, 1, 0)
```

```
  }else if(x == 6){
    result <- ifelse(data[,"Online"] == 1
                     | data[,"CreditCard"] == 1, 1, 0)
  }else if(x == 0){
    result <- ifelse(bd_prestamo[, "SecuritiesAccount"] == 1
                     | bd_prestamo[, "CDAccount"] == 1
                     | bd_prestamo[, "Online"] == 1
                     | bd_prestamo[, "CreditCard"] == 1, 1, 0)
  }

  as.integer(table(result)[2])/nrow(data)
}

get_confidence <- function(x, data){
  if(x == 1){
    result <- ifelse(data[,"SecuritiesAccount"] == 1
                     & data[,"CDAccount"] == 1, 1, 0)
  }else if(x == 2){
    result <- ifelse(data[,"SecuritiesAccount"] == 1
                     & data[,"Online"] == 1, 1, 0)
  }else if(x == 3){
    result <- ifelse(data[,"SecuritiesAccount"] == 1
                     & data[,"CreditCard"] == 1, 1, 0)
  }else if(x == 4){
    result <- ifelse(data[,"CDAccount"] == 1
                     & data[,"Online"] == 1, 1, 0)
  }else if(x == 5){
    result <- ifelse(data[,"CDAccount"] == 1
                     & data[,"CreditCard"] == 1, 1, 0)
  }else if(x == 6){
    result <- ifelse(data[,"Online"] == 1
                     & data[,"CreditCard"] == 1, 1, 0)
  }else if(x == 0){
    result <- ifelse(bd_prestamo[, "SecuritiesAccount"] == 1
                     & bd_prestamo[, "CDAccount"] == 1
                     & bd_prestamo[, "Online"] == 1
                     & bd_prestamo[, "CreditCard"] == 1, 1, 0)
  }

  as.integer(table(result)[2])/nrow(data)
}
```

A continuación se procede a observar el soporte y la confianza de los casos a estudiar para examinar la venta cruzada y determinar unas reglas de asociación.

```
# Support of Securities Accounts and CD Accounts
sp1 <- paste(get_support(1, bd_prestamo), '%')
# Confidence of Securities Accounts and CD Accounts
cf1 <- paste(get_confidence(1, bd_prestamo), '%')
name1 <- '{Securities Accounts, CD Accounts}'

# Support of Securities Accounts and Online
sp2 <- paste(get_support(2, bd_prestamo), '%')
```

```r
# Confidence of Securities Accounts and Online
cf2 <- paste(get_confidence(2, bd_prestamo), '%')
name2 <- '{Securities Accounts, Online}'

# Support of Securities Accounts and Credit Card
sp3 <- paste(get_support(3, bd_prestamo), '%')
# Confidence of Securities Accounts and Credit Card
cf3 <- paste(get_confidence(3, bd_prestamo), '%')
name3 <- '{Securities Accounts, Credit Card}'

# Support of CD Accounts and Online
sp4 <- paste(get_support(4, bd_prestamo), '%')
# Confidence of CD Accounts and Online
cf4 <- paste(get_confidence(4, bd_prestamo), '%')
name4 <- '{CD Accounts, Online}'

# Support of CD Accounts and Credit Card
sp5 <- paste(get_support(5, bd_prestamo), '%')
# Confidence of CD Accounts and Credit Card
cf5 <- paste(get_confidence(5, bd_prestamo), '%')
name5 <- '{CD Accounts, Credit Card}'

# Support of Online and Credit Card
sp6 <- paste(get_support(6, bd_prestamo), '%')
# Confidence of Online and Credit Card
cf6 <- paste(get_confidence(6, bd_prestamo), '%')
name6 <- '{Online, Credit Card}'

# Create vectors of support, confidence and item names
support_vector = c(sp1, sp2, sp3, sp4, sp5, sp6)
confidence_vector = c(cf1, cf2, cf3, cf4, cf5, cf6)
name_vector = c(name1, name2, name3, name4, name5, name6)

# Create dataframe of support
support_df = data.frame(name_vector, support_vector)
names(support_df) <- c('Item', 'Soporte')

# Create dataframe of confidence
confidence_df = data.frame(name_vector, confidence_vector)
names(confidence_df) <- c('Item', 'Confianza')

# Print support dataframe
kable(support_df)
```

Item	Soporte
{Securities Accounts, CD Accounts}	0.13128003494976 %
{Securities Accounts, Online}	0.637177806902578 %
{Securities Accounts, Credit Card}	0.371996505024028 %
{CD Accounts, Online}	0.599388379204893 %
{CD Accounts, Credit Card}	0.304936653560507 %
{Online, Credit Card}	0.713630406290957 %

```r
# Print confidence dataframe
kable(confidence_df)
```

Item	Confianza
{Securities Accounts, CD Accounts}	0.026867627785059 %
{Securities Accounts, Online}	0.0633464394932285 %
{Securities Accounts, Credit Card}	0.027304499781564 %
{CD Accounts, Online}	0.0504587155963303 %
{CD Accounts, Credit Card}	0.0436871996505024 %
{Online, Credit Card}	0.17737003058104 %

Tal y como se puede observar, el número de casos en los que se realiza alguna venta cruzada es de aproximidamente el *6,44%* de los casos totales, por lo que a continuación procedemos a ver el soporte y la confianza del item que contiene todos los productos.

```r
# Support of All Services
per1 <- paste(get_support(0, bd_prestamo), '%')
# Confidence of All Services
per2 <- paste(get_confidence(0, bd_prestamo), '%')

# Create vector of percentages
percent_vector <- c(per1, per2)

# Create dataframe
total_df <- data.frame(c('Soporte', 'Confianza'), percent_vector)
names(total_df) <- c('Característica', 'Porcentaje')

# Print dataframe
kable(total_df)
```

Característica	Porcentaje
Soporte	0.743774574049803 %
Confianza	0.0163826998689384 %

Se puede observar que tan sólo en el *1,6%* de los casos se realiza una venta cruzada que contenga todos los productos evaluados, teniendo esta un soporte del *74,4%*, es decir, un *74,4%* de los clientes obtienen algún producto, pero para nuestro objetivo, nos interesan únicamente los casos donde se realiza la venta cruzada y ver las relaciones entre dichos clientes, para ello se restringiran posteriormente los resultados obtenidos.

Generación de reglas

El algoritmo escogido para la generación de reglas es el de Apriori, el cual se ha implementado utilizando el paquete arules, y utilizando la función apriori.

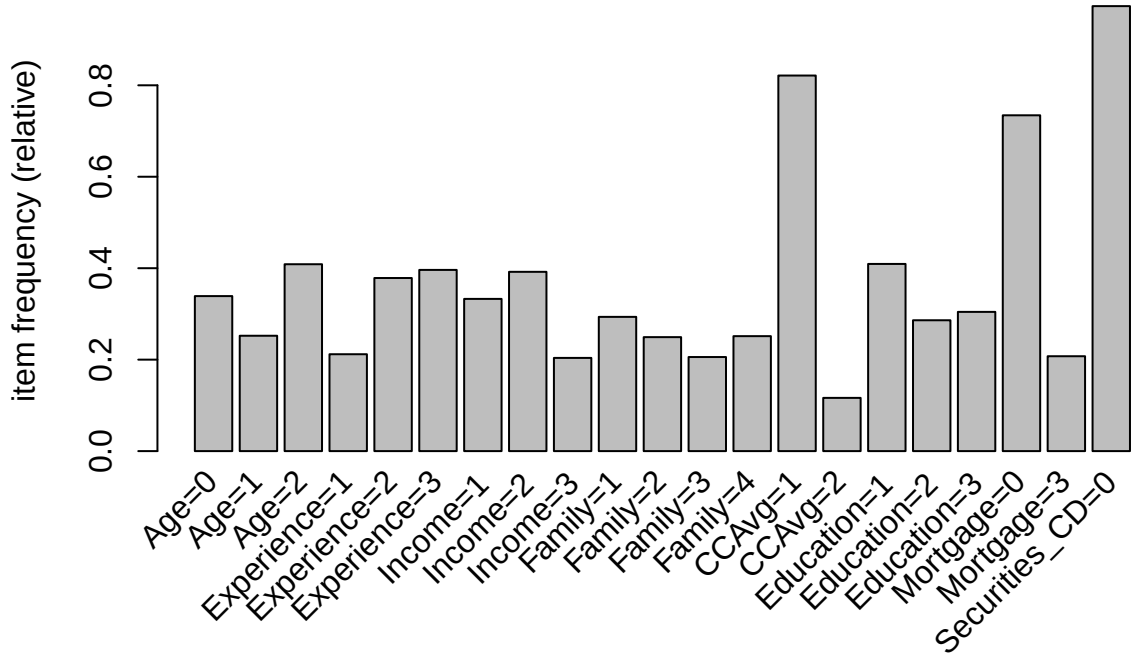
Además, se ha utilizado también la función itemFrequencyPlot, que permite ver en un histograma la frecuencia de un item, y la función interestMeasure, que permite obtener tanto el *coverage* como el valor de la Prueba exacta de Fisher

Por último, para visualizar las gráficas se ha utilizado el paquete arulesViz, que contiene los métodos para

poder visualizar gráficamente las diferentes reglas, los modelos que se han escogido son *Scatter plot*, *Graph plot* y *Parallel coordinates plot*.

Securities Accounts y CD Account

```
# Create dataframe using function
bd_prestamo_1 <- get_dataframe(1, bd_prestamo)
# Transform dataframe into transactions
bd_prestamo_1 <- as(bd_prestamo_1,"transactions")

# Plot item frequency
itemFrequencyPlot(bd_prestamo_1,support=0.1)
```



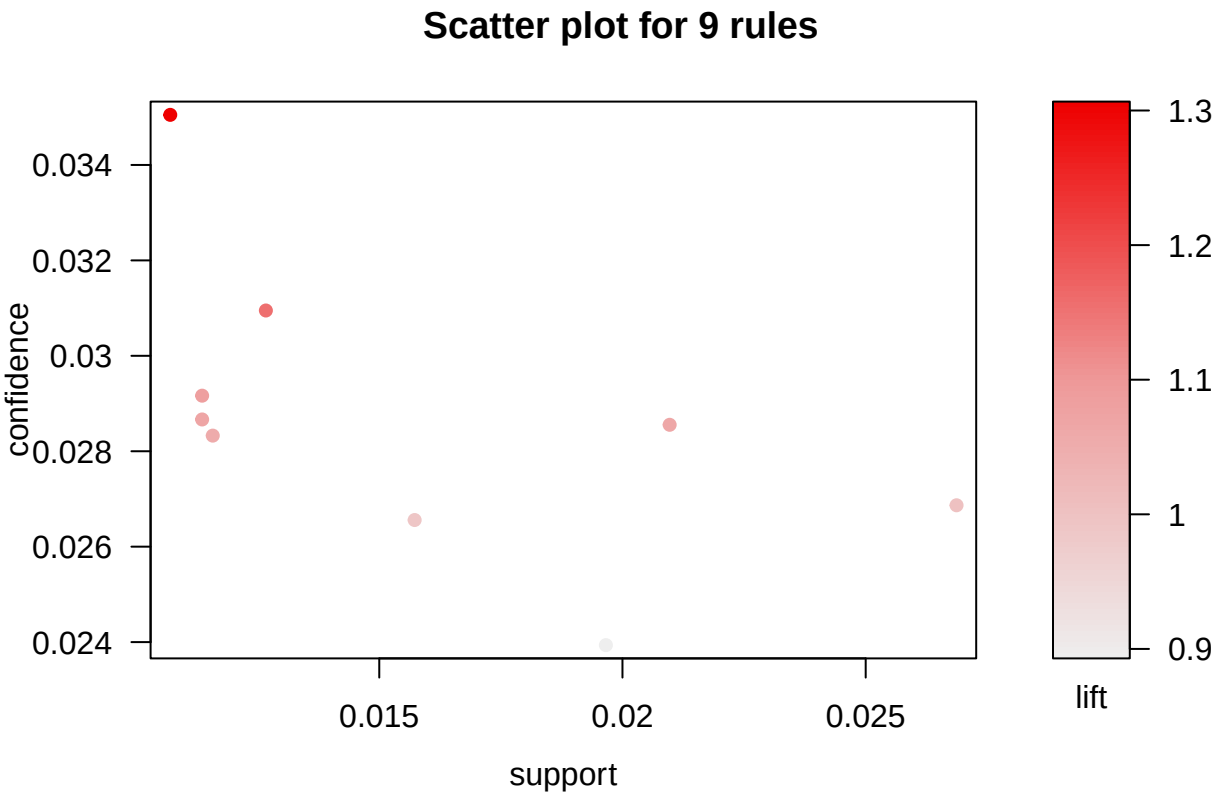

```
# Generate rules
rules <- apriori(bd_prestamo_1,
                 parameter = list(supp=0.01, conf=0.02, target="rules"),
                 appearance = list(rhs=c("Securities_CD=1"),
                                   default="lhs"),
                 control = list(verbose=F))

# Sort rules
rules.sorted <- sort(rules, by="lift")
# Inspect sorted rules
inspect(rules.sorted)
```

```
##     lhs                          rhs                support    confidence
## [1] {Education=1,Mortgage=0} => {Securities_CD=1} 0.01070336 0.03505007
## [2] {Education=1}            => {Securities_CD=1} 0.01266929 0.03094984
## [3] {Age=2,Experience=3}     => {Securities_CD=1} 0.01135867 0.02916433
## [4] {Experience=3}           => {Securities_CD=1} 0.01135867 0.02866593
## [5] {Mortgage=0}             => {Securities_CD=1} 0.02096986 0.02855443
## [6] {Age=2}                  => {Securities_CD=1} 0.01157711 0.02832710
## [7] {}                       => {Securities_CD=1} 0.02686763 0.02686763
## [8] {CCAvg=1,Mortgage=0}     => {Securities_CD=1} 0.01572739 0.02655847
## [9] {CCAvg=1}                => {Securities_CD=1} 0.01965924 0.02393617
##     coverage  lift      count
## [1] 0.3053735 1.3045466  49
## [2] 0.4093491 1.1519379  58
## [3] 0.3894714 1.0854821  52
## [4] 0.3962429 1.0669320  52
## [5] 0.7343818 1.0627820  96
## [6] 0.4086938 1.0543208  53
## [7] 1.0000000 1.0000000 123
## [8] 0.5921800 0.9884931  72
## [9] 0.8213194 0.8908926  90
```

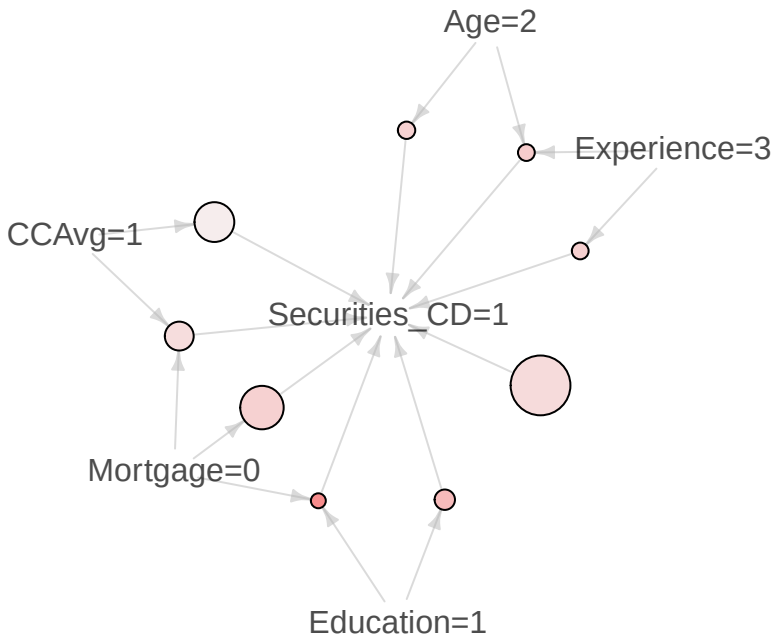
```r
# Coverage and Fishers Exact Test
interestMeasure(rules, measure = c("coverage", "fishersExactTest"),
                           transactions = bd_prestamo_1)
```

```
##    coverage fishersExactTest
## 1 1.0000000       1.00000000
## 2 0.3962429       0.30138317
## 3 0.4086938       0.33761585
## 4 0.4093491       0.09245633
## 5 0.7343818       0.14164948
## 6 0.8213194       0.99564013
## 7 0.3894714       0.24913394
## 8 0.3053735       0.01644606
## 9 0.5921800       0.60018356
```

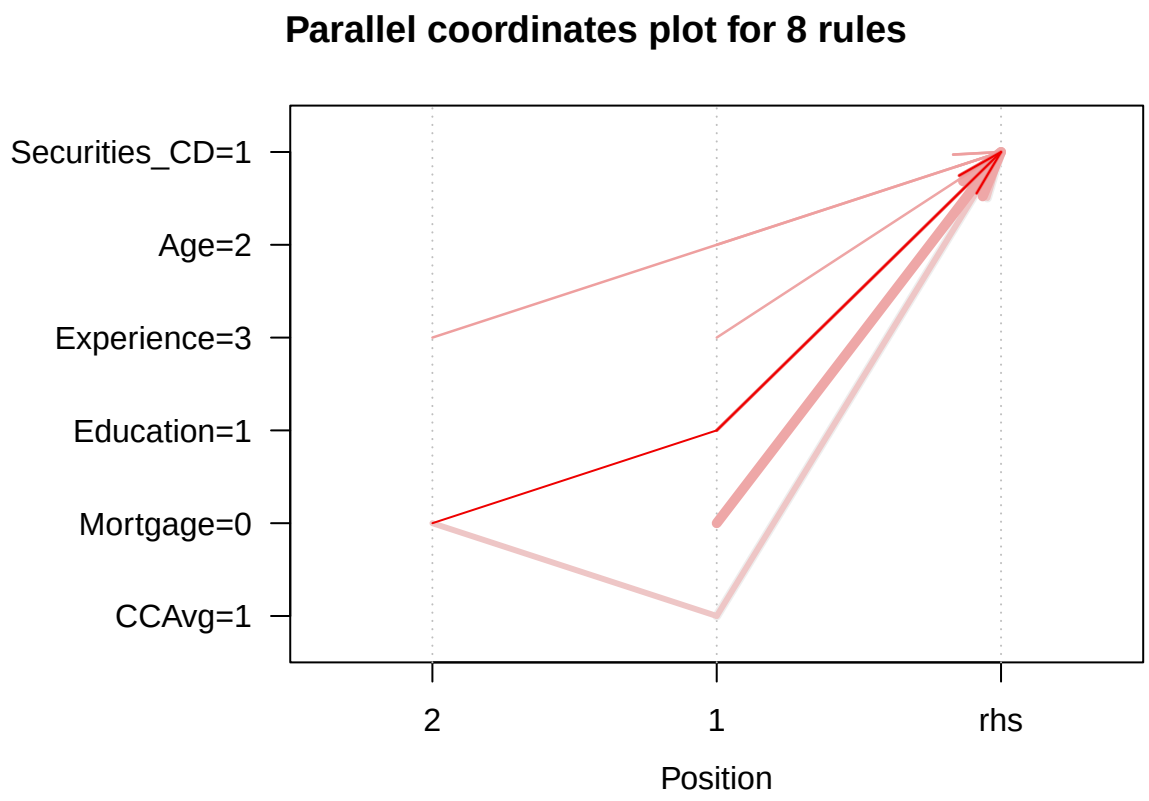
```r
# Scatter plot
plot(rules)
```




```
# Graph plot
plot(rules, method="graph")
```

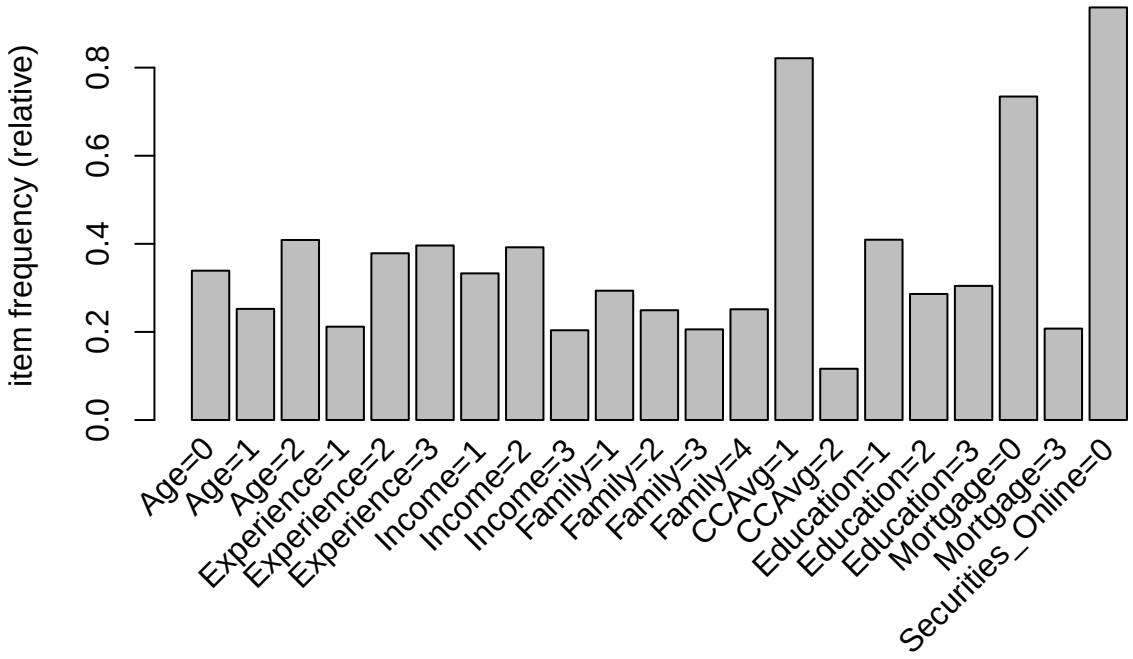



```
# Parallel coordinates plot
plot(rules, method="paracoord", control=list(reorder=TRUE))
```



Securities Accounts y Online

```
# Create dataframe using function
bd_prestamo_2 <- get_dataframe(2, bd_prestamo)
# Transform dataframe into transactions
bd_prestamo_2 <- as(bd_prestamo_2,"transactions")

# Plot item frequency
itemFrequencyPlot(bd_prestamo_2,support=0.1)
```




```
# Generate rules
rules <- apriori(bd_prestamo_2,
                 parameter = list(supp=0.02, conf=0.063, target="rules"),
                 appearance = list(rhs=c("Securities_Online=1"),
                                   default="lhs"),
                 control = list(verbose=F))

# Sort rules
rules.sorted <- sort(rules, by="lift")
# Inspect sorted rules
inspect(rules.sorted)
```

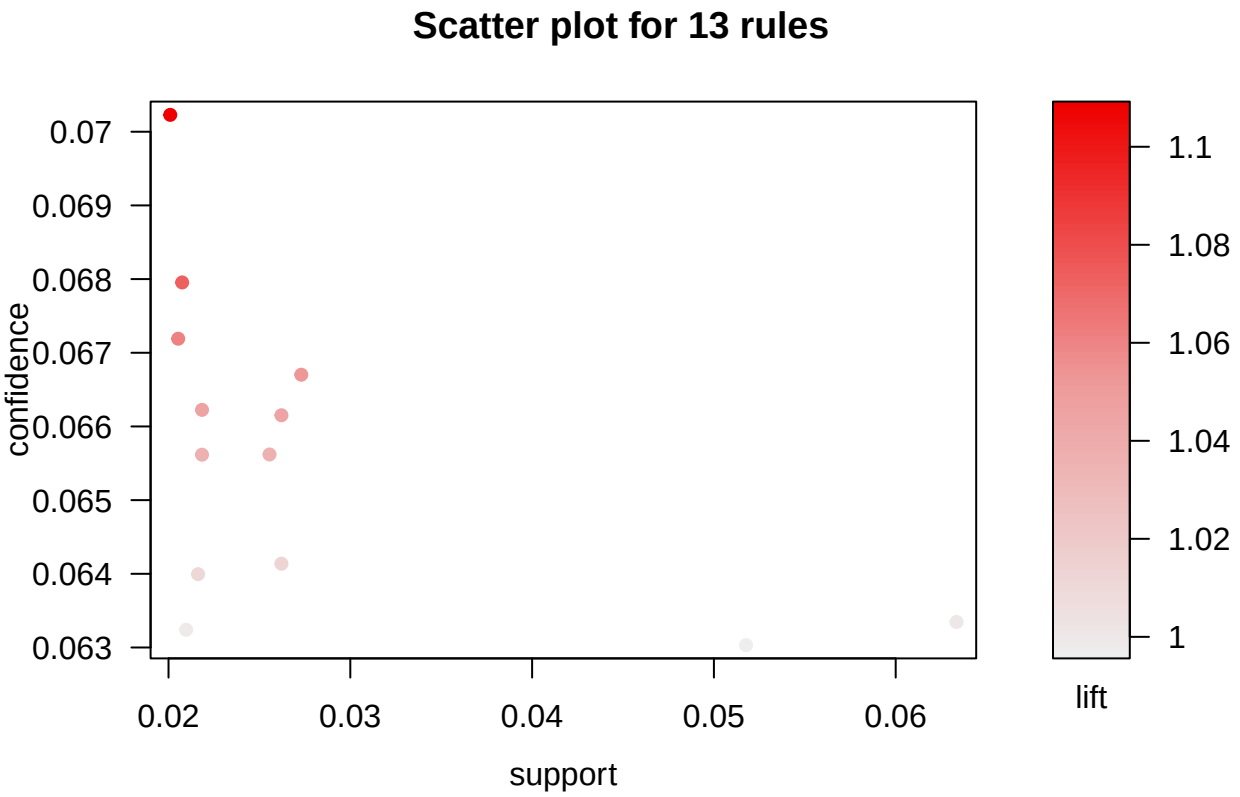
```
##      lhs                                rhs                     support
## [1]  {Education=2}                   => {Securities_Online=1} 0.02009611
## [2]  {Education=1,Mortgage=0}        => {Securities_Online=1} 0.02075142
## [3]  {CCAvg=1,Education=1}           => {Securities_Online=1} 0.02053298
## [4]  {Education=1}                   => {Securities_Online=1} 0.02730450
## [5]  {Income=1,CCAvg=1}              => {Securities_Online=1} 0.02184360
## [6]  {Experience=3}                  => {Securities_Online=1} 0.02621232
## [7]  {Age=2,Experience=3}            => {Securities_Online=1} 0.02555701
## [8]  {Income=1}                      => {Securities_Online=1} 0.02184360
## [9]  {Age=2}                         => {Securities_Online=1} 0.02621232
## [10] {Experience=3,CCAvg=1}          => {Securities_Online=1} 0.02162516
## [11] {}                              => {Securities_Online=1} 0.06334644
## [12] {Age=2,Experience=3,CCAvg=1}    => {Securities_Online=1} 0.02096986
## [13] {CCAvg=1}                       => {Securities_Online=1} 0.05176933
```

```
##      confidence coverage  lift      count
## [1]  0.07022901 0.2861512 1.1086496  92
## [2]  0.06795422 0.3053735 1.0727394  95
## [3]  0.06719085 0.3055920 1.0606887  94
## [4]  0.06670224 0.4093491 1.0529754 125
## [5]  0.06622517 0.3298384 1.0454442 100
## [6]  0.06615215 0.3962429 1.0442915 120
## [7]  0.06561974 0.3894714 1.0358868 117
## [8]  0.06561680 0.3328965 1.0358403 100
## [9]  0.06413683 0.4086938 1.0124772 120
## [10] 0.06399483 0.3379205 1.0102356  99
## [11] 0.06334644 1.0000000 1.0000000 290
## [12] 0.06324111 0.3315858 0.9983372  96
## [13] 0.06303191 0.8213194 0.9950348 237
```

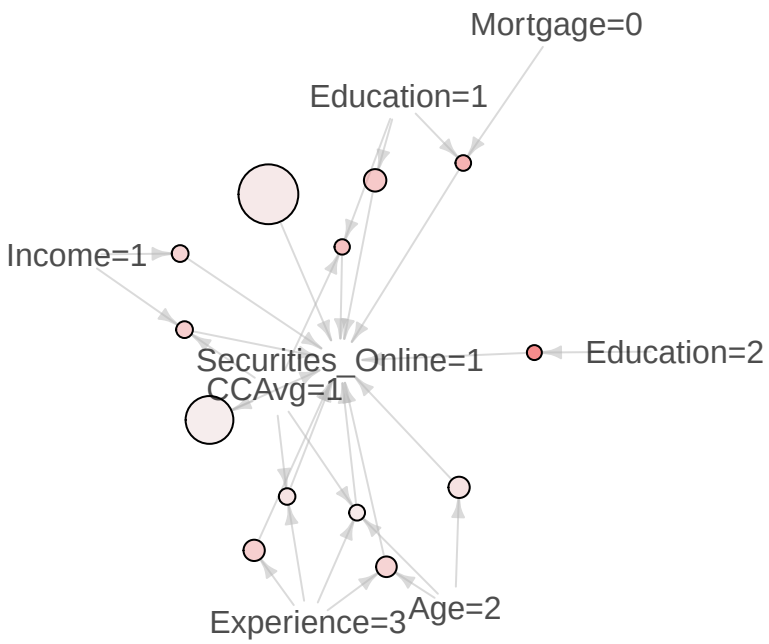
```
# Coverage and Fishers Exact Test
interestMeasure(rules, measure = c("coverage", "fishersExactTest"),
                            transactions = bd_prestamo_2)
```

```
##     coverage fishersExactTest
## 1  1.0000000        1.0000000
## 2  0.2861512        0.1269374
## 3  0.3328965        0.3496058
## 4  0.3962429        0.2837366
## 5  0.4086938        0.4506508
## 6  0.4093491        0.2370511
## 7  0.8213194        0.6103315
## 8  0.3298384        0.3081232
## 9  0.3894714        0.3280640
## 10 0.3379205        0.4718780
## 11 0.3053735        0.2160492
## 12 0.3055920        0.2588686
## 13 0.3315858        0.5313796
```

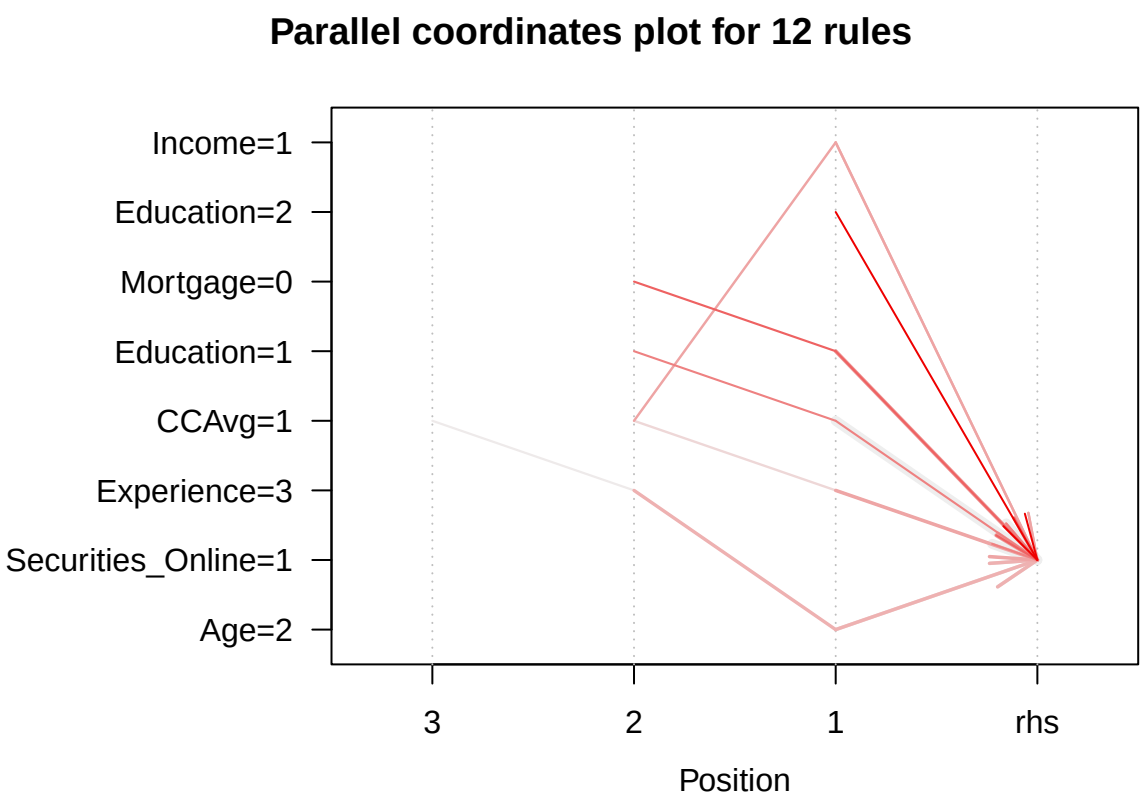
```
# Scatter plot
plot(rules)
```




```
# Graph plot
plot(rules, method="graph")
```

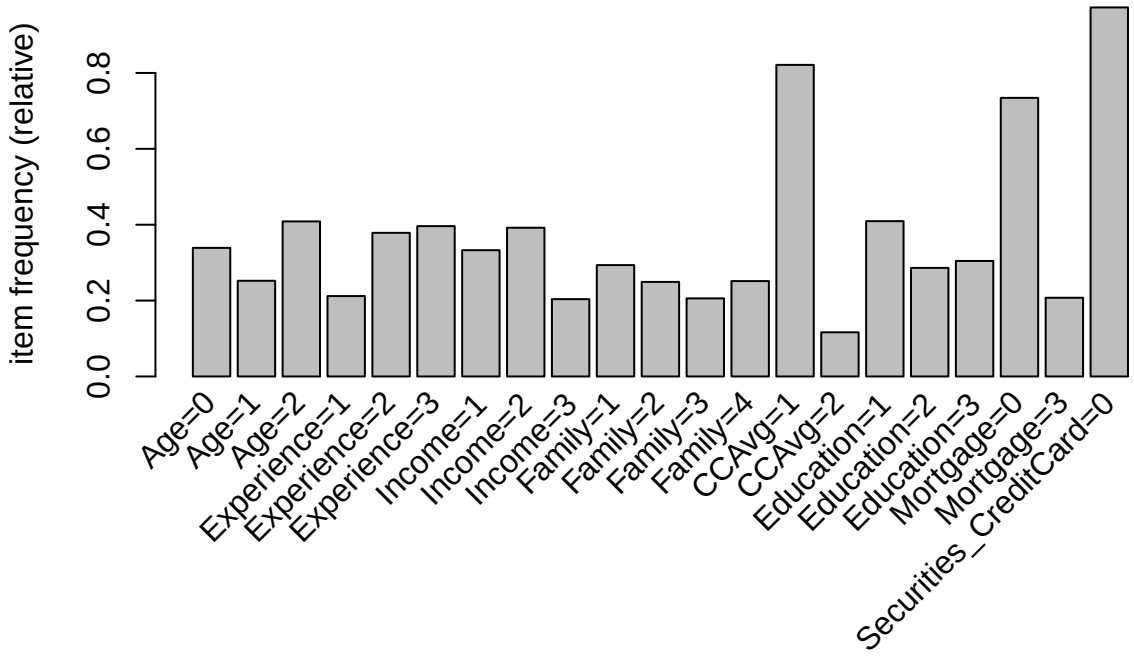



```
# Parallel coordinates plot
plot(rules, method="paracoord", control=list(reorder=TRUE))
```



Securities Accounts y Credit Card

```
# Create dataframe using function
bd_prestamo_3 <- get_dataframe(3, bd_prestamo)
# Transform dataframe into transactions
bd_prestamo_3 <- as(bd_prestamo_3,"transactions")

# Plot item frequency
itemFrequencyPlot(bd_prestamo_3,support=0.1)
```




```
# Generate rules
rules <- apriori(bd_prestamo_3,
                 parameter = list(supp=0.01, conf=0.025, target="rules"),
                 appearance = list(rhs=c("Securities_CreditCard=1"),
                                   default="lhs"),
                 control = list(verbose=F))

# Sort rules
rules.sorted <- sort(rules, by="lift")
# Inspect sorted rules
inspect(rules.sorted)
```

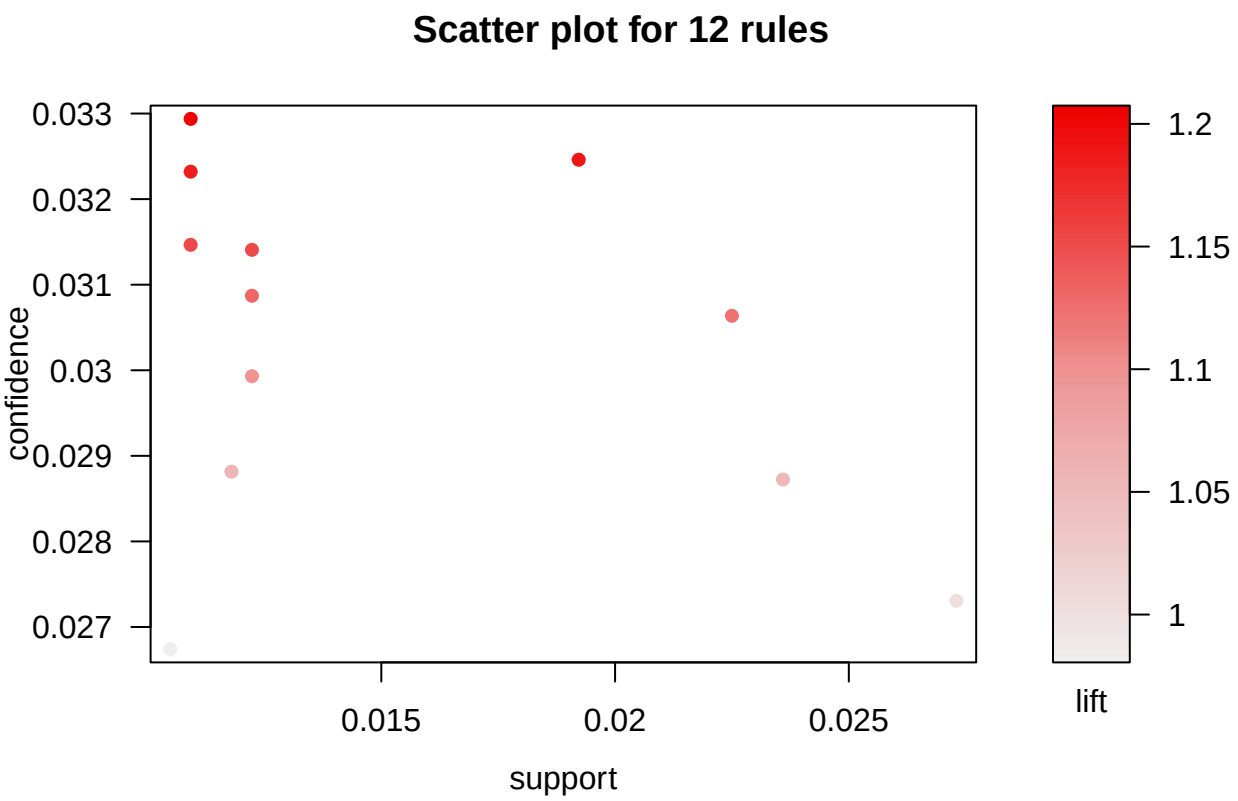
```
##      lhs                rhs                          support confidence  coverage      lift count
## [1]  {Age=2,
##       Experience=3,
##       CCAvg=1}       => {Securities_CreditCard=1} 0.01092180 0.03293808 0.3315858 1.2063241    50
## [2]  {CCAvg=1,
##       Mortgage=0}    => {Securities_CreditCard=1} 0.01922237 0.03246035 0.5921800 1.1888277    88
## [3]  {Experience=3,
##       CCAvg=1}       => {Securities_CreditCard=1} 0.01092180 0.03232062 0.3379205 1.1837104    50
## [4]  {Age=2,
##       CCAvg=1}       => {Securities_CreditCard=1} 0.01092180 0.03146633 0.3470948 1.1524229    50
## [5]  {Age=2,
##       Experience=3}  => {Securities_CreditCard=1} 0.01223242 0.03140774 0.3894714 1.1502771    56
## [6]  {Experience=3}  => {Securities_CreditCard=1} 0.01223242 0.03087100 0.3962429 1.1306196    56
## [7]  {Mortgage=0}    => {Securities_CreditCard=1} 0.02249891 0.03063653 0.7343818 1.1220321   103
```

```
## [8]  {Age=2}        => {Securities_CreditCard=1} 0.01223242 0.02993052 0.4086938 1.0961753    56
## [9]  {Education=1}  => {Securities_CreditCard=1} 0.01179554 0.02881537 0.4093491 1.0553340    54
## [10] {CCAvg=1}      => {Securities_CreditCard=1} 0.02359109 0.02872340 0.8213194 1.0519660   108
## [11] {}             => {Securities_CreditCard=1} 0.02730450 0.02730450 1.0000000 1.0000000   125
## [12] {Income=2}     => {Securities_CreditCard=1} 0.01048493 0.02674095 0.3920926 0.9793604    48
```

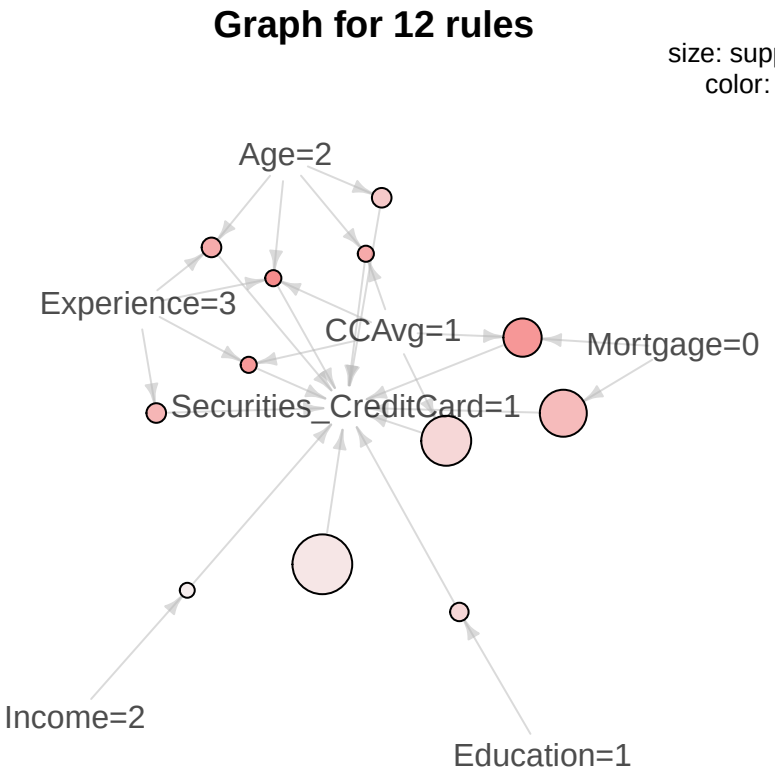
```r
# Coverage and Fishers Exact Test
interestMeasure(rules, measure = c("coverage", "fishersExactTest"),
                            transactions = bd_prestamo_3)
```

```
##     coverage fishersExactTest
## 1  1.0000000      1.000000000
## 2  0.3920926      0.608251772
## 3  0.3962429      0.134421712
## 4  0.4086938      0.207304988
## 5  0.4093491      0.332118631
## 6  0.7343818      0.011631881
## 7  0.8213194      0.124272679
## 8  0.3894714      0.103054181
## 9  0.3379205      0.083173939
## 10 0.3470948      0.122659005
## 11 0.5921800      0.005781949
## 12 0.3315858      0.061937245
```

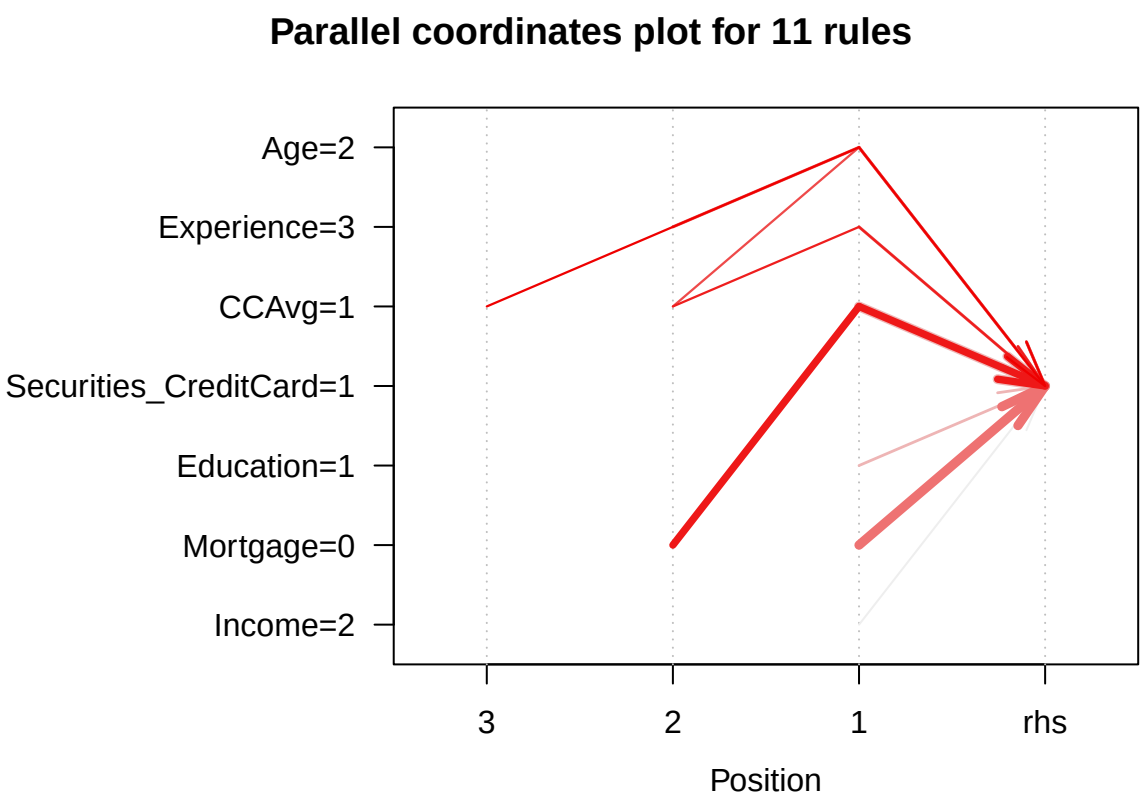
```r
# Scatter plot
plot(rules)
```



```
# Graph plot
plot(rules, method="graph")
```

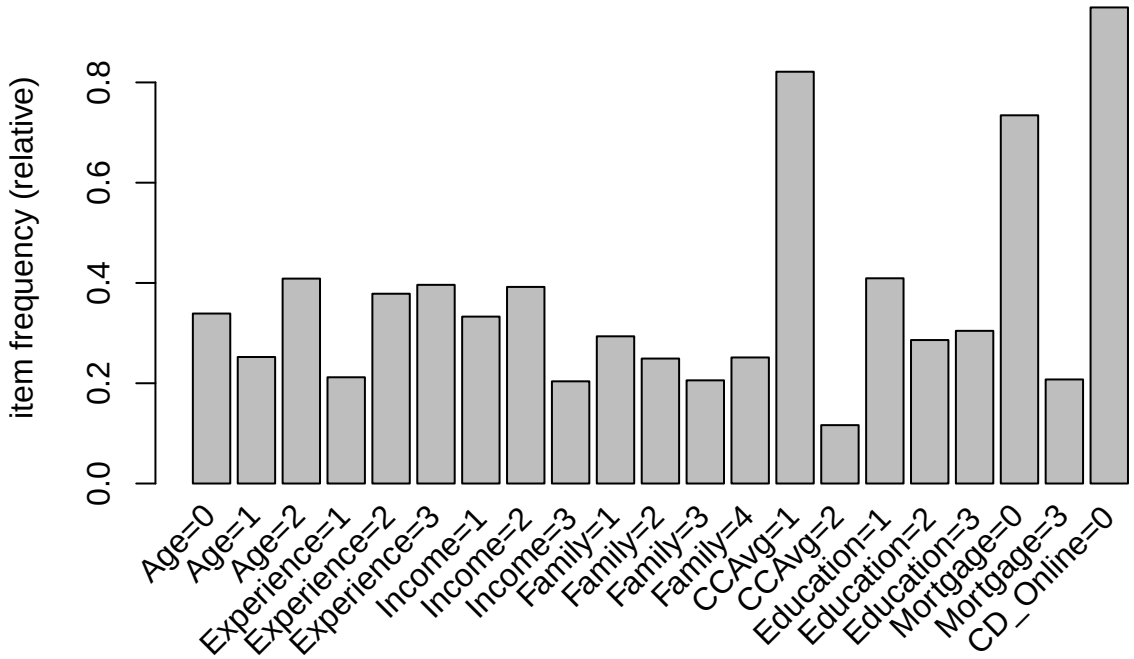



```
# Parallel coordinates plot
plot(rules, method="paracoord", control=list(reorder=TRUE))
```



CD Account y Online

```
# Create dataframe using function
bd_prestamo_4 <- get_dataframe(4, bd_prestamo)
# Transform dataframe into transactions
bd_prestamo_4 <- as(bd_prestamo_4,"transactions")

# Plot item frequency
itemFrequencyPlot(bd_prestamo_4,support=0.1)
```




```
# Generate rules
rules <- apriori(bd_prestamo_4,
                 parameter = list(supp=0.01, conf=0.06, target="rules"),
                 appearance = list(rhs=c("CD_Online=1"),
                                   default="lhs"),
                 control = list(verbose=F))

# Sort rules
rules.sorted <- sort(rules, by="lift")
# Inspect sorted rules
inspect(rules.sorted)
```

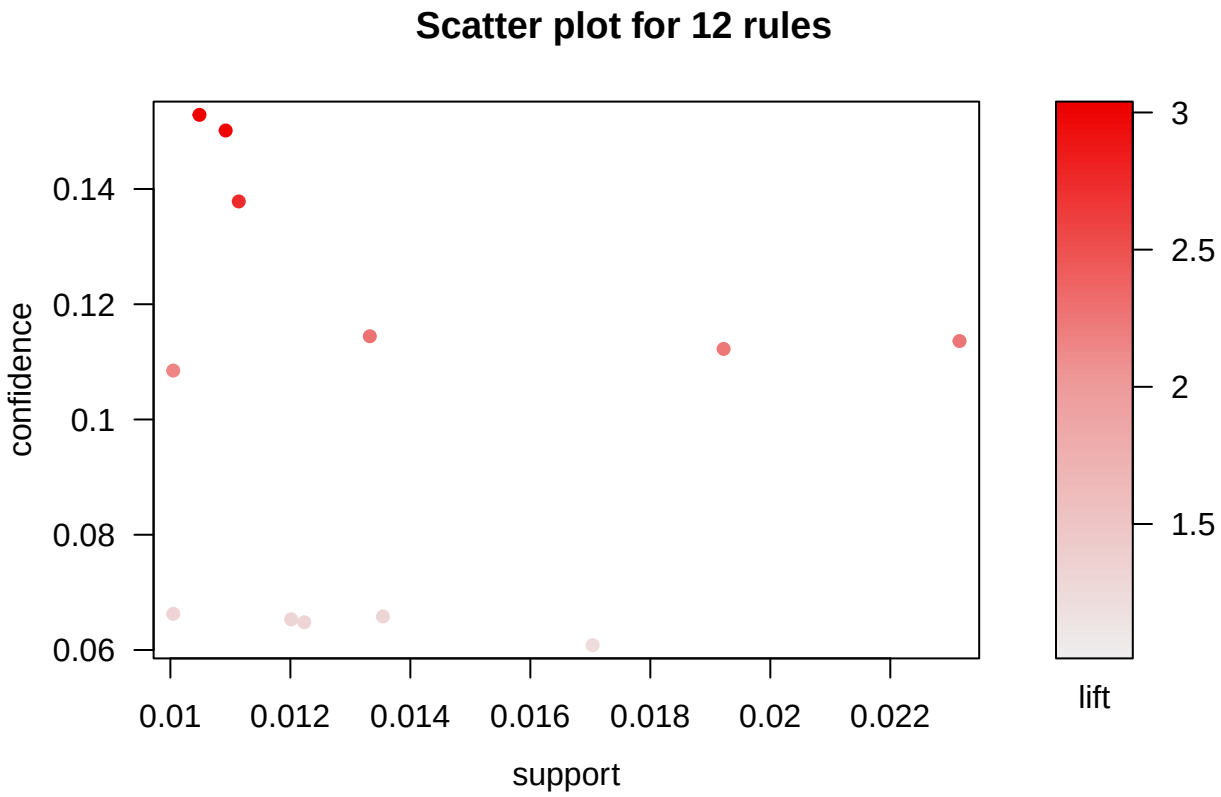
```
##      lhs                                    rhs           support    confidence
## [1]  {Experience=2,Income=3,Mortgage=0} => {CD_Online=1} 0.01048493 0.15286624
## [2]  {Income=3,CCAvg=2}                 => {CD_Online=1} 0.01092180 0.15015015
## [3]  {Experience=2,Income=3}            => {CD_Online=1} 0.01114024 0.13783784
## [4]  {CCAvg=2}                          => {CD_Online=1} 0.01332460 0.11444653
## [5]  {Income=3}                         => {CD_Online=1} 0.02315422 0.11361200
## [6]  {Income=3,Mortgage=0}              => {CD_Online=1} 0.01922237 0.11224490
## [7]  {CCAvg=2,Mortgage=0}               => {CD_Online=1} 0.01004806 0.10849057
## [8]  {Family=3,Mortgage=0}              => {CD_Online=1} 0.01004806 0.06628242
## [9]  {Family=3}                         => {CD_Online=1} 0.01354303 0.06581741
## [10] {Age=1,Experience=2,Mortgage=0}    => {CD_Online=1} 0.01201398 0.06532067
## [11] {Age=1,Mortgage=0}                 => {CD_Online=1} 0.01223242 0.06481481
## [12] {Experience=2,Mortgage=0}          => {CD_Online=1} 0.01703801 0.06084243
##      coverage   lift     count
```

```
## [1]  0.06858890 3.029531  48
## [2]  0.07273919 2.975703  50
## [3]  0.08082132 2.731695  51
## [4]  0.11642639 2.268122  61
## [5]  0.20380079 2.251583 106
## [6]  0.17125382 2.224490  88
## [7]  0.09261686 2.150086  46
## [8]  0.15159458 1.313597  46
## [9]  0.20576671 1.304381  62
## [10] 0.18392311 1.294537  55
## [11] 0.18872870 1.284512  56
## [12] 0.28003495 1.205786  78
```

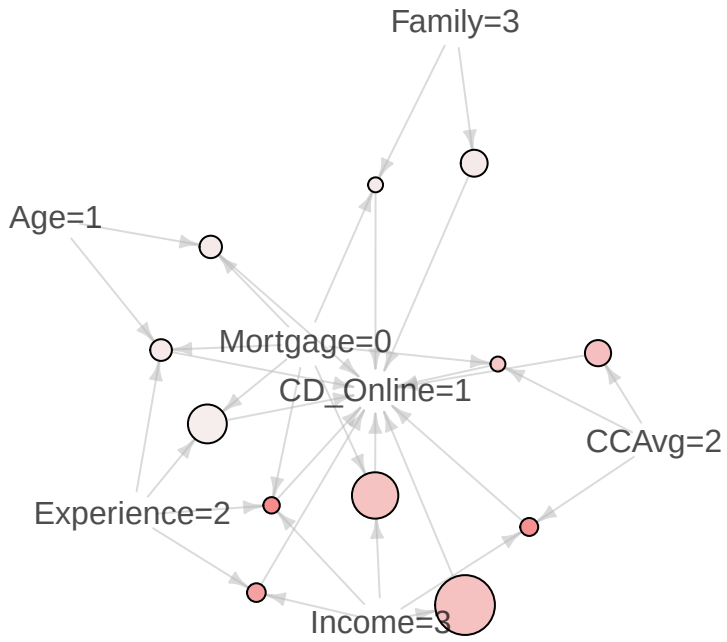
```
# Coverage and Fishers Exact Test
interestMeasure(rules, measure = c("coverage", "fishersExactTest"),
                            transactions = bd_prestamo_4)
```

```
##      coverage fishersExactTest
## 1  0.11642639     1.583818e-10
## 2  0.20380079     2.491546e-19
## 3  0.20576671     1.141424e-02
## 4  0.07273919     4.711144e-13
## 5  0.09261686     2.590480e-07
## 6  0.08082132     7.943761e-12
## 7  0.17125382     4.298078e-15
## 8  0.15159458     2.716411e-02
## 9  0.18872870     2.227783e-02
## 10 0.28003495     2.853570e-02
## 11 0.06858890     7.700152e-13
## 12 0.18392311     2.041999e-02
```

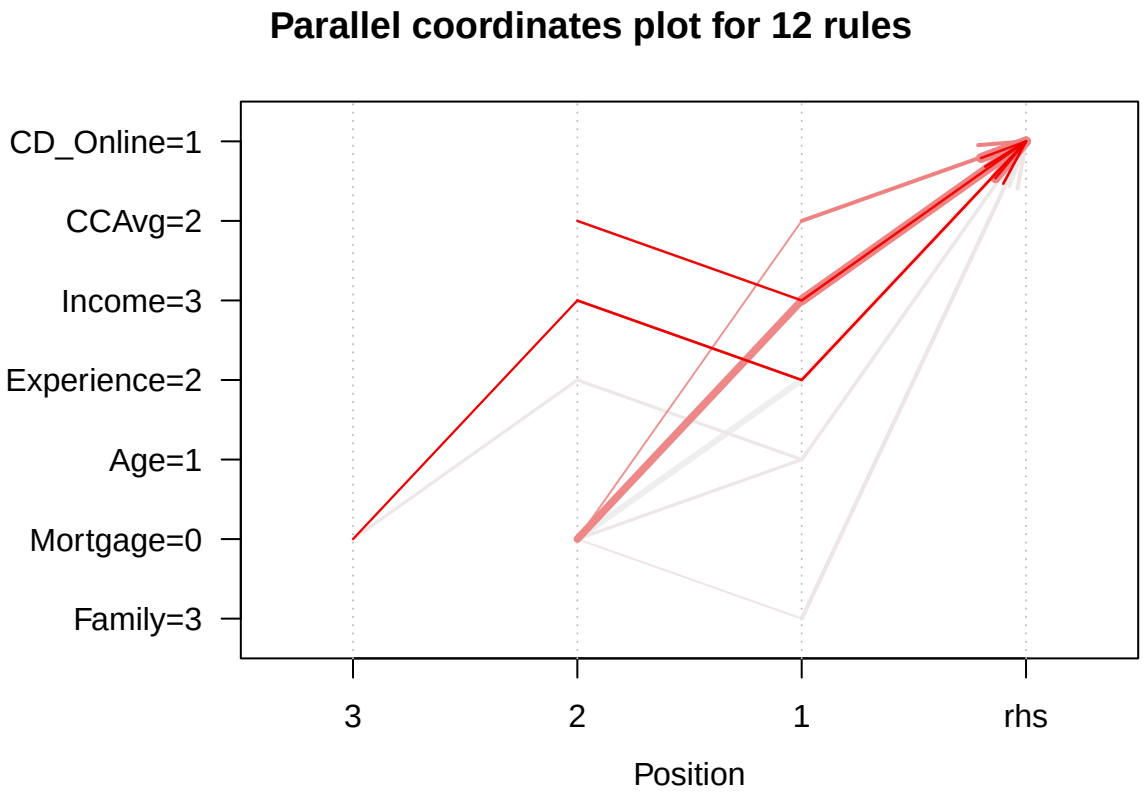
```
# Scatter plot
plot(rules)
```




```
# Graph plot
plot(rules, method="graph")
```

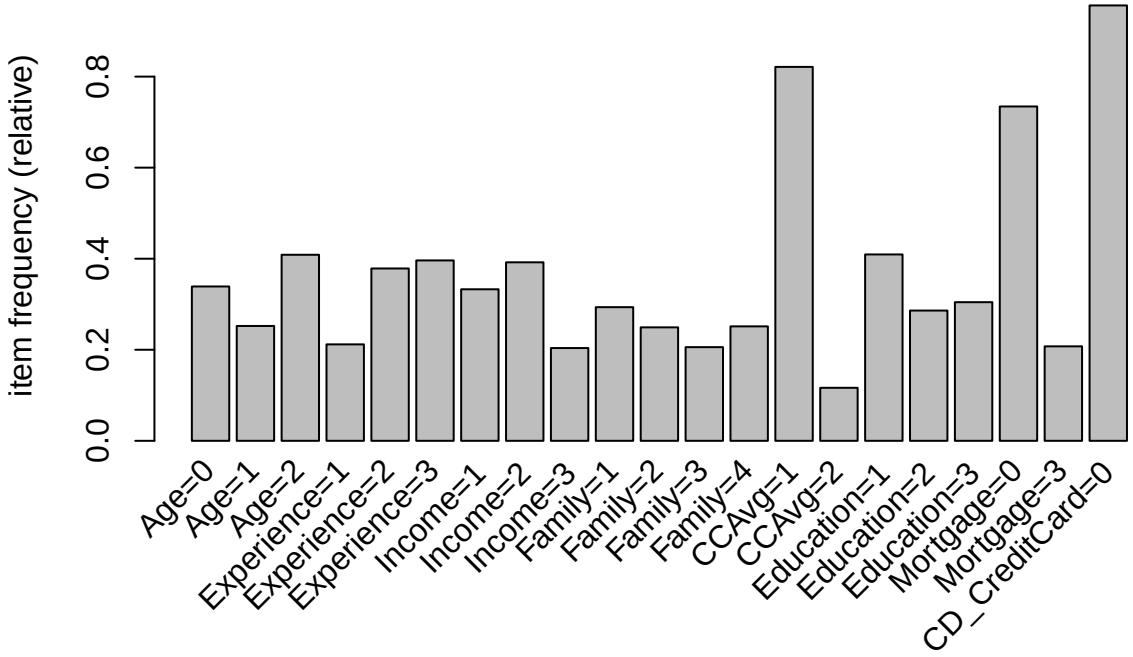



```
# Parallel coordinates plot
plot(rules, method="paracoord", control=list(reorder=TRUE))
```



CD Account y Credit Card

```
# Create dataframe using function
bd_prestamo_5 <- get_dataframe(5, bd_prestamo)
# Transform dataframe into transactions
bd_prestamo_5 <- as(bd_prestamo_5,"transactions")

# Plot item frequency
itemFrequencyPlot(bd_prestamo_5,support=0.1)
```




```
# Generate rules
rules <- apriori(bd_prestamo_5,
                 parameter = list(supp=0.01, conf=0.05, target="rules"),
                 appearance = list(rhs=c("CD_CreditCard=1"),
                                   default="lhs"),
                 control = list(verbose=F))

# Sort rules
rules.sorted <- sort(rules, by="lift")
# Inspect sorted rules
inspect(rules.sorted)
```

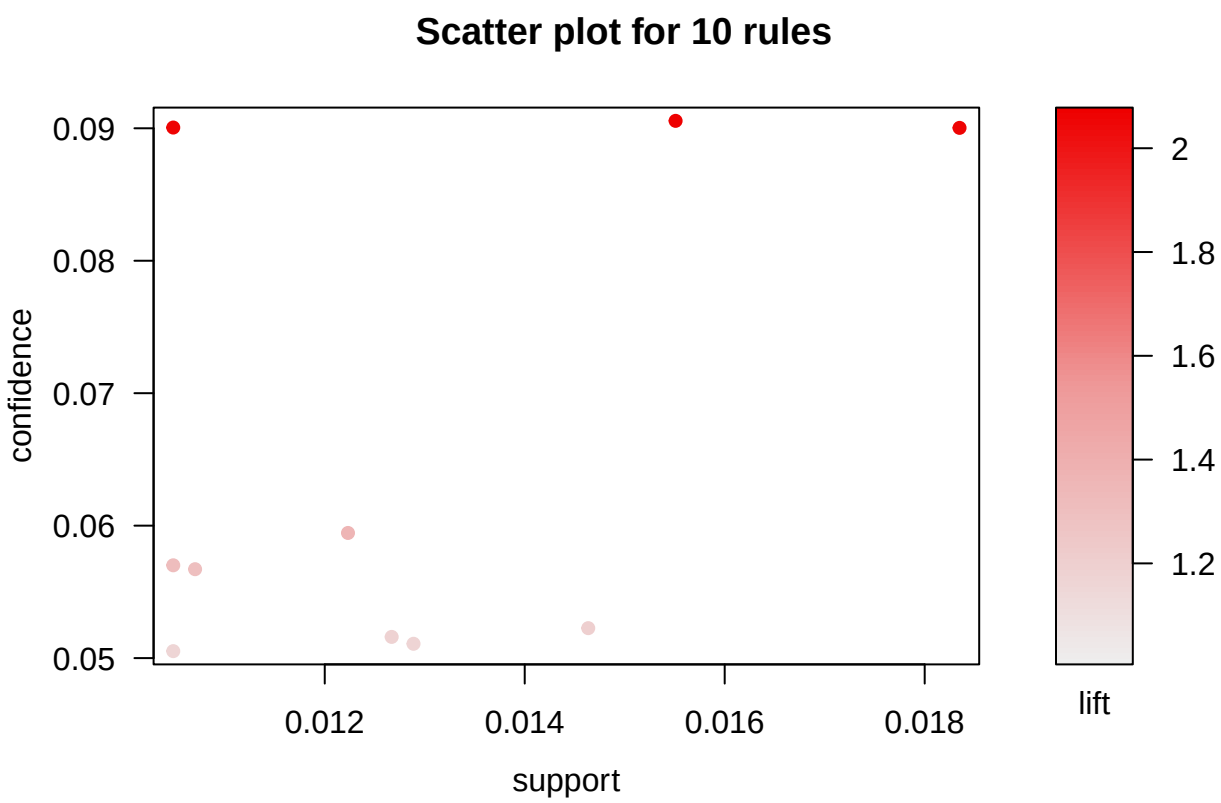
```
##      lhs                                  rhs                 support    confidence
## [1]  {Income=3,Mortgage=0}             => {CD_CreditCard=1} 0.01550896 0.09056122
## [2]  {CCAvg=2}                         => {CD_CreditCard=1} 0.01048493 0.09005629
## [3]  {Income=3}                        => {CD_CreditCard=1} 0.01834862 0.09003215
## [4]  {Family=3}                        => {CD_CreditCard=1} 0.01223242 0.05944798
## [5]  {Age=1,Experience=2,Mortgage=0} => {CD_CreditCard=1} 0.01048493 0.05700713
## [6]  {Age=1,Mortgage=0}                => {CD_CreditCard=1} 0.01070336 0.05671296
## [7]  {Experience=2,Mortgage=0}         => {CD_CreditCard=1} 0.01463521 0.05226209
## [8]  {Age=1,Experience=2}              => {CD_CreditCard=1} 0.01266929 0.05160142
## [9]  {Age=1}                           => {CD_CreditCard=1} 0.01288772 0.05108225
## [10] {Mortgage=3}                      => {CD_CreditCard=1} 0.01048493 0.05052632
##      coverage  lift     count
## [1]  0.1712538 2.072946 71
## [2]  0.1164264 2.061388 48
```

```
## [3]  0.2038008 2.060836 84
## [4]  0.2057667 1.360764 56
## [5]  0.1839231 1.304893 48
## [6]  0.1887287 1.298160 49
## [7]  0.2800349 1.196279 67
## [8]  0.2455221 1.181157 58
## [9]  0.2522936 1.169273 59
## [10] 0.2075142 1.156547 48
```

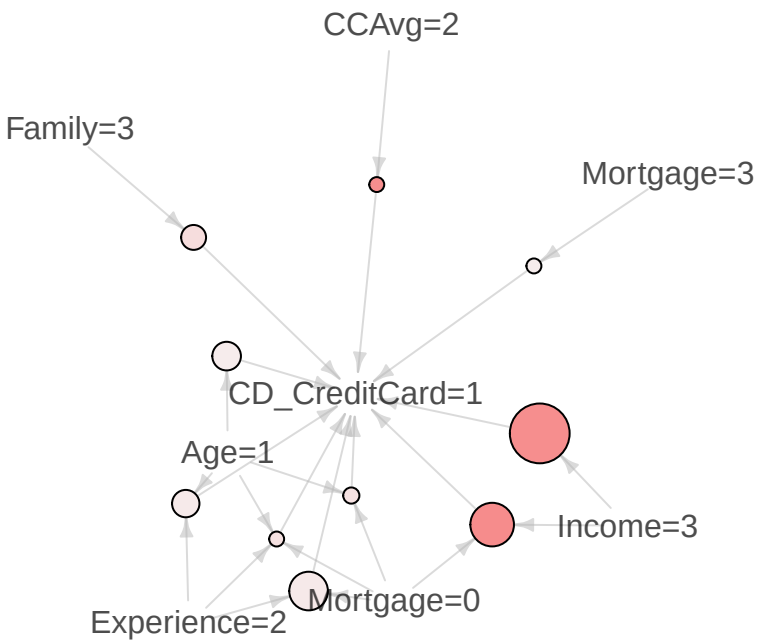
```r
# Coverage and Fishers Exact Test
interestMeasure(rules, measure = c("coverage", "fishersExactTest"),
                            transactions = bd_prestamo_5)
```

```
##     coverage fishersExactTest
## 1  0.1164264     4.097466e-07
## 2  0.2038008     8.863418e-13
## 3  0.2057667     6.381106e-03
## 4  0.2075142     1.428490e-01
## 5  0.2522936     9.177986e-02
## 6  0.1712538     1.145074e-10
## 7  0.2455221     8.090410e-02
## 8  0.1887287     2.603544e-02
## 9  0.2800349     4.726160e-02
## 10 0.1839231     2.539132e-02
```

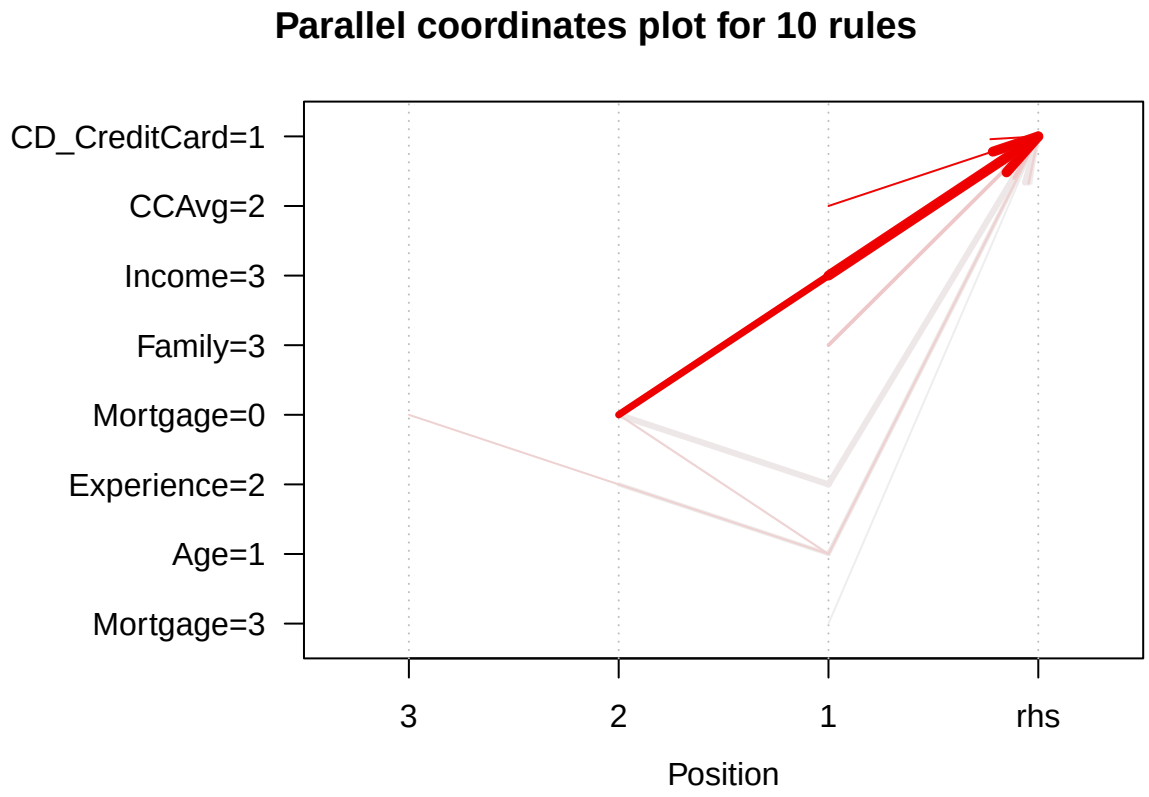
```r
# Scatter plot
plot(rules)
```




```
# Graph plot
plot(rules, method="graph")
```

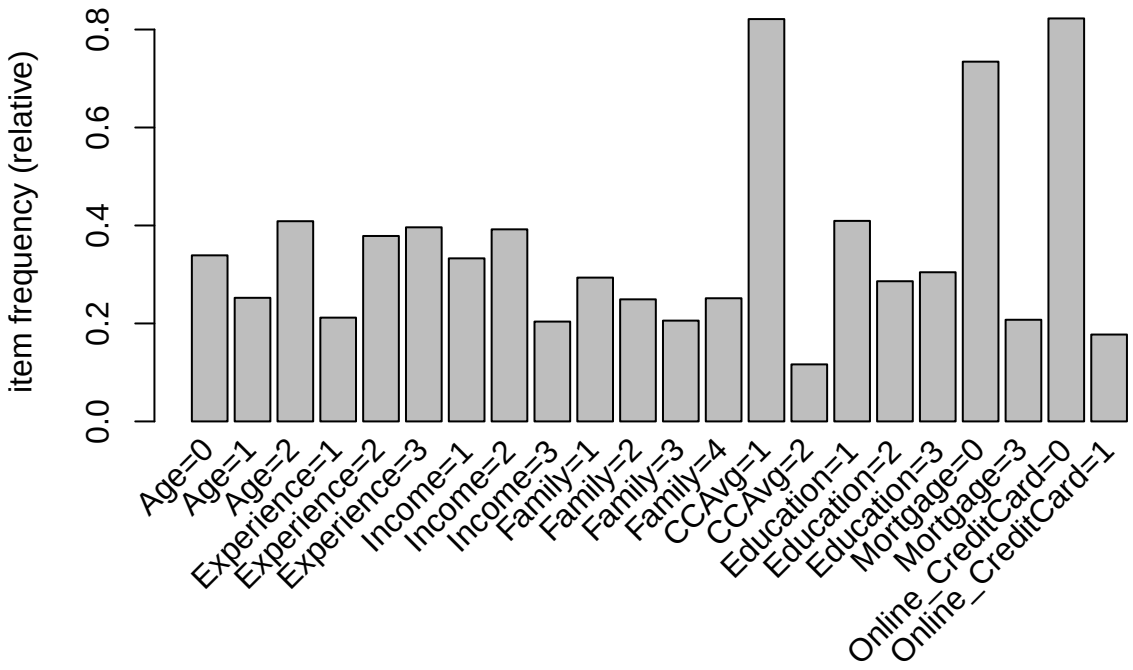



```
# Parallel coordinates plot
plot(rules, method="paracoord", control=list(reorder=TRUE))
```



Online y Credit Card

```
# Create dataframe using function
bd_prestamo_6 <- get_dataframe(6, bd_prestamo)
# Transform dataframe into transactions
bd_prestamo_6 <- as(bd_prestamo_6,"transactions")

# Plot item frequency
itemFrequencyPlot(bd_prestamo_6,support=0.1)
```




```
# Generate rules
rules <- apriori(bd_prestamo_6,
                 parameter = list(supp=0.02, conf=0.2, target="rules"),
                 appearance = list(rhs=c("Online_CreditCard=1"),
                                   default="lhs"),
                 control = list(verbose=F))

# Sort rules
rules.sorted <- sort(rules, by="lift")
# Inspect sorted rules
inspect(rules.sorted)
```

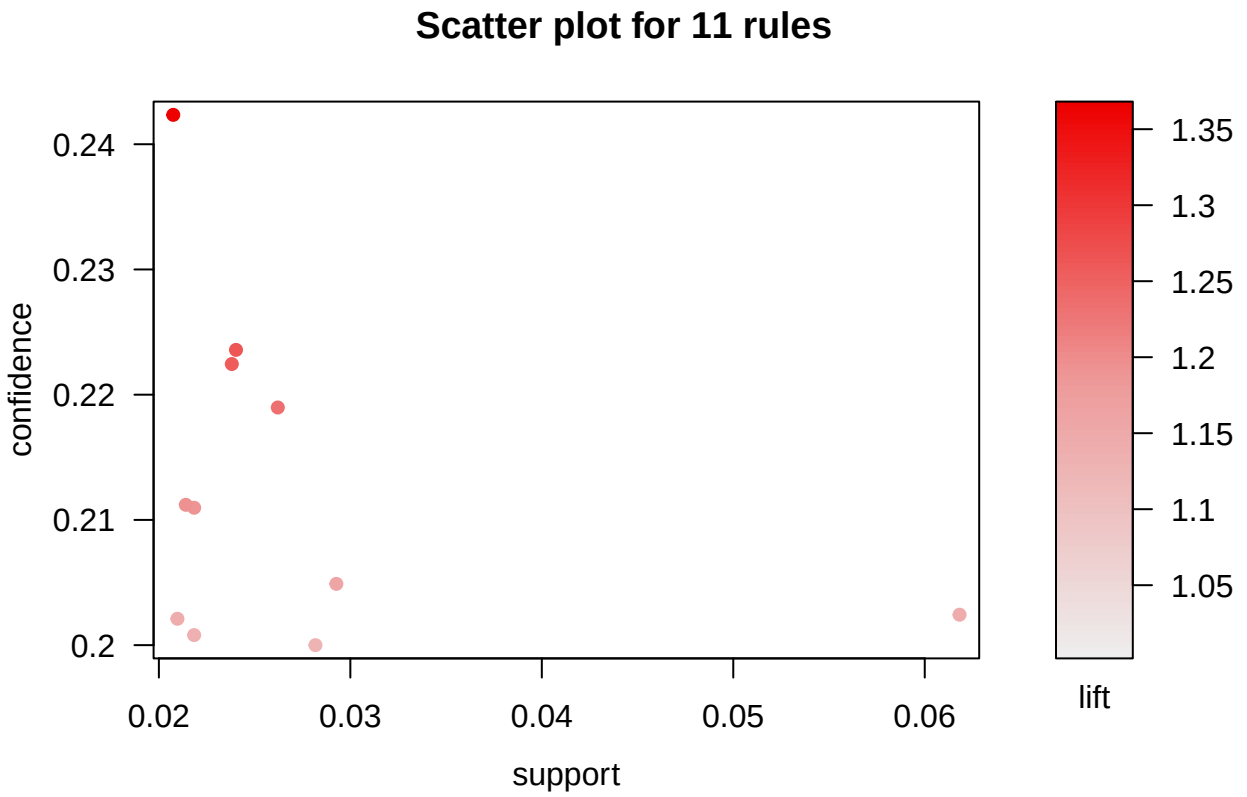
```
##      lhs               rhs                      support confidence   coverage     lift count
## [1]  {Family=4,
##       Education=1}  => {Online_CreditCard=1} 0.02075142  0.2423469 0.08562691 1.366335    95
## [2]  {Income=1,
##       Education=1}  => {Online_CreditCard=1} 0.02402796  0.2235772 0.10747051 1.260513   110
## [3]  {Income=1,
##       CCAvg=1,
##       Education=1}  => {Online_CreditCard=1} 0.02380952  0.2224490 0.10703364 1.254152   109
## [4]  {Experience=2,
##       Education=1,
##       Mortgage=0}   => {Online_CreditCard=1} 0.02621232  0.2189781 0.11970293 1.234583   120
## [5]  {Age=0,
##       CCAvg=1,
##       Education=1}  => {Online_CreditCard=1} 0.02140673  0.2112069 0.10135430 1.190770    98
```

```
## [6]  {Age=0,
##       Education=1,
##       Mortgage=0}   => {Online_CreditCard=1} 0.02184360  0.2109705 0.10353866 1.189437   100
## [7]  {Age=0,
##       Education=1}  => {Online_CreditCard=1} 0.02927042  0.2048930 0.14285714 1.155172   134
## [8]  {Education=1,
##       Mortgage=0}   => {Online_CreditCard=1} 0.06181739  0.2024320 0.30537353 1.141298   283
## [9]  {Family=1,
##       Education=1,
##       Mortgage=0}   => {Online_CreditCard=1} 0.02096986  0.2021053 0.10375710 1.139456    96
## [10] {Income=2,
##       Education=1,
##       Mortgage=0}   => {Online_CreditCard=1} 0.02184360  0.2008032 0.10878113 1.132115   100
## [11] {Experience=3,
##       Income=1}     => {Online_CreditCard=1} 0.02817824  0.2000000 0.14089122 1.127586   129
```

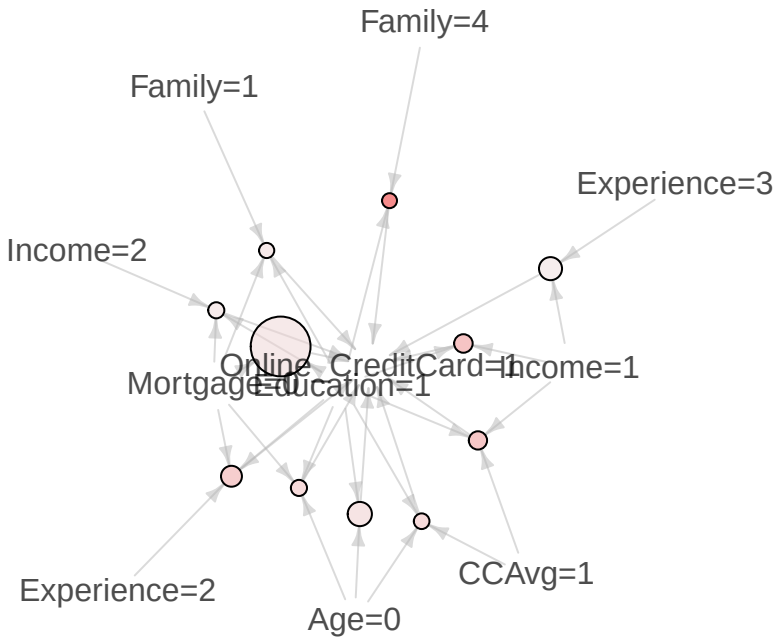
```
# Coverage and Fishers Exact Test
interestMeasure(rules, measure = c("coverage", "fishersExactTest"),
                            transactions = bd_prestamo_6)
```

```
##      coverage fishersExactTest
## 1  0.08562691     0.0004215497
## 2  0.14089122     0.0598486393
## 3  0.10747051     0.0033151190
## 4  0.14285714     0.0278430030
## 5  0.30537353     0.0020030911
## 6  0.10375710     0.0782394412
## 7  0.10703364     0.0041128780
## 8  0.10353866     0.0267284232
## 9  0.10135430     0.0273787059
## 10 0.11970293     0.0045909813
## 11 0.10878113     0.0839325880
```

```
# Scatter plot
plot(rules)
```




```
# Graph plot
plot(rules, method="graph")
```




```
# Parallel coordinates plot
plot(rules, method="paracoord", control=list(reorder=TRUE))
```

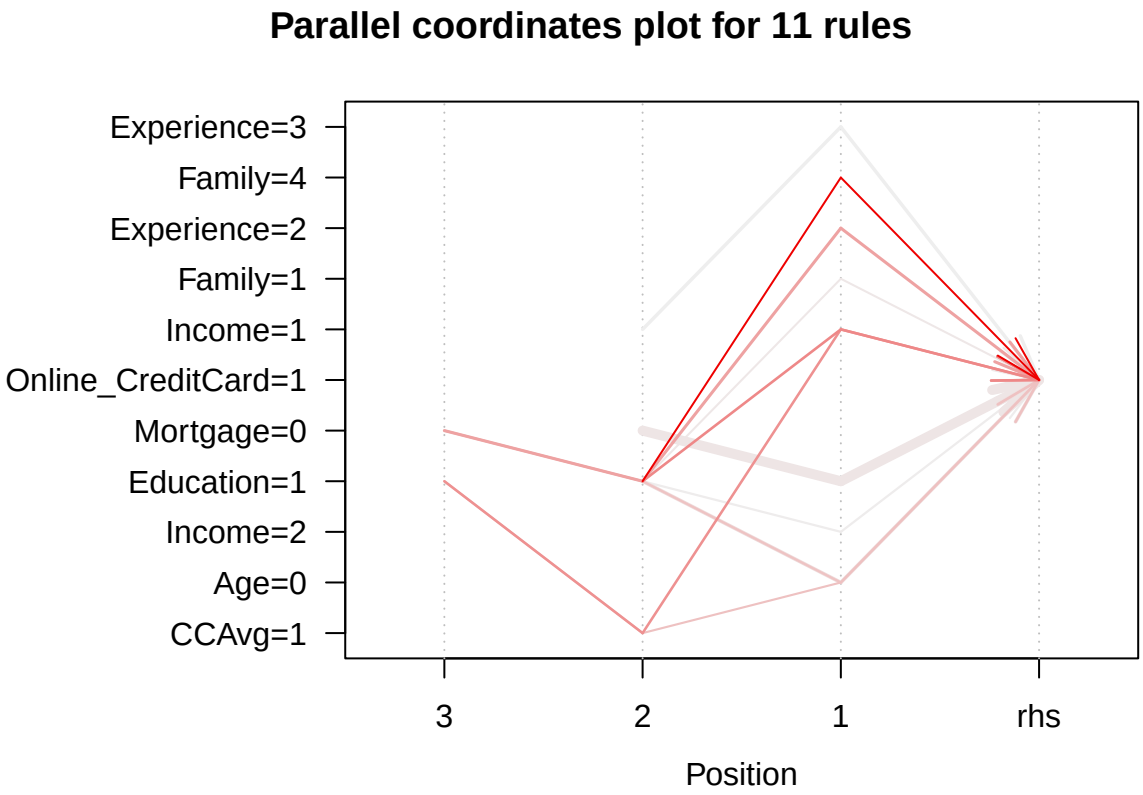


Tabla de resultados

Servicio	Edad	Exp. laboral	Educación	Ingresos	Media gasto con Tarjeta	Fam.	Hipoteca
Cuentas de valores y Certificados de depósitos	50-70 años	25-45 años	Sin graduado	*¿?*	*¿?*	*¿?*	0 €
Cuentas de valores y Banca en línea	50-70 años	25-45 años	Sin graduado - Graduado	20-49 mil €	0.1-2.9 mil €	*¿?*	0 €
Cuentas de valores y Tarjetas de crédito	50-70 años	25-45 años	Sin graduado	*¿?*	0.1-2.9 mil €	*¿?*	0 €
Certificados de depósitos y Banca en línea	40-49 años	10-24 años	Sin graduado	100-185 mil €	3.0-5.9 mil €	3	0 €
Certificados de depósitos y Tarjetas de crédito	40-49 años	10-24 años	*¿?*	100-185 mil €	3.0-5.9 mil €	3	0 €
Banca en línea y Tarjetas de crédito	20-39 años	10-24 años	Sin graduado	20-49 mil €	0.1-2.9 mil €	4	0 €

Conclusiones

Tras realizar la generación de reglas y analizar los reglas obtenidas, se ha realizado un análisis en el que se llegan a las siguientes conclusiones:

- Los factores más comunes encontrados son los siguientes:
  - Clientes que no se han graduado.
  - Clientes que no poseen hipoteca.
- Los factores que se consideran menos son los siguientes:
  - Integrantes de la familia del miembro.
  - Clientes con hipoteca.
  - Clientes con graduado.
- Para enfocar la campaña de cara a realizar una venta cruzada, habría que considerar los siguientes perfiles:
  - *Cuentas de valores* y *Certificados de depósitos*: Se busca un perfil de una persona mayor, con bastante experiencia laboral.
  - *Cuentas de valores* y *Banca en línea*: Se busca un perfil de una persona mayor, con bastante experiencia laboral, y con un salario medio, pero con pocos gastos a través de la tarjeta de crédito, por lo que se entiende que es un perfil que compra lo necesario.
  - *Cuentas de valores* y *Tarjetas de créditos*: Se busca un perfil de una persona mayor, con bastante experiencia laboral, con pocos gastos a través de tarjeta de crédito.
  - *Certificados de depósitos* y *Banca en línea*: Se busca un perfil de un adulto, con una considerable experiencia laboral y con una renta alta, que además realice unos gastos considerables.
  - *Certificados de depósitos* y *Tarjeta de crédito*: Se busca un perfil de un adulto, con una considerable experiencia laboral y con una renta alta, que además realice unos gastos considerables. Por media el número de integrantes de su familia es de 3 miembros.
  - *Banca en línea* y *Tarjeta de crédito*: Se busca un perfil de un adulto joven, con una considerable experiencia laboral y con una renta baja, y por lo tanto unos gastos bajos, con familia normalmente de 4 integrantes.

Una vez consideradas las anteriores anotaciones, se llega a la conclusión de que la campaña a realizar debe enfocarse principalmente en perfiles de clientes sin hipoteca y que no se hayan graduado, y teniendo en cuenta toda la información obtenida se propone una campaña que siga las siguientes instrucciones:

- Ofrecer una promoción que consista en vender servicios de *Cuentas de valores*, junto con *Banca en línea* o *Tarjetas de crédito*, aunque pudieran ofertarse los tres en conjunto, a clientes que sean personas mayores, premiando aquellos que no realizan grandes gastos mediante la tarjeta de crédito y con una renta baja o media.
- Ofrecer una promoción que consista en vender servicios de *Certificados de depósitos*, junto con *Banca en línea* y *Tarjetas de crédito* a clientes adultos con una alta renta y que realicen unos gastos considerables.
- Ofrecer una promoción que consista en vender servicios de *Banca en línea* y *Tarjetas de crédito* a clientes jóvenes, que tengan una renta considerable y con un gasto bajo, priorizando aquellos que posean familias más numerosas.
- Ofrecer una promoción que consista en vender servicios de *Cuentas de valores* y *Certificados de depósitos* a clientes mayores, que basándonos en experiencias previas, no posean una renta alta y realicen pocos gastos.