

Prevención del fraude y blanqueo de capitales en entidades financieras.

Juan Luis Espinoza Lopez

28/05/2023

## 0 Preambulo

Desde la llegada de Internet ylas nuevas tecnologias, la revolución digital ha aumentado y se ha infiltrado en todos los aspectos de nuestras vidas. Una de las revoluciones digitales más importantes ocurrió en el sistema financiero y especialmente en la transacción de dinero a alguien de cualquier parte del mundo digitalmente. Las transacciones digitales se han convertido en parte de la vida diaria, como comprar un producto en línea, enviar dinero a amigos, depositar efectivo en una cuenta bancaria, fines de inversión, etc. Tenían muchos beneficios y allanaron el camino para actividades fraudulentas. La gente comenzó a usar medios de transacciones de dinero digital para lavar dinero y hacer parecer que el dinero proviene de una fuente legal. Además mediante estas tramas delictivas es bien sabido que es una práctica común a la hora de financiar el terrorismo.

## 1 Lavado de dinero

### 1.1 Que signfica el blanqueo de capitales ?

El lavado de dinero es el proceso a través del cual es encubierto el origen de los fondos generados mediante el ejercicio de algunas actividades ilegales (siendo las más comunes, tráfico de drogas o estupefacientes, contrabando de armas, corrupción, fraude, trata de personas, prostitución, extorsión, piratería, evasión fiscal y terrorismo). El objetivo de la operación, que generalmente se realiza en varios niveles, consiste en hacer que los fondos o activos obtenidos a través de actividades ilícitas aparezcan como el fruto de actividades legítimas y circulen sin problema en el sistema financiero.

### 1.2 Historia del blanqueo de capitales

Estados Unidos fue una de las primeras naciones en introducir leyes contra el lavado de dinero con la Ley de Secreto Bancario (BSA) en 1970. Estas leyes de la BSA fueron un primer intento de detectar y prevenir el lavado de dinero y, desde entonces, han sido enmendadas y fortalecidas por otras leyes contra el lavado de dinero. La Red de Ejecución de Delitos Financieros es ahora la agencia responsable de administrar estas leyes. Su misión es proteger al sistema financiero de los abusos de los delitos financieros, incluido el financiamiento del terrorismo, el lavado de dinero y otras actividades ilegales.

En 1989, varios países y organizaciones fundaron el Grupo de Acción Financiera Internacional (GAFI) o en inglés Financial Action Task Force (FATF). Su tarea es desarrollar y hacer cumplir las pautas internacionales contra el lavado de dinero. Poco después del 11 de septiembre, el FATF amplió su mandato para incluir AML (Anti-Money Laundering) y contrarrestar el financiamiento del terrorismo. Otro organismo importante es el Fondo Monetario Internacional (FMI). Está compuesta por 189 países miembros y vela principalmente por la estabilidad del sistema monetario internacional. El FMI está preocupado por el impacto que el lavado de dinero y los delitos relacionados pueden tener en la integridad y estabilidad del sector financiero y la economía en general.

Desde los ataques terroristas de 2001, el GAFI ahora también incluye vigilancia terrorista en un esfuerzo por mitigar el financiamiento del terrorismo. Recientemente, la criptomoneda ha sido objeto de escrutinio, ya que brinda anonimato a sus usuarios. Esto ha facilitado un método de menor riesgo para que los delincuentes realicen sus transacciones.

### 1.3 Ciclo del blanqueo de capitales

Hay un total de tres etapas seguidas en el proceso de lavado de dinero que ayuda a los delincuentes a “limpiar” su dinero sucio.

**Colocación:** el dinero sucio o conseguido mediante acciones ilegales (venta de drogas, proxenetismo, trata de blancas, etc.) se integra en el sistema financiero y de esta manera legarlizarlo.

**Capas:** llevar a cabo transacciones financieras complejas para camuflar la fuente ilegal. Esta es la etapa en la que ocurre el fraude de transacciones cuando los lavadores realizan muchas transferencias electrónicas y crean un bucle complejo en el que se vuelve difícil identificar el origen de ese dinero.

**Integración:** ahora que el dinero está oculto desde su origen, los lavadores reingresarán el dinero a la economía para la compra de activos de lujo, inversiones financieras y comerciales.

### 1.4 Problema en las instituciones financieras

Las industrias bancarias y financieras buscan la manera de protegerse contra el fraude a todos los niveles y el blanqueo de capitales.

El lavado de dinero es una forma de delito económico. El objetivo es disfrazar los fondos obtenidos ilegalmente (dinero sucio) de tal manera que se dé la impresión de que provienen de una fuente de dinero acreditada.

Para ello los departamentos de compliance tiene muy en cuenta la AML (Anti Money laundering). AML se refiere a las actividades que las instituciones financieras realizan para cumplir con los requisitos legales para monitorear activamente y reportar actividades sospechosas.

La legislación AML se está volviendo cada vez más estricta para los proveedores de servicios financieros. Debe evitarse que financien el blanqueo de capitales y/o el terrorismo siguiendo las pautas de la OFAC (Office of Foreign Assests Control) y las recomendaciones del FATF (financial Action Task Force). Estos procedimientos se utilizan para identificar y bloquear las actividades fraudulentas en el sistema. Se implementa un algoritmo basado en reglas en el sistema para lanzar una alerta cada vez que se identifican transacciones fraudulentas. Estas alertas luego son investigadas por el analista y tomarán decisiones sobre si esta transacción es un fraude (Verdadero positivo) o no fraude (Falso positivo).

En el escenario actual, hay muchas alertas de falsos positivos dadas por el algoritmo y los investigadores cierran esas alertas dentro de un tiempo estipulado. Dado que hay una gran cantidad de alertas de falsos positivos, se invierte más tiempo en eliminar las alertas y las organizaciones requieren una gran cantidad de recursos humanos para investigar estas alertas. Por lo tanto, se desperdicia mucho tiempo y dinero solo porque el algoritmo basado en reglas no es lo suficientemente inteligente para identificar las transacciones no fraudulentas.

### 1.5 Solución al problema en al industria AML

Supongamos que un algoritmo basado en reglas podría detectar una transacción de alto valor entre marido y mujer. Aquí, el algoritmo trata a las partes como cliente A y cliente B y no comprende el motivo subyacente de la transacción y la relación entre ellos. Además, el algoritmo basado en reglas es fijo y no aprende ni se adapta a las tendencias. Los lavadores de dinero están tratando de evitar que sus transacciones sean detectadas por este algoritmo y mejoran su estrategia o maneras de actuar para evitar ser atrapados.

Por lo tanto, es necesario que el sistema también avance para aprender los patrones fraudulentos en las transacciones y activar solo las transacciones fraudulentas reales y evitar los falsos positivos. Esto se puede lograr entrenando un modelo de aprendizaje automático con los datos anteriores que contienen detalles de transacciones, ya sea que el algoritmo lo marque como fraude y el resultado del investigador en esas alertas marcadas (en realidad fraude o no).

### 1.6 AML vs Sanciones

Las sanciones y el cumplimiento AML son distintos, a menudo combinados, y frecuentemente convergente. Los requisitos AML varían según el país. A menudo surgen malentendidos acerca de las expectativas regulatorias para sanciones y cumplimiento AML, por lo que los profesionales de cumplimiento deben educar con frecuencia a las partes interesadas internas sobre estas diferentes expectativas y requisitos.

En los Estados Unidos, los requisitos AML se aplican a las instituciones financieras cubiertas por el Bank Secrecy Act (BSA). Como tal, el universo de entidades sujetas a varias formas de requisitos AML bajo la BSA son órdenes de magnitud menor que el universo de personas estadounidenses que están obligadas a cumplir con las sanciones.

Los requisitos AML se centran principalmente en la recopilación de información, retención de registros e intercambio de información con elTreasury Department’s Financial Crimes Enforcement Network (FinCEN) y otros organismos gubernamentales.

A diferencia de AML, las sanciones económicas se aplican a todas las personas estadounidenses y a algunas personas no estadounidenses. Las personas estadounidenses incluyen ciudadanos estadounidenses y residentes permanentes independientemente de dónde se encuentren, todas las entidades incorporadas en los EE. UU. y sus sucursales en el extranjero. Las personas no estadounidenses incluyen extranjeros subsidiarias de propiedad o controladas por empresas estadounidenses. Ciertos programas también requieren que las personas extranjeras en posesión de bienes de origen estadounidense cumplan con las normas establecidas.

### 1.7 Multas y sanciones a insituciones financieras en el siglo XXI

Para bien o para mal, gran parte del mundo mide el éxito o la falta de él en la lucha contra los delitos financieros a través de las multas o sanciones cada vez mayores impuestas a los bancos más grandes del mundo. Si bien las multas y las sanciones continuarán y, sin duda, proliferarán aún más, si han sido efectivas y si es hora de pensar más profundamente y considerar nuevos enfoques es digno de un estudio y discusión más profundos. En general, cualquier multa para una institución financiera no solo supone un duro golpe economico, sino que además esta en juego la reputación que se verá duramente afectada medianta la perdida de confianza del los clientes.

En el siglo XXI, se impusieron 96 multas en 17 países, a 57 bancos por 38,47 mil millones de dolates, siendo 21,47 mil millones para AML y 16,9 mil millones de dolares para Sanciones: un promedio anual de 1,77 billones de dolares (977 millones para AML y 800 millones para Sanciones). Las multas se pueden estructurar por areas, de esa manera podemos agruparlas en las siguientes areas dependiendo del delito o irregularidad que se haya cometido: ABC (Anti Bribery and Corruption) AML Non CBR (Non Correspondent Banking Relationship), AML CBR (Correspondent Banking Relationship) AML Fraud AML Markets. (Se excluye multas relacionadas a impuestos o abuso de mercados).

* **Correspondent Banking** se refiere a una institución financiera que brinda servicios a otra, generalmente en otro país (Correspondent Bank o Bancos corresponales). Actúa como intermediario o agente, facilitando las transferencias electrónicas, realizando transacciones comerciales, aceptando depósitos y recopilando documentos en nombre de otro banco. Los bancos corresponsales son utilizados por bancos nacionales para atender transacciones que se originan o se completan en países extranjeros. Los bancos nacionales suelen utilizar bancos corresponsales para obtener acceso a los mercados financieros extranjeros y atender a clientes internacionales sin tener que abrir sucursales en el extranjero.

#### 1.7.1 Las sanciones más grandes

Las multas AML impuestas más altas son:

* 1. Goldman Sachs 7.200 millones de dolares - Caso de esquema de soborno de Malasia
  2. JP Morgan 2,600 millones de dolares – Caso Madoff
  3. HSBC 1,600 millones de dolares – México/EE.UU. ML

Las multas más frecuentes incluyen:

* Deutsche Bank (7 multas)
* Goldman Sachs (5 multas)
* Credit Suisse (4 multas)
* J.P. Morgan (3 multas)
* HSBC (3 multas)

### 1.8 Librerias utilizadas en el proyecto

#Libraries  
suppressPackageStartupMessages(library(tidyverse))  
suppressPackageStartupMessages(library(gridExtra))  
suppressPackageStartupMessages(library(caret))  
suppressPackageStartupMessages(library(dplyr))  
suppressPackageStartupMessages(library(ggplot2))  
suppressPackageStartupMessages(library(R.utils))  
suppressPackageStartupMessages(library(GGally))

## 2 Dataset

Hay una falta de datos disponibles públicamente sobre servicios financieros y especialmente en el dominio emergente de transacciones de dinero. Parte del problema es la naturaleza intrínsecamente privada de las transacciones financieras, que conduce a que no haya conjuntos de datos disponibles públicamente.

Debido a la privacidad de los datos, es dificil encontrar datos de transacciones de caracter público. Por lo tanto, para llevar a cabo este proyecto usaremos datos simulados por Paysim que simula transacciones de dinero basadas en una muestra de transacciones reales extraídas de un mes de registros financieros de un servicio de dinero móvil implementado en un país africano.

Los registros originales fueron proporcionados por una empresa multinacional, que es el proveedor del servicio financiero móvil que actualmente se ejecuta en más de 14 países de todo el mundo.

Utiliza datos agregados del conjunto de datos privados para generar un conjunto de datos sintético que se asemeja al funcionamiento normal de las transacciones e inyecta comportamiento malicioso para evaluar posteriormente el rendimiento de los métodos de detección de fraude.

### 2.1 Descripción del dataset

En este apartado procederemos a explicar las columnas que tenemos en nuestro dataset. Hay un total de 11 columnas que hacen referencias a las 11 variables que tenemos. En este dataset tenemos tanto variables cualitativas cómo variables cuantitativas.

-**step:** mapea una unidad de tiempo en el mundo real. En este caso 1 paso es 1 hora de tiempo.

-**type:** CASH-IN, CASH-OUT, DEBIT, PAYMENT y TRANSFER.

-**amount:** cantidad de la transacción en moneda local.

-**nameOrig:** cliente que inició la transacción

-**oldbalanceOrg:** saldo inicial antes de la transacción

-**newbalanceOrig:** nuevo saldo después de la transacción

-**nameDest:** cliente que es el destinatario de la transacción

-**oldbalanceDest:** saldo inicial del destinatario antes de la transacción. Tenemos que tener cuenta que no hay información para clientes que comiencen con M (Merchants).

-**newbalanceDest:** saldo del destinatario después de la transacción. Tenemos que tener cuenta que no hay información para clientes que comiencen con M (Merchants).

-**isFraud:** identifica un transacción fraudulenta (1) y una transacción no fraudulenta (0).

-**isFlaggedFraud:** identifica intentos ilegales de transferir más de 200.000 en una sola transacción.

## 3 Preprocesamiento de los datos

A continuación vamos cargar el fichero con el cual trabajaremos a lo largo de este proyecto. Para ellos utilizaremos la función read.csv y le pasaremos el nombre del archivo csv y el caracter de separación.

# Cargamos el fichero con el cual trabajaremos a lo largo de este proyecto  
transacciones <- read.csv('bankTransaction\_data.csv', sep=",")

Una vez hemos cargado el dataset y lo hemos guardado en una variable, en este caso nuestra variable es transacciones, utilizaremos la función str que devolverá la estructura de nuestro dataset mostrando las observacions o filas y el número de columnas que nos indicará las variables del dataset.

str(transacciones)

## 'data.frame': 6362620 obs. of 11 variables:  
## $ step : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ type : chr "PAYMENT" "PAYMENT" "TRANSFER" "CASH\_OUT" ...  
## $ amount : num 9840 1864 181 181 11668 ...  
## $ nameOrig : chr "C1231006815" "C1666544295" "C1305486145" "C840083671" ...  
## $ oldbalanceOrg : num 170136 21249 181 181 41554 ...  
## $ newbalanceOrig: num 160296 19385 0 0 29886 ...  
## $ nameDest : chr "M1979787155" "M2044282225" "C553264065" "C38997010" ...  
## $ oldbalanceDest: num 0 0 0 21182 0 ...  
## $ newbalanceDest: num 0 0 0 0 0 ...  
## $ isFraud : int 0 0 1 1 0 0 0 0 0 0 ...  
## $ isFlaggedFraud: int 0 0 0 0 0 0 0 0 0 0 ...

En este caso tenemos un conjunto de datos con 6 millones de observaciones y 11 variables. Además podemos ver el tipo de variable que tenemos. En este caso tenemos 8 variables numericas y 3 variables de tipo character.

Otra función de R que nos puede ayudar a tener un overview de datos estadísticos del conjunto de datos es summary. A continuación mostramos la estadística descriptiva de nuestro conjunto de datos.

summary(transacciones)

## step type amount nameOrig   
## Min. : 1.0 Length:6362620 Min. : 0 Length:6362620   
## 1st Qu.:156.0 Class :character 1st Qu.: 13390 Class :character   
## Median :239.0 Mode :character Median : 74872 Mode :character   
## Mean :243.4 Mean : 179862   
## 3rd Qu.:335.0 3rd Qu.: 208721   
## Max. :743.0 Max. :92445517   
## oldbalanceOrg newbalanceOrig nameDest oldbalanceDest   
## Min. : 0 Min. : 0 Length:6362620 Min. : 0   
## 1st Qu.: 0 1st Qu.: 0 Class :character 1st Qu.: 0   
## Median : 14208 Median : 0 Mode :character Median : 132706   
## Mean : 833883 Mean : 855114 Mean : 1100702   
## 3rd Qu.: 107315 3rd Qu.: 144258 3rd Qu.: 943037   
## Max. :59585040 Max. :49585040 Max. :356015889   
## newbalanceDest isFraud isFlaggedFraud   
## Min. : 0 Min. :0.000000 Min. :0.0e+00   
## 1st Qu.: 0 1st Qu.:0.000000 1st Qu.:0.0e+00   
## Median : 214661 Median :0.000000 Median :0.0e+00   
## Mean : 1224996 Mean :0.001291 Mean :2.5e-06   
## 3rd Qu.: 1111909 3rd Qu.:0.000000 3rd Qu.:0.0e+00   
## Max. :356179279 Max. :1.000000 Max. :1.0e+00

### 3.1 Limpieza de datos

Dado que los datos son muy grandes, limpiar el conjunto de datos de antemano es un requisito previo para la eficiencia, ya que la computadora tendrá que manejar una gran cantidad de datos. El siguiente paso tratará la limpieza de datos, mirando si hay valores vacíos o nulos en el conjunto de datos.

colSums(is.na(transacciones))

## step type amount nameOrig oldbalanceOrg   
## 0 0 0 0 0   
## newbalanceOrig nameDest oldbalanceDest newbalanceDest isFraud   
## 0 0 0 0 0   
## isFlaggedFraud   
## 0

colSums(transacciones=="")

## step type amount nameOrig oldbalanceOrg   
## 0 0 0 0 0   
## newbalanceOrig nameDest oldbalanceDest newbalanceDest isFraud   
## 0 0 0 0 0   
## isFlaggedFraud   
## 0

Podemos ver que no tenemos registros vacíos o nulos.

Además podemos observar que, además de las 6 características numéricas, tenemos 3 columnas de enteros y 3 de caracteres. Nos centraremos en estos primeros.

De esta primera impresión:

* isFlaggedFraud se parece a la predicción del algoritmo existente. No queremos que nuestro modelo dependa de la predicción de terceros, por lo que nos desharemos de él.
* isFraud, la variable objetivo, se convertirá en un factor y se le cambiará el nombre para facilitar la trazabilidad (los niveles de los factores se convertirán en etiquetas significativas).
* nameDest y nameOrig parecen registros aleatorios y sin poder predictivo, pero los exploraremos a fondo en caso de que podamos extraer algún valor de ellos.
* type es una característica potencialmente poderosa. Podemos suponer fácilmente que el fraude es propenso a algunos tipos de transacciones. Por ejemplo, es bastante improbable que el ingreso en efectivo y el pago (dar dinero) coexistan con un esquema de fraude. Esto también se explorará en detalle.
* step es solo la variable de tiempo. Probablemente no será de utilidad en nuestra predicción, ya que el fraude puede ocurrir en cualquier momento. Aún así, lo trataremos como predictivo hasta que se demuestre lo contrario.

Para empezar, echaremos un vistazo a la cantidad de valores únicos que contienen las columnas de caracteres. Esto nos ayudará a dar forma a nuestra estrategia de limpieza.

print("Numero de valor distintos en campos de tipos char")

## [1] "Numero de valor distintos en campos de tipos char"

transacciones[ , sapply(transacciones,is.character)] %>% sapply(n\_distinct) %>% print()

## type nameOrig nameDest   
## 5 6353307 2722362

**nameOrig y nameDest** (nombre de origen y nombre de destino respectivamente) contienen demasiados valores distintos para ser tratados como factores. Los números son obviamente aleatorios, probablemente generados por una función hash. Sin embargo, el prefijo puede simbolizar alguna información implícita (por ejemplo, el tipo de cuenta, ya sea una cuenta de ahorros o corriente, etc.). Para probar eso, veremos la cantidad de prefijos distintos y su frecuencia.

print("Distintos prefijos en las columnas nameOrig y nameDest ")

## [1] "Distintos prefijos en las columnas nameOrig y nameDest "

transacciones %>% mutate(origin\_name\_prefix = str\_sub(nameOrig,1,1), #creamos columnas con los prefijos  
 dest\_name\_prefix = str\_sub(nameDest, 1, 1)) %>%  
 select( c( origin\_name\_prefix,dest\_name\_prefix ) ) %>%   
 table() %>% #count de la distinas ocurrences  
 print()

## dest\_name\_prefix  
## origin\_name\_prefix C M  
## C 4211125 2151495

Parece que solo hay dos prefijos, C y M, casi igualmente presentes en la columna dest\_name\_prefix. Sin embargo, origin\_name\_prefix contiene solo C, por lo que podría no tener poder predictivo.

Como resultado, podemos:

* añadir la columna de prefijo de destino,
* omitir la creación de la entidad respectiva a partir del prefijo de origen
* deshacernos de las columnas iniciales
* deshacernos de la columna isFlaggedFraud

Ademas, convertiremos las columnas restantes de enteros y caracteres en factores.

Finalmente, para que nuestros gráficos sean más legibles, cambiaremos el nombre de las columnas de balance y el de la variable objetivo.

transacciones <- transacciones %>% mutate(dest\_name\_prefix = str\_sub(nameDest, 1, 1)) %>% #creamos columnas creando prefijos  
 select(-c('nameOrig','nameDest','isFlaggedFraud')) %>% #eliminamos columnas deseadas  
 mutate\_if(is.integer, as.factor) %>% #convertimos variables integers en factors  
 mutate\_if(is.character,as.factor) %>% #convertimos variables chars en factores  
 rename("Old\_Balance\_of\_Origin" = "oldbalanceOrg" , #renombramos a formato legible  
 "New\_Balance\_of\_Origin" = "newbalanceOrig",  
 "Old\_Balance\_of\_Destination" = "oldbalanceDest",  
 "New\_Balance\_of\_Destination" = "newbalanceDest")  
  
levels(transacciones$isFraud) <- c("No\_Fraud", "Fraud" ) #formato legible

head(transacciones)

## step type amount Old\_Balance\_of\_Origin New\_Balance\_of\_Origin  
## 1 1 PAYMENT 9839.64 170136 160296.36  
## 2 1 PAYMENT 1864.28 21249 19384.72  
## 3 1 TRANSFER 181.00 181 0.00  
## 4 1 CASH\_OUT 181.00 181 0.00  
## 5 1 PAYMENT 11668.14 41554 29885.86  
## 6 1 PAYMENT 7817.71 53860 46042.29  
## Old\_Balance\_of\_Destination New\_Balance\_of\_Destination isFraud  
## 1 0 0 No\_Fraud  
## 2 0 0 No\_Fraud  
## 3 0 0 Fraud  
## 4 21182 0 Fraud  
## 5 0 0 No\_Fraud  
## 6 0 0 No\_Fraud  
## dest\_name\_prefix  
## 1 M  
## 2 M  
## 3 C  
## 4 C  
## 5 M  
## 6 M

## 4 Analisis gráfico

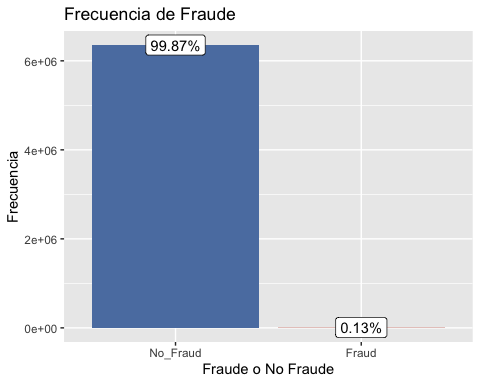
En este apartado mostraremos gráficas que nos dará información sobre el conjunto de datos que tenemos.

### 4.1 Variables cuantitativas

#### 4.1.1 Frecuencia de la variable objetivo

A continuación veremos la frecuencia y el porcentaje de cada tipo de variable objetivo (Fraude vs No Fraude).

transacciones %>% ggplot(aes(x = isFraud, y = (..count..))) +   
 geom\_bar(fill = c( "#5b7fb0", "#c2695b" ) , stat = "count")+  
 geom\_label(stat='count',aes( label= paste0( round( ((..count..)/sum(..count..)) ,4)\*100 , "%" ) ) )+ #añadimos el porcentaje a cada etiqueta  
 labs(x = "Fraude o No Fraude", y = "Frecuencia", title = "Frecuencia de Fraude")



Comp podemos ver tenemos un conjunto de datos no balanceado ya que el porcentaje de observacions de fraude vs el de no fraude es muy grande. Este gran desequilibrio entre las dos clases objetivo conducirá a nuestro modelo a un sesgo hacia la variable dominante.

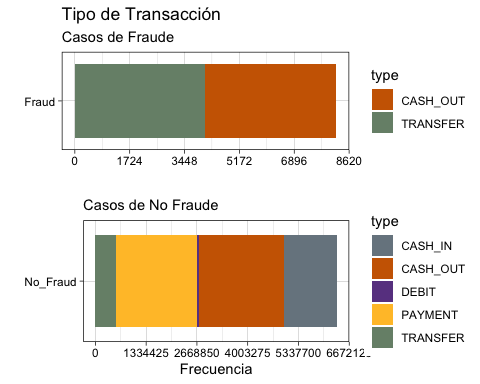
Esto significa que dado que más del 99% de los casos no son fraudes, nuestros algoritmos se sentirán cómodos con la predicción de “No Fraude”.

Más adelante, submuestrearemos la clase mayoritaria para crear un conjunto de datos más balanceado.

#### 4.1.2 Tipo de transacción

A continuación, exploraremos la presencia de cada tipo de transacción en los casos de fraude y no fraude.

#Creamos la paleta de colores   
paleta<- c(CASH\_IN = "#78858f", CASH\_OUT = "#CC6600", DEBIT ="#6A4491", PAYMENT = "#FFC233", TRANSFER = "#788f78")  
  
#recuento de tipos de transacciones   
p1 <- transacciones %>% filter(isFraud == "Fraud") %>%   
 ggplot( aes( x = isFraud, fill = type ) ) +  
 geom\_bar()+  
 scale\_fill\_manual(values=paleta) +  
 scale\_y\_continuous(breaks = function(x){ seq(0,max(x), floor( (max(x)/5) ) )} )+  
 labs(title = "Tipo de Transacción", subtitle = "Casos de Fraude", x = "", y = "" ) + coord\_flip() + theme\_linedraw()  
  
p2 <-transacciones %>% filter(isFraud == "No\_Fraud") %>%   
 ggplot( aes( x = isFraud, fill = type ) ) +  
 geom\_bar()+  
 scale\_fill\_manual(values=paleta) +  
 scale\_y\_continuous(breaks = function(x){ seq(0,max(x), floor( (max(x)/5) ) )} )+  
 labs(subtitle = "Casos de No Fraude", x = "", y = "Frecuencia" ) + coord\_flip() + theme\_linedraw()  
  
grid.arrange(p1,p2,nrow = 2)

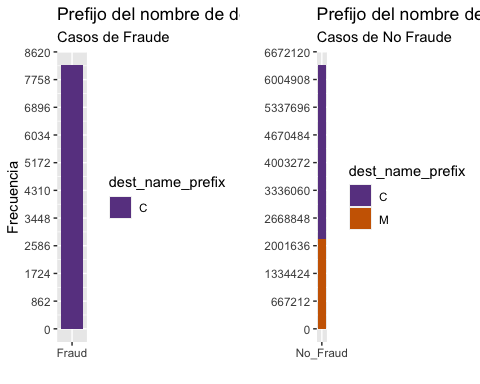


Como podemos ver, los registros de fraude coinciden con las transaccions de tipo Transfer y Cash Out. Eso significa que el resto de los tipos de transacciones (Cash In, Debit, Payment) nunca están asociados con el fraude. Por lo tanto vamos a deshacernos de todas las observaciones con estas transacciones, ya que solo conducen a “No Fraude”.

#### 4.1.3 Nombre de destinatario

Antes de eliminar estos tipos de transacciones, abordaremos de manera similar la columna dest\_name\_prefix para tener en cuenta efectos similares.

#creamos la paleta de colores asociado a cada tipo de destinatario  
paleta\_dest<- c("C" = "#6A4491", "M" = "#CC6600")  
  
#recuentoswa de prefijo en casos de fraude  
  
p3 <- transacciones %>% filter(isFraud == "Fraud") %>%   
 ggplot( aes( x = isFraud, fill = dest\_name\_prefix ) ) +  
 geom\_bar()+  
 scale\_fill\_manual(values=paleta\_dest) +  
 scale\_y\_continuous(breaks = function(x){ seq(0,max(x), floor( (max(x)/10) ) )} )+  
 labs(title = "Prefijo del nombre de destino",subtitle = "Casos de Fraude", x = "", y = "Frecuencia" )  
  
p4 <-transacciones %>% filter(isFraud == "No\_Fraud") %>%   
 ggplot( aes( x = isFraud, fill = dest\_name\_prefix ) ) +  
 geom\_bar()+  
 scale\_fill\_manual(values=paleta\_dest) +  
 scale\_y\_continuous(breaks = function(x){ seq(0,max(x), floor( (max(x)/10) ) )} )+  
 labs(title = "Prefijo del nombre de destino", subtitle = "Casos de No Fraude", x = "", y = "" )  
  
grid.arrange(p3,p4,ncol = 2)

 A partir de la gráfica anterior, el fraude coincide únicamente con el prefijo “C” por lo tanto podríamos eliminar las observaciones que contengan la “M”. Después de la eliminación, esta columna contiene solo “C” (es decir, varianza cero), lo que significa que podemos eliminarla.

Este proceso nos resultó útil no como una nueva columna que puede predecir el resultado, sino como una evidencia que nos llevó a descartar una cantidad significativa de filas que no nos aportan mucho. A continuación procederemos a eliminar los registros los cuales hemos analisado y razonado el porqué de la eliminación anteriormente.

rows\_before <- dim(transacciones)[1]  
  
transacciones <- transacciones %>% filter( (!(type %in% c("CASH\_IN", "DEBIT", "PAYMENT") )) , #eliminamos estos tipos transacciones  
 dest\_name\_prefix == "C") %>% #eliminamos los destinatarios con prefijo M  
 select(-dest\_name\_prefix) #eliminamos la columna dest\_name\_prefix

A continuación calcularemos el número de registros que se han eliminado y el porcentaje equivalente en el dataset.

rows\_after <- dim(transacciones)[1]  
diff <- rows\_before - rows\_after  
diff\_perc <- round(diff/rows\_before \* 100, 2)  
sprintf("%i rows removed, equivalent to %s%% of the data",diff, diff\_perc)

## [1] "3592211 rows removed, equivalent to 56.46% of the data"

Más de la mitad de los registros iniciales se han eliminado.

head(transacciones)

## step type amount Old\_Balance\_of\_Origin New\_Balance\_of\_Origin  
## 1 1 TRANSFER 181.0 181.00 0  
## 2 1 CASH\_OUT 181.0 181.00 0  
## 3 1 CASH\_OUT 229133.9 15325.00 0  
## 4 1 TRANSFER 215310.3 705.00 0  
## 5 1 TRANSFER 311685.9 10835.00 0  
## 6 1 CASH\_OUT 110414.7 26845.41 0  
## Old\_Balance\_of\_Destination New\_Balance\_of\_Destination isFraud  
## 1 0 0.00 Fraud  
## 2 21182 0.00 Fraud  
## 3 5083 51513.44 No\_Fraud  
## 4 22425 0.00 No\_Fraud  
## 5 6267 2719172.89 No\_Fraud  
## 6 288800 2415.16 No\_Fraud

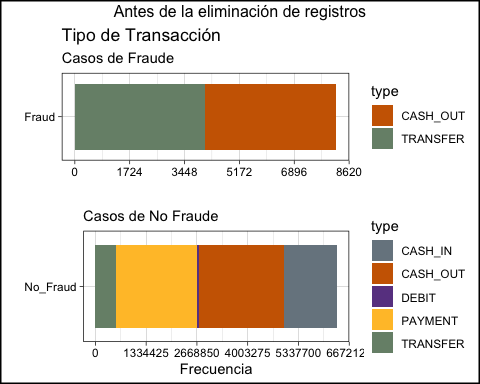
dim(transacciones)[1]

## [1] 2770409

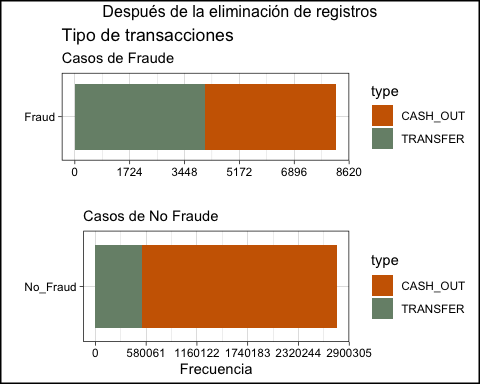
Ahora vamos a visualizar el tipo de transacciones antes y despues de la eliminación de registros.

p11 <- transacciones %>% filter(isFraud == "Fraud") %>%   
 ggplot( aes( x = isFraud, fill = type ) ) +  
 geom\_bar()+  
 scale\_fill\_manual(values=paleta) +  
 scale\_y\_continuous(breaks = function(x){ seq(0,max(x), floor( (max(x)/5) ) )}) +  
 labs(title = "Tipo de transacciones", subtitle = "Casos de Fraude", x = "", y = "" ) + coord\_flip() + theme\_linedraw()  
  
p12 <-transacciones %>% filter(isFraud == "No\_Fraud") %>%   
 ggplot( aes( x = isFraud, fill = type ) ) +  
 geom\_bar()+  
 scale\_fill\_manual(values=paleta) +  
 scale\_y\_continuous(breaks = function(x){ seq(0,max(x), floor( (max(x)/5) ) )} )+  
 labs(subtitle = "Casos de No Fraude", x = "", y = "Frecuencia" ) + coord\_flip() + theme\_linedraw()

library(grid)  
grid.arrange(p1,p2,top = "Antes de la eliminación de registros")  
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))



grid.arrange(p11,p12,top = "Después de la eliminación de registros")  
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))



Al parecer los casos de no fraude contienen más transacciones de retiros en efectivo, mientras que los casos de fraude las transferencias y los retiros en efectivos esta presentes casi por igual. Esto podría ser un indicio del poder predictivo de esta caracteristica especifica.

### 4.2 Variables cuantitativas

#### 4.2.1 Amount

Dado que los valores numéricos van desde cantidades insignificantes hasta cantidades enormes, las graficaremos después de aplicarles la función log10. De esta manera, la distancia entre cantidades bajas y altas se minimiza y se vuelve más fácil para el ojo humano detectar patrones.

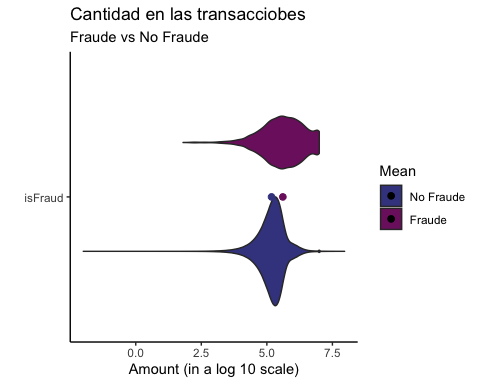
#Cantidad maxima en la variable Amount  
max(transacciones$amount)

## [1] 92445517

#Cantidad minima en la variable Amount  
min(transacciones$amount)

## [1] 0

transacciones %>%   
 ggplot(aes(x = "isFraud", y = log(amount,10) , fill = isFraud)) +   
 geom\_violin() + coord\_flip() +   
 scale\_fill\_manual(values=c("#40448f", "#7d1f6d") , name="Mean", labels=c( "No Fraude","Fraude"))+  
 labs(x = "", y = "Amount (in a log 10 scale)", title = "Cantidad en las transacciobes", subtitle = "Fraude vs No Fraude" ) +  
 stat\_summary(fun= "mean",geom = "point", size = 2, col =c("#40448f", "#7d1f6d"))+  
 theme\_classic()

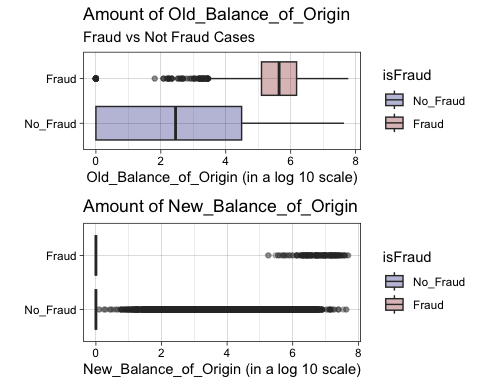
 Podemos ver que, el monto de la transacción se ve diferente en los casos de fraude frente a no fraude. Las cantidades de No\_fraud oscilan alrededor de 100000 (10^5), mientras que los casos de fraude se distribuyen de manera más uniforme. Esto podría ser una pista de que la columna Amount puede hacer una contribución en nuestro modelo.

#### 4.2.2 Balance

En contraste con la columna Amount, las columnas balance contienen valores con ceros. Los ceros devuelven un -Infinito cuando se les aplica la función logaritmo.

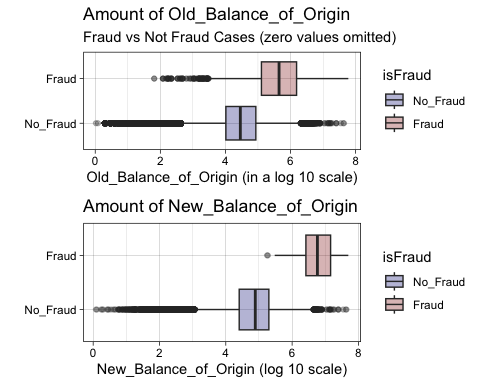
Una solución conveniente a este problema es agregar 1 en nuestro vector numérico antes de pasarlo a la función logarítmica. De esta manera, los 0 se convierten en 1 y producen un 0 después de aplicar la función. Esto sucede porque cualquier número elevado a la potencia de 0 devuelve 1.

which(transacciones %>% sapply(is.numeric))[-1] %>% names() -> labs # store names for use in plot labels  
  
p5 <- transacciones %>%   
 ggplot(aes(x = isFraud, y = log(Old\_Balance\_of\_Origin+1,10) , fill = isFraud)) +   
 geom\_boxplot(alpha = .3) + coord\_flip() +   
 scale\_fill\_manual(values=c("navy", "darkred"))+  
 labs(x = "", y = paste(labs[1], "(in a log 10 scale)"), title = paste("Amount of", labs[1] ), subtitle = "Fraud vs Not Fraud Cases" ) +  
 theme\_linedraw()  
  
  
p6 <- transacciones %>%  
 ggplot(aes(x = isFraud, y = log(New\_Balance\_of\_Origin+1,10) , fill = isFraud)) +   
 geom\_boxplot(alpha = .3) + coord\_flip() +   
 scale\_fill\_manual(values=c("navy", "darkred"))+  
 labs(x = "", y = paste(labs[2], "(in a log 10 scale)"), title = paste("Amount of", labs[2] ) ) +  
 theme\_linedraw()  
  
grid.arrange(p5,p6)



Observamos una concentración muy alta de valores 0 que hacen que nuestros gráficos no sean informativos. A continuación trataremos de omitirlos.

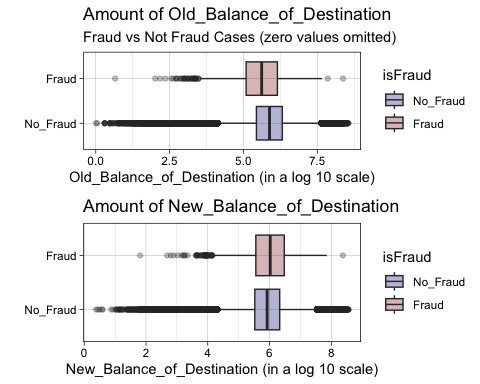
p7 <- transacciones %>% filter(Old\_Balance\_of\_Origin != 0) %>%  
 ggplot(aes(x = isFraud, y = log(Old\_Balance\_of\_Origin+1,10) , fill = isFraud)) +   
 geom\_boxplot(alpha = .3) + coord\_flip() +   
 scale\_fill\_manual(values=c("navy", "darkred"))+  
 labs(x = "", y = paste(labs[1], "(in a log 10 scale)"), title = paste("Amount of", labs[1] ), subtitle = "Fraud vs Not Fraud Cases (zero values omitted)" )+  
 theme\_linedraw()  
  
  
p8 <- transacciones %>% filter(New\_Balance\_of\_Origin != 0) %>%  
 ggplot(aes(x = isFraud, y = log(New\_Balance\_of\_Origin+1,10) , fill = isFraud)) +   
 geom\_boxplot(alpha = .3) + coord\_flip() +   
 scale\_fill\_manual(values=c("navy", "darkred"))+  
 labs(x = "", y = paste(labs[2], "(log 10 scale)"), title = paste("Amount of", labs[2] ) )+  
 theme\_linedraw()  
  
grid.arrange(p7,p8)



Tanto en Old como en New Balance of Origin, los fraudes suelen ocurrir en mayor cantidad de balance. Esto también podría ser un signo de poder predictivo.

Aplicaremos el mismo procedimiento a Balance of destination.

p9 <- transacciones %>% filter(Old\_Balance\_of\_Destination != 0) %>%  
 ggplot(aes(x = isFraud, y = log(Old\_Balance\_of\_Destination+1,10) , fill = isFraud)) +   
 geom\_boxplot(alpha = .3) + coord\_flip() +   
 scale\_fill\_manual(values=c("navy", "darkred"))+  
 labs(x = "", y = paste(labs[3], "(in a log 10 scale)"), title = paste("Amount of", labs[3] ), subtitle = "Fraud vs Not Fraud Cases (zero values omitted)" )+  
 theme\_linedraw()  
  
p10 <- transacciones %>% filter(New\_Balance\_of\_Destination != 0) %>%  
 ggplot(aes(x = isFraud, y = log(New\_Balance\_of\_Destination+1,10) , fill = isFraud)) +   
 geom\_boxplot(alpha = .3) + coord\_flip() +   
 scale\_fill\_manual(values=c("navy", "darkred"))+  
 labs(x = "", y = paste(labs[4], "(in a log 10 scale)"), title = paste("Amount of", labs[4] ) )+  
 theme\_linedraw()  
  
grid.arrange(p9,p10)



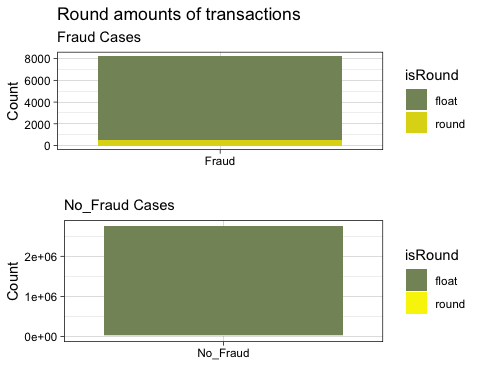
A diferencia de los saldos de origen, los saldos de destino no difieren mucho entre los casos de fraude y no fraude.

### 4.3 El efecto “entero”

Los estafadores pueden pensar que transferir una cantidad no redonda (por ejemplo, 12 325 en lugar de 10 000) parecerá una transacción más común y no despertará sospechas. Además, a la mente humana siempre se le ocurren números redondeados y ese sería el primer pensamiento de todos los estafadores (no tan profesionales). Por eso comprobaremos si los importes redondos de las transacciones coinciden con el fraude.

#Creamos una nueva columan isRound  
transacciones <- mutate(transacciones, isRound = as.factor(ifelse( test = (transacciones$amount - as.integer(transacciones$amount)) == 0 , yes = 'round', no = 'float' )))

p11 <- transacciones %>% filter( isFraud == "Fraud" ) %>% ggplot() +   
 geom\_bar(aes( x = isFraud , fill = isRound ) ) +  
 scale\_fill\_manual(values = c( "#849468", "#ded718" )) +   
 labs( title = "Round amounts of transactions" , x= "", y = "Count", subtitle = "Fraud Cases" )+  
 theme\_linedraw()  
  
p12 <- transacciones %>% filter( isFraud == "No\_Fraud" ) %>% ggplot() +   
 geom\_bar(aes( x = isFraud , fill = isRound ) ) +  
 scale\_fill\_manual(values = c( "#849468", "#f7f300" ))+  
 labs( x= "", y = "Count", subtitle = "No\_Fraud Cases" ) +  
 theme\_linedraw()  
  
grid.arrange(p11,p12)



Curiosamente, los números enteros parecen ser más frecuentes en los casos de fraude. Esa es probablemente una pista de una variable interesante para posteriores calculos.

## 4.4 ¿Es el saldo realmente cero?

Otra suposición que se podría hacer es que los estafadores usan cuentas ficticias para enviar o (principalmente) recibir dinero. Es por esto que sería interesante explorar si las cuentas con saldos 0 están conectadas con el fraude. Para eso, crearemos un vector lógico para cada columna de Saldo que ya tenemos.

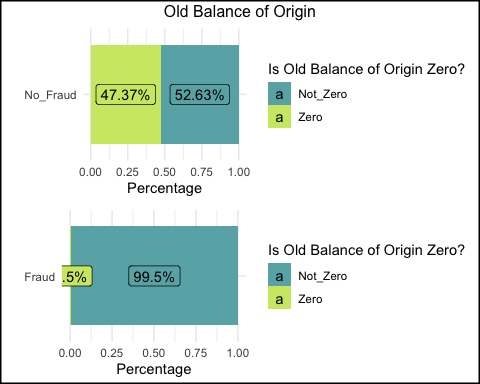
if(!require(R.utils)){  
 install.packages('R.utils', repos='http://cran.us.r-project.org')  
 library(R.utils)  
}

iszero <- transacciones %>% select(-amount) %>% #excluimos la cantidad que no puede ser 0   
 select\_if(is.numeric) %>% #select the remaining numerics - ie the balances  
 mutate\_all(R.utils::isZero) %>% #check whether they are zero  
 rename\_all(~paste0('isZero\_', substr(., 1, nchar(.) - 4))) #add an isZero prefix on the names  
  
transacciones <- transacciones %>% cbind(iszero) #merge with original df

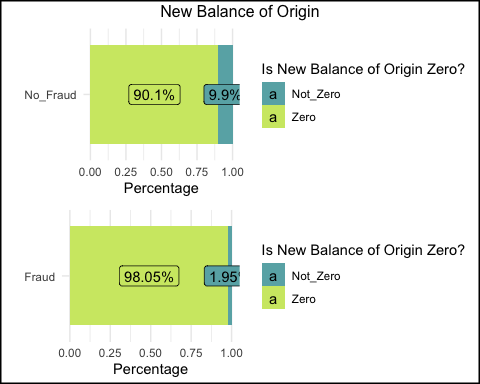
Primero, veremos las cuentas de **origen**.

p13 <- transacciones %>% filter(isFraud == "No\_Fraud") %>%  
 ggplot( aes( x = isFraud, fill = isZero\_Old\_Balance\_of\_Or ) ) +   
 geom\_bar(position = "fill") +   
 labs(x = "", y = "Percentage") +  
 geom\_label(stat="count",aes(label= paste0(round(((..count..)/sum(..count..)\*100),2),"%") ), position = position\_fill(vjust=0.5)) +  
 scale\_fill\_manual(values = c("#69b0b3", "#d0e872"), #colors  
 labels = c("Not\_Zero", "Zero"), #legend labels  
 name = "Is Old Balance of Origin Zero?") + #legend title  
 theme\_minimal()+   
 coord\_flip()  
  
p14 <- transacciones %>% filter(isFraud == "Fraud") %>%  
 ggplot( aes( x = isFraud, fill = isZero\_Old\_Balance\_of\_Or ) ) +   
 geom\_bar(position = "fill") +   
 labs(x = "", y = "Percentage") +  
 geom\_label(stat="count",aes(label= paste0(round(((..count..)/sum(..count..)\*100),2),"%") ), position = position\_fill(vjust=0.5)) +  
 scale\_fill\_manual(values = c("#69b0b3", "#d0e872"), #colors  
 labels = c("Not\_Zero", "Zero"), #legend labels  
 name = "Is Old Balance of Origin Zero?")+ #legend title  
 theme\_minimal()+  
 coord\_flip()  
  
p15 <- transacciones %>% filter(isFraud == "No\_Fraud") %>%  
 ggplot( aes( x = isFraud, fill = isZero\_New\_Balance\_of\_Or ) ) +   
 geom\_bar(position = "fill") +   
 labs(x = "", y = "Percentage") +  
 geom\_label(stat="count",aes(label= paste0(round(((..count..)/sum(..count..)\*100),2),"%") ), position = position\_fill(vjust=0.5)) +  
 scale\_fill\_manual(values = c("#69b0b3", "#d0e872"),   
 labels = c("Not\_Zero", "Zero"),  
 name = "Is New Balance of Origin Zero?") +  
 theme\_minimal() +  
 coord\_flip()  
  
p16 <- transacciones %>% filter(isFraud == "Fraud") %>%  
 ggplot( aes( x = isFraud, fill = isZero\_New\_Balance\_of\_Or ) ) +   
 geom\_bar(position = "fill") +   
 labs(x = "", y = "Percentage") +   
 geom\_label(stat="count",aes(label= paste0(round(((..count..)/sum(..count..)\*100),2),"%") ), position = position\_fill(vjust=0.5)) +  
 scale\_fill\_manual(values = c("#69b0b3", "#d0e872"),   
 labels = c("Not\_Zero", "Zero"),  
 name = "Is New Balance of Origin Zero?")+  
 theme\_minimal() + coord\_flip()

grid.arrange(p13,p14,nrow = 2, top = "Old Balance of Origin")  
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))



grid.arrange(p15,p16,nrow = 2, top = "New Balance of Origin")  
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))



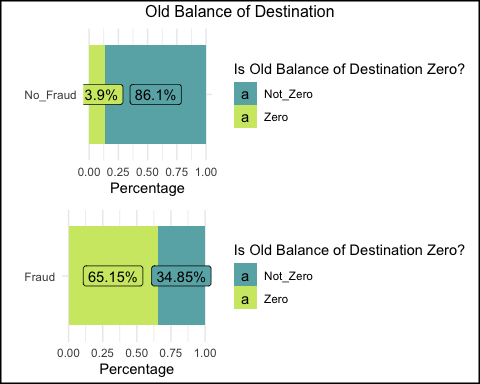
En el Old Balance of Origin, los casos de fraude muy rara vez contienen saldos cero. Eso es de esperar porque no tiene sentido que un estafador tenga un objetivo una cuenta con saldo cero.

En New Balance of Origin en cambio, el 98% de los casos de fraude lo dejan con saldo cero (es decir, los estafadores saquean las cuentas y las dejan a cero).

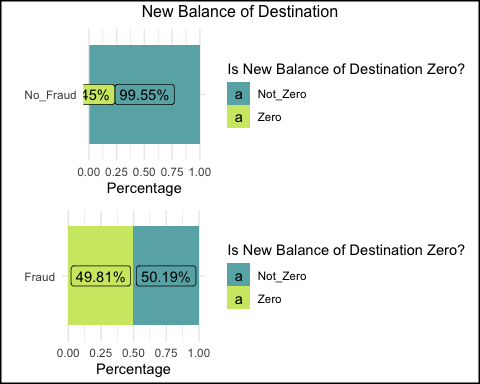
Ahora veremos que comportamiento obtenemos con las cuentas de **destino**, “Balance of Destination”.

p17 <- transacciones %>% filter(isFraud == "No\_Fraud") %>%  
 ggplot( aes( x = isFraud, fill = isZero\_Old\_Balance\_of\_Destina ) ) +   
 geom\_bar(position = "fill") +   
 labs(x = "", y = "Percentage") +  
 geom\_label(stat="count",aes(label= paste0(round(((..count..)/sum(..count..)\*100),2),"%") ), position = position\_fill(vjust=0.5)) +  
 scale\_fill\_manual(values = c("#69b0b3", "#d0e872"), #colors  
 labels = c("Not\_Zero", "Zero"), #legend labels  
 name = "Is Old Balance of Destination Zero?") + #legend title  
 theme\_minimal()+   
 coord\_flip()  
  
p18 <- transacciones %>% filter(isFraud == "Fraud") %>%  
 ggplot( aes( x = isFraud, fill = isZero\_Old\_Balance\_of\_Destina ) ) +   
 geom\_bar(position = "fill") +   
 labs(x = "", y = "Percentage") +  
 geom\_label(stat="count",aes(label= paste0(round(((..count..)/sum(..count..)\*100),2),"%") ), position = position\_fill(vjust=0.5)) +  
 scale\_fill\_manual(values = c("#69b0b3", "#d0e872"), #colors  
 labels = c("Not\_Zero", "Zero"), #legend labels  
 name = "Is Old Balance of Destination Zero?")+ #legend title  
 theme\_minimal()+  
 coord\_flip()  
  
p19 <- transacciones %>% filter(isFraud == "No\_Fraud") %>%  
 ggplot( aes( x = isFraud, fill = isZero\_New\_Balance\_of\_Destina ) ) +   
 geom\_bar(position = "fill") +   
 labs(x = "", y = "Percentage") +  
 geom\_label(stat="count",aes(label= paste0(round(((..count..)/sum(..count..)\*100),2),"%") ), position = position\_fill(vjust=0.5)) +  
 scale\_fill\_manual(values = c("#69b0b3", "#d0e872"),   
 labels = c("Not\_Zero", "Zero"),  
 name = "Is New Balance of Destination Zero?") +  
 theme\_minimal() +  
 coord\_flip()  
  
p20 <- transacciones %>% filter(isFraud == "Fraud") %>%  
 ggplot( aes( x = isFraud, fill = isZero\_New\_Balance\_of\_Destina ) ) +   
 geom\_bar(position = "fill") +   
 labs(x = "", y = "Percentage") +   
 geom\_label(stat="count",aes(label= paste0(round(((..count..)/sum(..count..)\*100),2),"%") ), position = position\_fill(vjust=0.5)) +  
 scale\_fill\_manual(values = c("#69b0b3", "#d0e872"),   
 labels = c("Not\_Zero", "Zero"),  
 name = "Is New Balance of Destination Zero?")+  
 theme\_minimal() + coord\_flip()

grid.arrange(p17,p18,nrow = 2, top = "Old Balance of Destination")   
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))



grid.arrange(p19,p20,nrow = 2, top = "New Balance of Destination")   
grid.rect(gp = gpar(lwd = 3, col = "black", fill = NA))



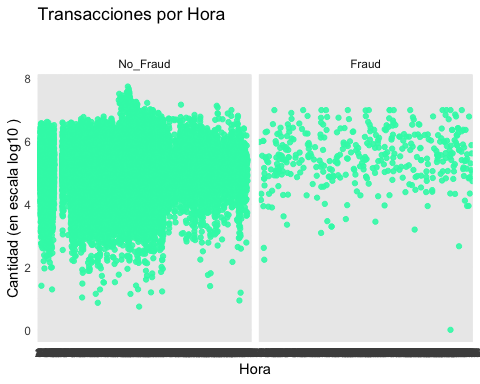
Tanto en Old como en New Balance of Destination, los casos de fraude coinciden con saldos cero con mucha más frecuencia que los casos de no\_fraud. Esto puede explicarse fácilmente por el hecho de que los estafadores suelen enviar dinero a cuentas ficticias vacías.

## 4.5 Tiempo en el que se hacen las transacciones

Si bien puede parecer irracional que los fraudes ocurran en una determinada instancia de tiempo, podemos graficar la cantidad frente al tiempo coloreada por la columna isFraud en caso de que haya algún patrón obvio.

Debido a que las observaciones son tantas, se cubren unas a otras y nuestra trama deja de ser informativa. Para eso, muestrearemos aleatoriamente el 5% de nuestros datos. De esta forma reducimos los puntos graficados evitando de estas manera posibles sesgos.

transacciones %>% sample\_n(floor(nrow(transacciones)\*.05)) %>%   
 ggplot()+  
 geom\_jitter(aes( x = step, y = log(amount+1,10)), alpha = .9, col = "#2ff7b5")+  
 facet\_grid(~isFraud)+  
 labs(x = "Hora", y = "Cantidad (en escala log10 )", title = "Transacciones por Hora", subtitle = "")+  
 theme(axis.text.x = element\_text(angle = 90)) +   
 theme\_minimal()



Los casos de fraude y no fraude se ven dispersos aleatoriamente a lo largo del tiempo y no parecen formar grupos. Sin embargo, hay algunas concentraciones leves de fraudes en algunas horas. Esto podría deberse al hecho de que cuando los estafadores logran cometer un fraude, intentan maximizar la cantidad de transacciones en el estrecho margen de tiempo que tienen.

## 4.6 Correlación

Luego, verificaremos la correlación entre nuestras características numéricas con la ayuda de un diagrama de pares (ggpairs). Debido a que el diagrama de pares es muy costoso desde el punto de vista computacional, elegiremos el 5 % del conjunto de datos para el diagrama.

set.seed(123)  
transacciones %>% sample\_n(floor(nrow(transacciones)\*.1)) -> transacciones\_small

transacciones\_small %>% str()

## 'data.frame': 277040 obs. of 13 variables:  
## $ step : Factor w/ 743 levels "1","2","3","4",..: 329 16 283 307 131 43 137 225 180 709 ...  
## $ type : Factor w/ 5 levels "CASH\_IN","CASH\_OUT",..: 2 5 2 2 2 5 2 2 2 2 ...  
## $ amount : num 32300 254581 1148 169419 322109 ...  
## $ Old\_Balance\_of\_Origin : num 473223 0 0 31139 93670 ...  
## $ New\_Balance\_of\_Origin : num 440923 0 0 0 0 ...  
## $ Old\_Balance\_of\_Destination : num 151180 1638300 56228 0 0 ...  
## $ New\_Balance\_of\_Destination : num 183480 2102542 57376 169419 322109 ...  
## $ isFraud : Factor w/ 2 levels "No\_Fraud","Fraud": 1 1 1 1 1 1 1 1 1 1 ...  
## $ isRound : Factor w/ 2 levels "float","round": 1 1 1 1 1 1 1 1 1 1 ...  
## $ isZero\_Old\_Balance\_of\_Or : logi FALSE TRUE TRUE FALSE FALSE FALSE ...  
## $ isZero\_New\_Balance\_of\_Or : logi FALSE TRUE TRUE TRUE TRUE TRUE ...  
## $ isZero\_Old\_Balance\_of\_Destina: logi FALSE FALSE FALSE TRUE TRUE FALSE ...  
## $ isZero\_New\_Balance\_of\_Destina: logi FALSE FALSE FALSE FALSE FALSE FALSE ...

#Tratamos la variable isRound  
transacciones$isRound <- as.character(transacciones$isRound)  
transacciones$isRound[transacciones$isRound =="float"] <- 0  
transacciones$isRound[transacciones$isRound =="round"] <- 1  
transacciones$isRound <- as.numeric(transacciones$isRound)

#tratamis la variable type   
transacciones$type <- as.character(transacciones$type)  
transacciones$type[transacciones$type =="TRANSFER"] = 0  
transacciones$type[transacciones$type =="CASH\_OUT"] = 1  
transacciones$type <- as.numeric(transacciones$type)  
unique(transacciones$type)

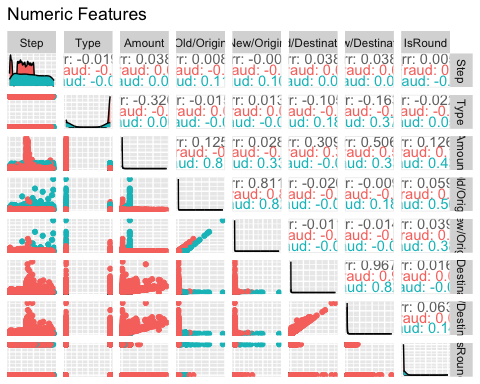
## [1] 0 1

#Pasamo s la variable step que es de tipo factor a numeric   
transacciones$step <- as.numeric(as.character(transacciones$step))

head(transacciones)

## step type amount Old\_Balance\_of\_Origin New\_Balance\_of\_Origin  
## 1 1 0 181.0 181.00 0  
## 2 1 1 181.0 181.00 0  
## 3 1 1 229133.9 15325.00 0  
## 4 1 0 215310.3 705.00 0  
## 5 1 0 311685.9 10835.00 0  
## 6 1 1 110414.7 26845.41 0  
## Old\_Balance\_of\_Destination New\_Balance\_of\_Destination isFraud isRound  
## 1 0 0.00 Fraud 1  
## 2 21182 0.00 Fraud 1  
## 3 5083 51513.44 No\_Fraud 0  
## 4 22425 0.00 No\_Fraud 0  
## 5 6267 2719172.89 No\_Fraud 0  
## 6 288800 2415.16 No\_Fraud 0  
## isZero\_Old\_Balance\_of\_Or isZero\_New\_Balance\_of\_Or  
## 1 FALSE TRUE  
## 2 FALSE TRUE  
## 3 FALSE TRUE  
## 4 FALSE TRUE  
## 5 FALSE TRUE  
## 6 FALSE TRUE  
## isZero\_Old\_Balance\_of\_Destina isZero\_New\_Balance\_of\_Destina  
## 1 TRUE TRUE  
## 2 FALSE TRUE  
## 3 FALSE FALSE  
## 4 FALSE TRUE  
## 5 FALSE FALSE  
## 6 FALSE FALSE

#labels for plotting  
labs <- c("Step", "Type",'Amount',"Old/Origin", "New/Origin", "Old/Destination","New/Destination", "IsRound")   
#Create a smaller dataset  
transacciones\_small <- transacciones %>% sample\_n(floor(nrow(transacciones)\*.1))  
  
#Pairplot  
transacciones\_small[ ,sapply(transacciones,is.numeric)] %>%   
 ggpairs(mapping=ggplot2::aes(colour = transacciones\_small$isFraud), title = "Numeric Features",columnLabels = labs, axisLabels = "none")



Del diagrama podemos ver que hay dos correlaciones muy altas se encuentran dentro de nuestros datos numéricos. Parece que old destination y new origin se correlacionan ya que tiene un factor de correlación alto 0.76 y step y new destination con 0.972 de factor de correlación son las variables mas correlacionadas. Para decidir qué columnas conservar, utilizaremos la función findCorrelation del paquete caret.

* Su primer argumento va a ser la tabla de correlación de las características numéricas de transacciones.
* Al elegir un límite (límite de correlación absoluta) de 0,7, significa que para cada correlación por debajo de -0,8 o por encima de 0,9, obtenemos una sugerencia de una columna para eliminar.
* names = TRUE asignará un vector de caracteres a nuestra variable recién creada, lo que facilitará su uso con la función select().

Además, notamos una leve concentración de casos de fraude con valores cercanos a cero para algunas de las columnas de balance. Esto fortalece nuestra suposición de que los saldos cero podrían coincidir con el fraude.

high\_cor\_feats <- findCorrelation(cor(transacciones[ ,sapply(transacciones,is.numeric)]), cutoff = .8, verbose = TRUE,   
 names = TRUE, exact = TRUE)

## Compare row 7 and column 6 with corr 0.97   
## Means: 0.249 vs 0.126 so flagging column 7   
## All correlations <= 0.8

high\_cor\_feats

## [1] "New\_Balance\_of\_Destination"

#Eliminamos las variables altamente correlacionadas  
transacciones <- transacciones %>% select( -high\_cor\_feats )

#Eliminamos las variables altamente correlacionadas  
head(transacciones)

## step type amount Old\_Balance\_of\_Origin New\_Balance\_of\_Origin  
## 1 1 0 181.0 181.00 0  
## 2 1 1 181.0 181.00 0  
## 3 1 1 229133.9 15325.00 0  
## 4 1 0 215310.3 705.00 0  
## 5 1 0 311685.9 10835.00 0  
## 6 1 1 110414.7 26845.41 0  
## Old\_Balance\_of\_Destination isFraud isRound isZero\_Old\_Balance\_of\_Or  
## 1 0 Fraud 1 FALSE  
## 2 21182 Fraud 1 FALSE  
## 3 5083 No\_Fraud 0 FALSE  
## 4 22425 No\_Fraud 0 FALSE  
## 5 6267 No\_Fraud 0 FALSE  
## 6 288800 No\_Fraud 0 FALSE  
## isZero\_New\_Balance\_of\_Or isZero\_Old\_Balance\_of\_Destina  
## 1 TRUE TRUE  
## 2 TRUE FALSE  
## 3 TRUE FALSE  
## 4 TRUE FALSE  
## 5 TRUE FALSE  
## 6 TRUE FALSE  
## isZero\_New\_Balance\_of\_Destina  
## 1 TRUE  
## 2 TRUE  
## 3 FALSE  
## 4 TRUE  
## 5 FALSE  
## 6 FALSE

## 4.7 Proceso PCA

Tanto el análisis de componentes principales, principal componente analysis (PCA) en inglés, como la descomposición de valores singulares, singular value decomposition (SVD) en inglés, son técnicas que nos permiten trabajar con nuevas características llamadas componentes, que ciertamente son independientes entre sí.

En realidad, estas dos técnicas nos permiten representar el juego de datos en un nuevo sistema de coordenadas que llamamos componentes principales. Este sistema está mejor adaptado a la distribución del juego de datos, por lo que recoge mejor su variabilidad.

#transacciones$type[transacciones$type == 1] <- "CASH\_OUT"  
#transacciones$type[transacciones$type == 0] <- "TRANSFER"  
#transacciones$isRound [transacciones$isRound == 1] <- "round"  
#transacciones$isRound [transacciones$isRound == 0] <- "float"  
#transacciones$isRound <- as.character(transacciones$isRound)  
#transacciones$type <- as.character(transacciones$type)

A continuación vamos a normalizar las demás columnas para asegurarnos de que cada variable contribuye por igual a nuestro análisis.

# Definimos la función de normalización  
 nor <-function(x) { (x -min(x))/(max(x)-min(x))}  
# Guardamos el nuevo dataset normalizado  
r = c(1,2,3,4,5,6,9,10,11,12)  
transacciones\_pca<- transacciones %>% select(all\_of(r))  
transacciones\_pca\_NOR <- as.data.frame(lapply(transacciones\_pca, nor))

Aplicamos el análisis de componentes principales al row data set por eso empezamos ejecutando la función **prcomp()**.

set.seed(123)  
pca.transacciones <- prcomp(transacciones\_pca\_NOR)  
summary(pca.transacciones)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6 PC7  
## Standard deviation 0.5379 0.3936 0.3123 0.2697 0.19077 0.07693 0.01222  
## Proportion of Variance 0.4404 0.2357 0.1484 0.1107 0.05539 0.00901 0.00023  
## Cumulative Proportion 0.4404 0.6762 0.8246 0.9352 0.99063 0.99963 0.99986  
## PC8 PC9 PC10  
## Standard deviation 0.008244 0.00460 0.001582  
## Proportion of Variance 0.000100 0.00003 0.000000  
## Cumulative Proportion 0.999960 1.00000 1.000000

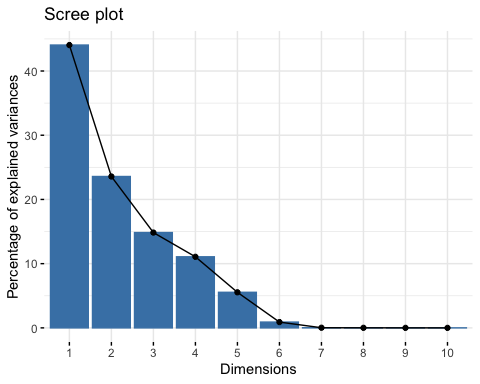
Como puede observarse la función summary, nos devuelve la proporción de varianza aplicada al conjunto total de cada atributo. Gracias a ello, el atributo 1 explica el 0.43 de variabilidad del total de datos; en cambio, el atributo 11 explica el 0.00000

A continuación se muestra un histograma para ver el peso de cada atributo sobre el conjunto total de datos:

library('factoextra')  
#Los valores propios corresponden a la cantidad de variación explicada por cada componente principal (PC).  
ev= get\_eig(pca.transacciones)  
ev

## eigenvalue variance.percent cumulative.variance.percent  
## Dim.1 2.893848e-01 4.404208e+01 44.04208  
## Dim.2 1.548901e-01 2.357305e+01 67.61513  
## Dim.3 9.751605e-02 1.484117e+01 82.45630  
## Dim.4 7.272239e-02 1.106777e+01 93.52408  
## Dim.5 3.639194e-02 5.538566e+00 99.06264  
## Dim.6 5.918151e-03 9.006958e-01 99.96334  
## Dim.7 1.492660e-04 2.271711e-02 99.98606  
## Dim.8 6.795577e-05 1.034233e-02 99.99640  
## Dim.9 2.116149e-05 3.220612e-03 99.99962  
## Dim.10 2.503026e-06 3.809408e-04 100.00000

fviz\_eig(pca.transacciones)

 En este ejercicio se decidió utilizar el método de Kaiser para decidir cuál de las variables obtenidas serán escogidas. Este criterio mantendrá todas aquellas variables cuya varianza sea superior a 1.

# Calculamos la varianza de los principales componentes a partir de la desviación estándar  
var\_trans <- pca.transacciones$sdev^2  
  
var\_trans

## [1] 2.893848e-01 1.548901e-01 9.751605e-02 7.272239e-02 3.639194e-02  
## [6] 5.918151e-03 1.492660e-04 6.795577e-05 2.116149e-05 2.503026e-06

Con los resultados obtenidos es muy complicado decidir cuáles son los componentes principales componentes a elegir. **Este hecho podría estar causado por no haber escalado los datos previamente.** Por tanto, el siguiente paso es escalar los datos y volver a calcular la varianza para ver qué datos seleccionas

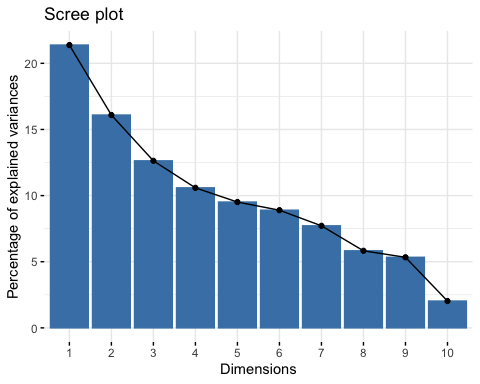
# Escalamos los datos  
trans\_scale <- scale(transacciones\_pca\_NOR)  
# Calculamos las componentes principales  
pca.trans\_scale <- prcomp(trans\_scale)  
# Mostramos la varianza de dichas variables:  
var\_trans\_scale <- pca.trans\_scale$sdev^2  
head(var\_trans\_scale)

## [1] 2.1381114 1.6090761 1.2628086 1.0586373 0.9512237 0.8896291

Después de analizar la varianza y aplicando el criterio de Káiser nos quedaremos con las principales componentes 1,2, 3 y 4 que son las superiores a 1. Este criterio tiene el problema de sobreestimar el número de factores, pero a pesar de ello es lo que aplicaremos para analizar los resultados.

Mostramos el histograma de porcentaje de varianza contado con los datos escalados:

set.seed(123)  
fviz\_eig(pca.trans\_scale)



ev = get\_eig(pca.trans\_scale)  
ev

## eigenvalue variance.percent cumulative.variance.percent  
## Dim.1 2.1381114 21.381114 21.38111  
## Dim.2 1.6090761 16.090761 37.47187  
## Dim.3 1.2628086 12.628086 50.09996  
## Dim.4 1.0586373 10.586373 60.68633  
## Dim.5 0.9512237 9.512237 70.19857  
## Dim.6 0.8896291 8.896291 79.09486  
## Dim.7 0.7715649 7.715649 86.81051  
## Dim.8 0.5825929 5.825929 92.63644  
## Dim.9 0.5336001 5.336001 97.97244  
## Dim.10 0.2027560 2.027560 100.00000

La suma de todos los valores propios da una varianza total de 10.

La proporción de variación explicada por cada valor propio se da en la segunda columna. Por ejemplo, 19.4 dividido por 10 es igual a 0,194, o alrededor del 19.4% de la variación se explica por ese primer valor propio. El porcentaje acumulado explicado se obtiene sumando las sucesivas proporciones de variación explicadas para obtener el total acumulado. Por ejemplo, el 19.4% más el 14.7% equivalen al 34%, y así sucesivamente. Por tanto, alrededor del 34% de la variación se explica por los dos primeros valores propios juntos.

Los valores propios se pueden utilizar para determinar el número de componentes principales a retener después de la PCA (Kaiser 1961):

* Un valor propio > 1 indica que los PCs representan más varianza de la que representa una de las variables originales de los datos estandarizados. Esto se utiliza habitualmente como punto de corte para el que se conservan los PCs. Esto sólo es cierto cuando los datos están estandarizados.
* También podemos limitar el número de componentes a ese número que representa una determinada fracción de la varianza total. Por ejemplo, si estamos satisfecho con el 70% de la varianza total explicada, utilizamos el número de componentes para conseguirlo que son las 4 componentes princiapls visto antes.

Continuamos con el análisis de los componentes principales. Después de aplicar el método Káiser se han seleccionado los 4 componentes principales.

var <- get\_pca\_var(pca.trans\_scale)  
var

## Principal Component Analysis Results for variables  
## ===================================================  
## Name Description   
## 1 "$coord" "Coordinates for the variables"   
## 2 "$cor" "Correlations between variables and dimensions"  
## 3 "$cos2" "Cos2 for the variables"   
## 4 "$contrib" "contributions of the variables"

Los componentes de get\_pca\_var() se pueden utilizar en el diagrama de variables de la siguiente manera:

* **var$coord**: coordenadas de variables para crear un diagrama de racimo
* **var$cos2**: representa la calidad de representación de las variables en el mapa de factores. Se calcula como las coordenadas al cuadrado: var.cos2 = var.coord \* var.coord.
* **var$contrib**: contiene las contribuciones (en porcentaje) de las variables a los componentes principales. La contribución de una variable (var) a un determinado componente principal es (en porcentaje): (var.cos2 \* 100) / (cos2 total del componente).

#Utilizamos las 4 componentes principales utilizadas anteriormente  
head(var$coord[,1:4],10)

## Dim.1 Dim.2 Dim.3 Dim.4  
## step 0.005155226 0.06310478 0.19487280 -0.624902576  
## type -0.108434361 -0.54017638 -0.41388897 -0.192517136  
## amount 0.029537393 0.70204806 0.43682408 0.005833958  
## Old\_Balance\_of\_Origin -0.822724354 0.34337996 -0.18912016 0.045831987  
## New\_Balance\_of\_Origin -0.801373776 0.30462993 -0.32092722 -0.032380251  
## Old\_Balance\_of\_Destination 0.176663656 0.60247168 0.03240367 -0.241873364  
## isZero\_Old\_Balance\_of\_Or 0.510910967 0.32817457 -0.54227676 0.079056526  
## isZero\_New\_Balance\_of\_Or 0.614840804 0.11599466 0.03015624 0.239001559  
## isZero\_Old\_Balance\_of\_Destina -0.320549707 -0.34968587 0.65196481 0.066526344  
## isZero\_New\_Balance\_of\_Destina -0.182640698 0.05759375 0.05358623 0.708228757

### 4.7.1 Calidad de representación

La calidad de representación de las variables en el mapa de factores se llama cos2 (coseno cuadrado, coordenadas cuadradas). Podemos acceder al cos2 de la siguiente manera:

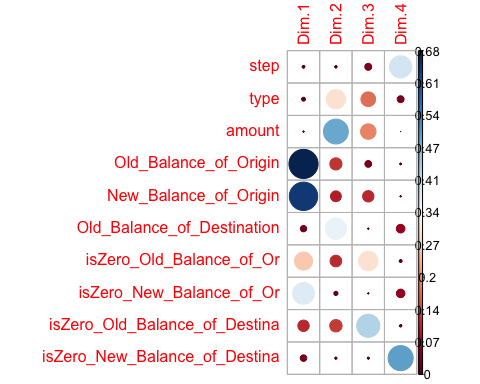
head(var$cos2[,1:4],13)

## Dim.1 Dim.2 Dim.3  
## step 2.657636e-05 0.003982214 0.0379754096  
## type 1.175801e-02 0.291790518 0.1713040801  
## amount 8.724576e-04 0.492871480 0.1908152730  
## Old\_Balance\_of\_Origin 6.768754e-01 0.117909798 0.0357664360  
## New\_Balance\_of\_Origin 6.421999e-01 0.092799397 0.1029942790  
## Old\_Balance\_of\_Destination 3.121005e-02 0.362972119 0.0010499979  
## isZero\_Old\_Balance\_of\_Or 2.610300e-01 0.107698548 0.2940640809  
## isZero\_New\_Balance\_of\_Or 3.780292e-01 0.013454762 0.0009093985  
## isZero\_Old\_Balance\_of\_Destina 1.027521e-01 0.122280208 0.4250581153  
## isZero\_New\_Balance\_of\_Destina 3.335762e-02 0.003317040 0.0028714845  
## Dim.4  
## step 3.905032e-01  
## type 3.706285e-02  
## amount 3.403507e-05  
## Old\_Balance\_of\_Origin 2.100571e-03  
## New\_Balance\_of\_Origin 1.048481e-03  
## Old\_Balance\_of\_Destination 5.850272e-02  
## isZero\_Old\_Balance\_of\_Or 6.249934e-03  
## isZero\_New\_Balance\_of\_Or 5.712175e-02  
## isZero\_Old\_Balance\_of\_Destina 4.425754e-03  
## isZero\_New\_Balance\_of\_Destina 5.015880e-01

library(corrplot)

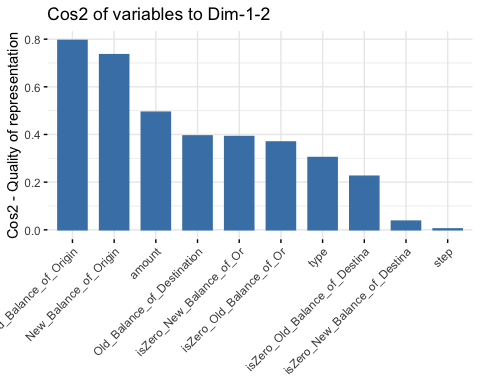
## corrplot 0.90 loaded

corrplot(var$cos2[,1:4], is.corr=FALSE)



También es posible crear un diagrama de barras de variables cos2 mediante la función fviz\_cos2():

fviz\_cos2(pca.trans\_scale, choice = "var", axes = 1:2)



* Un cos2 elevado indica una buena representación de la variable en el componente principal. En ese caso, la variable se coloca cerca de la circunferencia del círculo de correlación.
* Un cos2 bajo indica que la variable no está perfectamente representada por los PC. En este caso, la variable se encuentra cerca del centro del círculo.

Para una variable dada, la suma del cos2 de todos los componentes principales es igual a uno.

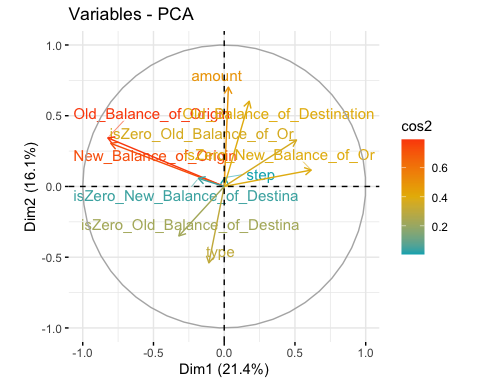
Si una variable está perfectamente representada por sólo dos componentes principales (Dim.1 y Dim.2), la suma del cos2 en estos dos PC es igual a uno. En ese caso las variables se colocarán en el círculo de correlaciones.

Para algunas variables, pueden ser necesarios más de 2 componentes para representar perfectamente los datos. En este caso, las variables se sitúan dentro del círculo de correlaciones.

En resumen:

* Los valores de cos2 se utilizan para estimar la calidad de la representación
* Cuanto más cercana esté una variable en el círculo de correlaciones, mejor será su representación en el mapa de factores (y más importante es interpretar estos componentes)
* Las variables que están cercanas al centro de la trama son menos importantes para los primeros componentes.

fviz\_pca\_var(pca.trans\_scale,  
 col.var = "cos2",   
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
 repel = TRUE   
 )



### 4.7.2 Contribución

Las contribuciones de las variables en la contabilización de la variabilidad de un determinado componente principal se expresan en porcentaje. Las variables que están correlacionadas con PC1 (es decir, Dim.1) y PC2 (es decir, Dim.2) son las más importantes para explicar la variabilidad en el conjunto de datos.

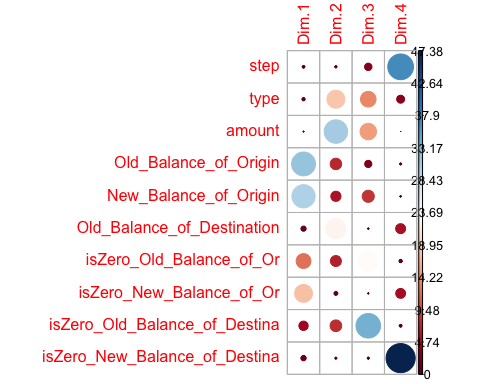
Las variables que no están correlacionadas con ningún PC o con las últimas dimensiones son variables con una contribución baja y pueden eliminarse para simplificar el análisis global. La contribución de las variables puede extraerse de la siguiente manera:

head(var$contrib[,1:4],13)

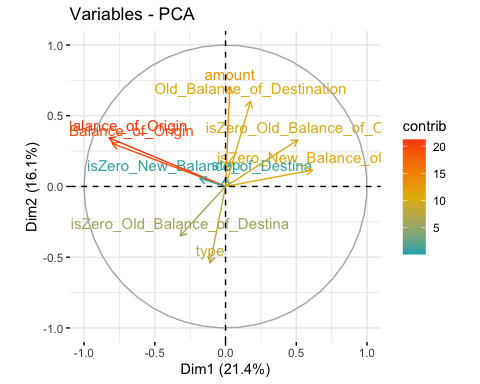
## Dim.1 Dim.2 Dim.3 Dim.4  
## step 0.001242983 0.2474845 3.00721827 36.887348636  
## type 0.549925080 18.1340411 13.56532465 3.500995842  
## amount 0.040805058 30.6307132 15.11038805 0.003214989  
## Old\_Balance\_of\_Origin 31.657629129 7.3277951 2.83229282 0.198422167  
## New\_Balance\_of\_Origin 30.035850454 5.7672473 8.15596937 0.099040594  
## Old\_Balance\_of\_Destination 1.459701680 22.5577972 0.08314783 5.526229308  
## isZero\_Old\_Balance\_of\_Or 12.208438826 6.6931918 23.28651321 0.590375410  
## isZero\_New\_Balance\_of\_Or 17.680520427 0.8361793 0.07201397 5.395780568  
## isZero\_Old\_Balance\_of\_Destina 4.805741967 7.5994050 33.65974309 0.418061456  
## isZero\_New\_Balance\_of\_Destina 1.560144397 0.2061456 0.22738874 47.380531031

Cuanto mayor sea el vale de la contribución, mayor contribución habrá en el componente.

corrplot(var$contrib[,1:4], is.corr=FALSE)

 Las variables más importantes (o que contribuyen) pueden resaltarse en la gráfica de correlación de la siguiente manera:

fviz\_pca\_var(pca.trans\_scale, col.var = "contrib",  
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")  
 )

 Las variables correlacionadas positivas apuntan al mismo lado de la trama. Las variables correlacionadas negativas apuntan a lados opuestos del gráfico. Por ejemplo, vemos que que New\_Balance\_of\_Origin y type apuntan a direcciones opuestas por tanto no están nada correlaciones, además lo hemos visto antes ya que tienen un coeficiente de correlación de 0.014.

Se observa que la variable que más aporta a los componentes principales son **new y old balance of origin**. Esto se debe a que están correlacionadas.

Por tanto, se puede afirmar que las técnicas aplicadas para saber si una transacción es fraudulenta o no, tiene un impacto fuerte sobre el new y old balance of origin sobre sus atributos correlacionados, mientras que otros atributos de las transacciones son los que crean las diferencias, como pueden ser isRound o Type con las que existen correlaciones muy bajas. Por tanto, se podría afirmar que las características principales que permiten diferenciar, clasificar y llegar a predecir de qué tipo de vino se trata, son aquellas variables que tienen una gran correlación.

## 4.8 Submuestreo de datos previo al modelado

El submuestreo se refiere a un grupo de técnicas diseñadas para equilibrar la distribución de clases para un conjunto de datos de clasificación que tiene una distribución de clases sesgada.

Una distribución de clases desequilibrada tendrá una o más clases con pocos ejemplos (las clases minoritarias) y una o más clases con muchos ejemplos (las clases mayoritarias). Se entiende mejor en el contexto de un problema de clasificación binaria (dos clases) donde la clase 0 es la clase mayoritaria y la clase 1 es la clase minoritaria (en nuestro caso, fraude o no fraude)

Existen varias técnicas diferentes en la práctica para tratar con conjuntos de datos desequilibrados. La clase de técnicas más común es el muestreo: cambiar los datos presentados al modelo submuestreando clases comunes, sobremuestreando (duplicando) clases raras, o ambos.

La principal ventaja del submuestreo es que los cientificos de datos pueden corregir datos no balanceados para reducir el riesgo de que su análisis o algoritmo de aprendizaje automático se desvíe hacia la mayoría (en este caso fraude). Sin volver a muestrear, los científicos podrían ejecutar un modelo de clasificación con un 90 % de precisión. Sin embargo, en una inspección más cercana, encontrarán que los resultados están en gran medida dentro de la clase mayoritaria. Esto se conoce como la paradoja de la precisión.

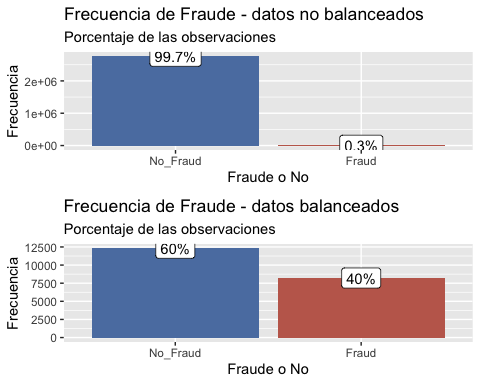
En pocas palabras, los eventos de las minorías son más difíciles de predecir porque hay menos (es decir las transaccions fraudulentas). Un algoritmo tiene menos información de la que aprender. Pero el remuestreo a través del submuestreo puede corregir este problema y hacer que la clase minoritaria sea igual a la clase mayoritaria a los efectos del análisis de datos.

Otras ventajas del submuestreo incluyen menos requisitos de almacenamiento y mejores tiempos de ejecución para el análisis. Menos datos significa que las empresas necesitan menos almacenamiento y tiempo para obtener información valiosa. En nuestro tenemos alrededor de 6M de transacciones que en la vide real no es nada del otro mundo, pero esta claro que no disponemos de los mismos recursos para poder analisis tantos datos en unos tiempos razonables.

En la práctica, los métodos de muestreo se utilizan a menudo para equilibrar conjuntos de datos con distribuciones de clases sesgadas porque varios clasificadores han mostrado empíricamente un mejor rendimiento cuando se entrenó en un conjunto de datos equilibrado o balanceado. Sin embargo, estos estudios no implican que los clasificadores no puedan aprender de conjuntos de datos no balanceados Por ejemplo, otros estudios también han demostrado que algunos clasificadores no mejoran sus actuaciones cuando el conjunto de datos de entrenamiento se equilibra utilizando técnicas de muestreo.

Como hemos visto anteriormente, tenemos un conjunto de datos para nada balanceado donde la gran mayoria de registros no son fraude, más del 99%. El objetivo es crear un conjunto de datos más balanaceado. En mi opinión, bajar la prevalencia de la clase mayoritaria al 50% elimina totalmente el valor informativo de este desequilibrio. Es por eso que en este apartado crearemos una variable objetivo del 60% - 40%, las instancias minoritarias equivalen a 40% y el número de instancias mayoritarias muestreadas será el 60 %.

#encontramos el numero de instancias de fraudes   
minoria<- transacciones %>% filter(isFraud == "Fraud") %>% nrow()  
  
mayoria <- transacciones %>% filter(isFraud == "No\_Fraud") %>% sample\_n( minoria\*6/4)  
  
#creamos el nuevo data set balanceado donde la cantidad de filaas para fraude vendra dada por 6/4 partes de la clase minoritaria  
transacciones\_bal <- transacciones %>% filter(isFraud == "Fraud") %>% rbind( mayoria ) %>% arrange(step)  
  
p5 <- transacciones %>% ggplot(aes(x = isFraud, y = (..count..))) +   
 geom\_bar(fill = c( "#5b7fb0", "#c2695b" ), stat = "count")+  
 geom\_label(stat='count',aes( label= paste0( round( ((..count..)/sum(..count..)) ,4)\*100 , "%" ) ) )+  
 labs(x = "Fraude o No", y = "Frecuencia", title = "Frecuencia de Fraude - datos no balanceados", subtitle = "Porcentaje de las observaciones")  
  
p6 <- transacciones\_bal %>% ggplot(aes(x = isFraud, y = (..count..))) +   
 geom\_bar(fill = c( "#5b7fb0", "#c2695b" ), stat = "count")+  
 geom\_label(stat='count',aes( label= paste0( round( ((..count..)/sum(..count..)) ,4)\*100 , "%" ) ) )+  
 labs(x = "Fraude o No", y = "Frecuencia", title = "Frecuencia de Fraude - datos balanceados ", subtitle = "Porcentaje de las observaciones")  
  
grid.arrange(p5,p6)



## 4.9 Conjunto de datos de entretamiento y test

Para este apartado y los apartados futuros relacionados utilizaremos funcion del paquete CARET. El paquete caret (Classification And REgression Training) contiene funciones para agilizar el proceso de entrenamiento del modelo para problemas complejos de regresión y clasificación.

Después de dividir nuestros datos en entrenamiento (train) y conjunto de prueba (test), crearemos una rutina de preprocesamiento basada en los datos de entrenamiento (el conjunto de prueba se considera invisible).

Nuestra transformación de preproceso incluirá:

* centrado, (es decir, restando de cada columna su propia media)
* escalado, (es decir, dividiendo por la varianza de la columna)

Luego, lo usaremos para transformar ambos conjuntos.

El mismo proceso se aplicará tanto en el submuestreo como en el conjunto de datos original. Necesitamos evaluar nuestro algoritmo basado en la vida real y no en los datos balanceados.

### 4.9.1 Dataset original

Primero, dividimos los datos en dos grupos: un conjunto de entrenamiento y un conjunto de prueba. Para hacer esto, se usa la función createDataPartition:

#Cremos indices para las filas test/train   
indx <- createDataPartition(y = transacciones$isFraud, p = .75, list = FALSE)   
train\_set <- transacciones[indx,]   
test\_set <- transacciones[-indx,]  
  
## Centrar y escalar los predictores para el entrenamiento conjunto y todas las muestras futuras.  
pp <- preProcess( train\_set[, sapply(train\_set,is.numeric) ] ,   
 method = c("center", "scale"))  
  
#Transformamos el conjunto de entrenamiento  
train\_set <- predict(pp, newdata = train\_set)  
  
#Transformamos el conjunto de prueba  
test\_set <- predict(pp, newdata = test\_set)  
  
train\_set %>% str()

## 'data.frame': 2077807 obs. of 12 variables:  
## $ step : num -1.7 -1.7 -1.7 -1.7 -1.7 ...  
## $ type : num -2.05 0.488 0.488 -2.05 -2.05 ...  
## $ amount : num -0.36005 -0.36005 -0.09972 -0.11544 -0.00585 ...  
## $ Old\_Balance\_of\_Origin : num -0.193 -0.193 -0.131 -0.191 -0.149 ...  
## $ New\_Balance\_of\_Origin : num -0.109 -0.109 -0.109 -0.109 -0.109 ...  
## $ Old\_Balance\_of\_Destination : num -0.404 -0.399 -0.403 -0.399 -0.403 ...  
## $ isFraud : Factor w/ 2 levels "No\_Fraud","Fraud": 2 2 1 1 1 1 1 1 1 1 ...  
## $ isRound : num 9.373 9.373 -0.107 -0.107 -0.107 ...  
## $ isZero\_Old\_Balance\_of\_Or : logi FALSE FALSE FALSE FALSE FALSE FALSE ...  
## $ isZero\_New\_Balance\_of\_Or : logi TRUE TRUE TRUE TRUE TRUE TRUE ...  
## $ isZero\_Old\_Balance\_of\_Destina: logi TRUE FALSE FALSE FALSE FALSE FALSE ...  
## $ isZero\_New\_Balance\_of\_Destina: logi TRUE TRUE FALSE TRUE FALSE FALSE ...

### 4.9.2 Dataset balanceado

Primero, dividimos los datos en dos grupos: un conjunto de entrenamiento y un conjunto de prueba. Para hacer esto, se usa la función createDataPartition. De forma predeterminada, createDataPartition realiza una división aleatoria estratificada de los datos.

#Cremos indices para las filas test/train   
indx <- createDataPartition(y = transacciones\_bal$isFraud, p = .75, list = FALSE)   
train\_set\_b <- transacciones\_bal[indx,]   
test\_set\_b <- transacciones\_bal[-indx,]  
  
## Centrar y escalar los predictores para el entrenamiento conjunto y todas las muestras futuras.  
pp\_b <- preProcess( train\_set\_b[, sapply(train\_set\_b,is.numeric) ] ,   
 method = c("center", "scale"))  
  
#Transformos el conjunto de entrenamiento  
train\_set\_b<- predict(pp\_b, newdata = train\_set\_b)  
  
#Transformamos el conjunto de prueba  
test\_set\_b <- predict(pp\_b, newdata = test\_set\_b)  
train\_set\_b$step <- as.numeric(as.character(train\_set\_b$step))  
test\_set\_b$step <- as.numeric(as.character(test\_set\_b$step))  
train\_set\_b %>% str()

## 'data.frame': 15400 obs. of 12 variables:  
## $ step : num -1.57 -1.57 -1.57 -1.57 -1.57 ...  
## $ type : num -1.481 -1.481 0.675 -1.481 0.675 ...  
## $ amount : num -0.442 -0.441 -0.441 -0.431 -0.431 ...  
## $ Old\_Balance\_of\_Origin : num -0.289 -0.288 -0.288 -0.281 -0.281 ...  
## $ New\_Balance\_of\_Origin : num -0.0686 -0.0686 -0.0686 -0.0686 -0.0686 ...  
## $ Old\_Balance\_of\_Destination : num -0.332 -0.332 -0.325 -0.332 -0.33 ...  
## $ isFraud : Factor w/ 2 levels "No\_Fraud","Fraud": 2 2 2 2 2 2 2 2 2 2 ...  
## $ isRound : num 5.57 5.57 5.57 5.57 5.57 ...  
## $ isZero\_Old\_Balance\_of\_Or : logi FALSE FALSE FALSE FALSE FALSE TRUE ...  
## $ isZero\_New\_Balance\_of\_Or : logi TRUE TRUE TRUE TRUE TRUE TRUE ...  
## $ isZero\_Old\_Balance\_of\_Destina: logi TRUE TRUE FALSE TRUE FALSE FALSE ...  
## $ isZero\_New\_Balance\_of\_Destina: logi TRUE TRUE TRUE TRUE FALSE FALSE ...

# 5 Modelado

En este apartado empezará el modelado y trataremos diversos algoritmos, tantos superviados como no supervisados. Para encontrar los parámetros óptimos del modelo, utilizaremos la validación cruzada para entrenarlo. Eso significa que cada vez que entrenamos nuestro algoritmo, dividimos los datos de nuestro entrenamiento en particiones, usamos una de ellas y luego lo evaluamos al predecir el resto de las particiones. En nuestro caso, medimos las predicciones resumiendo las probabilidades de cada clase con twoClassSummary. Esta función encaja bien ya que las opciones de predicción (los valores únicos de la variable de destino) son dos, Fraude o No Fraude.

Finalmente, para elegir diferentes medidas de rendimiento, se le dan argumentos adicionales a trainControl. El argumento summaryFunction se usa para pasar una función que toma los valores observados y predichos y estima alguna medida de rendimiento. Dos de estas funciones ya están incluidas en el paquete: defaultSummary y twoClassSummary. Este último calculará medidas específicas para problemas de dos clases, como el área bajo la curva ROC, la sensibilidad y la especificidad. Dado que la curva ROC se basa en las probabilidades de clase pronosticadas (que no se calculan automáticamente), se requiere otra opción. La opción classProbs = TRUE se utiliza para incluir estos cálculos.

Por último, la función seleccionará los parámetros de ajuste asociados con los mejores resultados. Dado que estamos utilizando medidas de rendimiento personalizadas, también se debe especificar el criterio que debe optimizarse. En la llamada para entrenar, podemos usar metric = “ROC” para hacer esto.

#Seteamos los parametros de entrenamiento  
tr <- trainControl(method = "repeatedcv", #repeated cross validation   
 number = 9, #9 folds  
 repeats = 3, #3 veces  
 classProbs = TRUE, #calculamos probabilidades para cada clase  
 summaryFunction = twoClassSummary)

## 5.1 Algoritmos supervisados

**El aprendizaje supervisado**  es un enfoque de aprendizaje automático que se define mediante el uso de conjuntos de **datos etiquetados**. Estos conjuntos de datos están diseñados para entrenar o “supervisar” algoritmos para clasificar los datos o predecir los resultados con precisión. Usando entradas y salidas etiquetadas, el modelo puede medir su precisión y aprender con el tiempo.

El aprendizaje supervisado se puede separar en dos tipos de problemas en la minería de datos: *clasificación y regresión*

### 5.1.1 Classification and Regression Tree

**CART (Classification And Regression Tree)** es una variación del algoritmo del árbol de decisión (Decission tree). Puede manejar tareas de clasificación y regresión. CART fue producido por primera vez por Leo Breiman, Jerome Friedman, Richard Olshen y Charles Stone en 1984.

CART es un algoritmo predictivo utilizado en el aprendizaje automático y explica cómo se pueden predecir los valores de la variable objetivo en función de otras variables llamadas predictoras. Es un árbol de decisiones en el que cada bifurcación se divide en una variable predictora y cada nodo tiene una predicción para la variable objetivo al final.

El algoritmo **rpart** que utilizaremos a continuación por defecto la impureza de Gini para dividir el conjunto de datos en un árbol de decisión. Lo hace buscando la mejor homogeneidad para los subnodos, con la ayuda del criterio del índice de Gini.

El índice Gini es una métrica para las tareas de clasificación en CART. Almacena la suma de probabilidades al cuadrado de cada clase. Calcula el grado de probabilidad de que una variable específica esté mal clasificada cuando se elige aleatoriamente y una variación del coeficiente de Gini. Funciona en variables categóricas, proporciona resultados de “éxito” o “fracaso” en nuestro caso fraude o no fraude y, por lo tanto, solo realiza divisiones binarias.

#Model Training - Regression Tree con CARET  
start\_time <- Sys.time()  
set.seed(123)  
rpart\_model\_b <- train(isFraud ~ ., #queremos predecir si es fraude o no usando todo los predictores  
 data = train\_set\_b, #usamos el dataset alanceado  
 method = "rpart", # libreria que contiene el algoritmo del arbol  
 tuneLength = 10, # El número de diferentes valores predeterminados utilizados para optimizar el parámetro principal del modelo.  
 metric = "ROC", # estimar el éxito usando área bajo la curva ROC  
 trControl = tr, #usamos cross validation definida arriba  
 parms=list(split='information'))   
  
end\_time <- Sys.time()  
  
end\_time - start\_time

## Time difference of 9.877772 secs

Una vez tenemos el modelo, podemos comprobar su calidad prediciendo la clase para los datos de prueba que nos hemos reservado al principio.

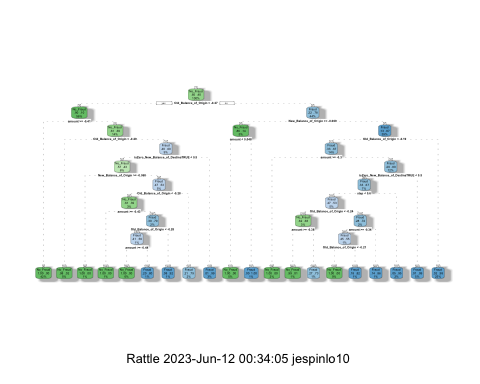
rpart\_test\_pred\_b <- predict(rpart\_model\_b, test\_set\_b) # prediccion en el conjunto de prueba  
confusionMatrix(test\_set\_b$isFraud, rpart\_test\_pred\_b) # resultado de predicciones

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No\_Fraud Fraud  
## No\_Fraud 2991 88  
## Fraud 16 2037  
##   
## Accuracy : 0.9797   
## 95% CI : (0.9755, 0.9834)  
## No Information Rate : 0.5859   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.958   
##   
## Mcnemar's Test P-Value : 3.352e-12   
##   
## Sensitivity : 0.9947   
## Specificity : 0.9586   
## Pos Pred Value : 0.9714   
## Neg Pred Value : 0.9922   
## Prevalence : 0.5859   
## Detection Rate : 0.5828   
## Detection Prevalence : 0.6000   
## Balanced Accuracy : 0.9766   
##   
## 'Positive' Class : No\_Fraud   
##

Podemos ver que tenemos un precisión del 92%, lo cual és un porcentaje alto. Sin embargo no podemos sacar de momento ninguna conclusión puesto que no hemos probado otros algoritmos con los cual comparar los resultados del modelo generado.

A continuación mostraremos el plot del model creado previamente.

if(!require(rattle)){  
 install.packages('rattle', repos='http://cran.us.r-project.org')  
 library(rattle)  
}  
fancyRpartPlot(rpart\_model\_b$finalModel)

 Otro algoritmo que podemos utilizar en R es el **C5.0**. Este método se basa en la entropía para seleccionar características óptimas para dividir. La alta entropía significa que hay poca o ninguna homogeneidad dentro del conjunto de valores de clase. El algoritmo tiene como objetivo encontrar divisiones que reduzcan la entropía.

La variable por la que clasificaremos es el campo isFraud donde vemos si la transaccion es fraudulent o no, que está en la columna 1 De esta forma, tendremos un conjunto de datos para el entrenamiento y uno para la validación.

head(transacciones\_bal)

## step type amount Old\_Balance\_of\_Origin New\_Balance\_of\_Origin  
## 1 1 0 181 181 0  
## 2 1 1 181 181 0  
## 3 1 0 2806 2806 0  
## 4 1 1 2806 2806 0  
## 5 1 0 20128 20128 0  
## 6 1 1 20128 20128 0  
## Old\_Balance\_of\_Destination isFraud isRound isZero\_Old\_Balance\_of\_Or  
## 1 0 Fraud 1 FALSE  
## 2 21182 Fraud 1 FALSE  
## 3 0 Fraud 1 FALSE  
## 4 26202 Fraud 1 FALSE  
## 5 0 Fraud 1 FALSE  
## 6 6268 Fraud 1 FALSE  
## isZero\_New\_Balance\_of\_Or isZero\_Old\_Balance\_of\_Destina  
## 1 TRUE TRUE  
## 2 TRUE FALSE  
## 3 TRUE TRUE  
## 4 TRUE FALSE  
## 5 TRUE TRUE  
## 6 TRUE FALSE  
## isZero\_New\_Balance\_of\_Destina  
## 1 TRUE  
## 2 TRUE  
## 3 TRUE  
## 4 TRUE  
## 5 TRUE  
## 6 FALSE

set.seed(666)  
y <- transacciones\_bal[,7]   
n = c(1,2,3,4,5,6,8,9,10,11,12)  
X <- transacciones\_bal[,n]

Podemos crear directamente una función de un parámetro, en este caso del “split\_prop”. Definimos un parámetro que controla el split de forma dinámica en el test.

split\_prop <- 4  
indexes = sample(1:nrow(transacciones\_bal), size=floor(((split\_prop-1)/split\_prop)\*nrow(transacciones\_bal)))  
trainX<-X[indexes,]  
trainy<-y[indexes]  
testX<-X[-indexes,]  
testy<-y[-indexes]

Después de una extracción aleatoria de casos, es altamente recomendable efectuar un análisis de datos mínimo para asegurarnos de no obtener clasificadores sesgados por los valores que contiene cada muestra. En este caso, verificaremos que la proporción de las transacciones es más o menos constante en ambos conjuntos:

summary(trainX);

## step type amount Old\_Balance\_of\_Origin  
## Min. : 1.0 Min. :0.0000 Min. : 0 Min. : 0   
## 1st Qu.:160.0 1st Qu.:0.0000 1st Qu.: 91882 1st Qu.: 0   
## Median :276.0 Median :1.0000 Median : 212480 Median : 37013   
## Mean :293.2 Mean :0.6852 Mean : 774046 Mean : 681595   
## 3rd Qu.:397.0 3rd Qu.:1.0000 3rd Qu.: 508767 3rd Qu.: 306465   
## Max. :743.0 Max. :1.0000 Max. :30238359 Max. :59585040   
## New\_Balance\_of\_Origin Old\_Balance\_of\_Destination isRound   
## Min. : 0 Min. : 0 Min. :0.00000   
## 1st Qu.: 0 1st Qu.: 0 1st Qu.:0.00000   
## Median : 0 Median : 216454 Median :0.00000   
## Mean : 83395 Mean : 1271637 Mean :0.03169   
## 3rd Qu.: 0 3rd Qu.: 1113761 3rd Qu.:0.00000   
## Max. :49585040 Max. :236230517 Max. :1.00000   
## isZero\_Old\_Balance\_of\_Or isZero\_New\_Balance\_of\_Or  
## Mode :logical Mode :logical   
## FALSE:11040 FALSE:1060   
## TRUE :4359 TRUE :14339   
##   
##   
##   
## isZero\_Old\_Balance\_of\_Destina isZero\_New\_Balance\_of\_Destina  
## Mode :logical Mode :logical   
## FALSE:10083 FALSE:12281   
## TRUE :5316 TRUE :3118   
##   
##   
##

summary(trainy)

## No\_Fraud Fraud   
## 9209 6190

summary(testX)

## step type amount Old\_Balance\_of\_Origin  
## Min. : 1.0 Min. :0.0000 Min. : 0 Min. : 0   
## 1st Qu.:157.0 1st Qu.:0.0000 1st Qu.: 94550 1st Qu.: 0   
## Median :273.0 Median :1.0000 Median : 212096 Median : 31145   
## Mean :290.4 Mean :0.6889 Mean : 785904 Mean : 696096   
## 3rd Qu.:394.0 3rd Qu.:1.0000 3rd Qu.: 522465 3rd Qu.: 306587   
## Max. :743.0 Max. :1.0000 Max. :23697296 Max. :45674548   
## New\_Balance\_of\_Origin Old\_Balance\_of\_Destination isRound   
## Min. : 0 Min. : 0 Min. :0.00000   
## 1st Qu.: 0 1st Qu.: 0 1st Qu.:0.00000   
## Median : 0 Median : 217175 Median :0.00000   
## Mean : 93364 Mean : 1212922 Mean :0.02883   
## 3rd Qu.: 0 3rd Qu.: 1069225 3rd Qu.:0.00000   
## Max. :35674548 Max. :78123403 Max. :1.00000   
## isZero\_Old\_Balance\_of\_Or isZero\_New\_Balance\_of\_Or  
## Mode :logical Mode :logical   
## FALSE:3639 FALSE:319   
## TRUE :1494 TRUE :4814   
##   
##   
##   
## isZero\_Old\_Balance\_of\_Destina isZero\_New\_Balance\_of\_Destina  
## Mode :logical Mode :logical   
## FALSE:3365 FALSE:4108   
## TRUE :1768 TRUE :1025   
##   
##   
##

summary(testy)

## No\_Fraud Fraud   
## 3110 2023

Se crea el árbol de decisión usando los datos de entrenamiento (no hay que olvidar que la variable outcome es de tipo factor). Además tenemos que pasar alguna variables booleanas a enteros 1/0 ya que el algoritmo C50 no aceptat variables booleanas.

trainX$isZero\_Old\_Balance\_of\_Or = as.numeric(trainX$isZero\_Old\_Balance\_of\_Or)  
trainX$isZero\_New\_Balance\_of\_Or = as.numeric(trainX$isZero\_New\_Balance\_of\_Or)  
trainX$isZero\_Old\_Balance\_of\_Destina = as.numeric(trainX$isZero\_Old\_Balance\_of\_Destina)  
trainX$isZero\_New\_Balance\_of\_Destina = as.numeric(trainX$isZero\_New\_Balance\_of\_Destina)

str(trainX)

## 'data.frame': 15399 obs. of 11 variables:  
## $ step : num 546 332 346 403 16 477 554 18 183 252 ...  
## $ type : num 1 1 0 0 1 1 1 1 1 1 ...  
## $ amount : num 84907 96106 384085 6392 350923 ...  
## $ Old\_Balance\_of\_Origin : num 30056 211691 0 6392 51842 ...  
## $ New\_Balance\_of\_Origin : num 0 115585 0 0 0 ...  
## $ Old\_Balance\_of\_Destination : num 0 689797 523589 0 244917 ...  
## $ isRound : num 0 0 0 0 0 0 1 0 0 0 ...  
## $ isZero\_Old\_Balance\_of\_Or : num 0 0 1 0 0 0 0 0 1 1 ...  
## $ isZero\_New\_Balance\_of\_Or : num 1 0 1 1 1 1 1 1 1 1 ...  
## $ isZero\_Old\_Balance\_of\_Destina: num 1 0 0 1 0 0 0 0 0 0 ...  
## $ isZero\_New\_Balance\_of\_Destina: num 0 0 0 1 1 0 0 0 0 0 ...

A continuación aplicamos el agoritmo C5.0:

set.seed(123)  
trainy = as.factor(trainy)  
C50\_model <- C50::C5.0(trainX, trainy,rules=TRUE )  
C50\_model

##   
## Call:  
## C5.0.default(x = trainX, y = trainy, rules = TRUE)  
##   
## Rule-Based Model  
## Number of samples: 15399   
## Number of predictors: 11   
##   
## Number of Rules: 37   
##   
## Non-standard options: attempt to group attributes

summary(C50\_model)

##   
## Call:  
## C5.0.default(x = trainX, y = trainy, rules = TRUE)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Mon Jun 12 00:34:08 2023  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 15399 cases (12 attributes) from undefined.data  
##   
## Rules:  
##   
## Rule 1: (2198/2, lift 1.7)  
## amount > 52661.86  
## Old\_Balance\_of\_Origin > 2231  
## Old\_Balance\_of\_Origin <= 52663  
## -> class No\_Fraud [0.999]  
##   
## Rule 2: (637, lift 1.7)  
## amount <= 106460.7  
## isZero\_New\_Balance\_of\_Or <= 0  
## -> class No\_Fraud [0.998]  
##   
## Rule 3: (598, lift 1.7)  
## amount > 21990.52  
## Old\_Balance\_of\_Origin > 12494  
## Old\_Balance\_of\_Origin <= 21785.61  
## -> class No\_Fraud [0.998]  
##   
## Rule 4: (1743/2, lift 1.7)  
## amount > 31358.72  
## Old\_Balance\_of\_Origin > 2231  
## Old\_Balance\_of\_Origin <= 31856  
## isZero\_New\_Balance\_of\_Destina <= 0  
## -> class No\_Fraud [0.998]  
##   
## Rule 5: (1363/2, lift 1.7)  
## amount > 49966.72  
## amount <= 106460.7  
## Old\_Balance\_of\_Origin <= 50209  
## -> class No\_Fraud [0.998]  
##   
## Rule 6: (2222/4, lift 1.7)  
## amount > 56847.17  
## Old\_Balance\_of\_Origin > 2231  
## Old\_Balance\_of\_Origin <= 57461  
## isZero\_New\_Balance\_of\_Destina <= 0  
## -> class No\_Fraud [0.998]  
##   
## Rule 7: (2185/4, lift 1.7)  
## amount > 89388.81  
## Old\_Balance\_of\_Origin > 2231  
## Old\_Balance\_of\_Origin <= 89209  
## -> class No\_Fraud [0.998]  
##   
## Rule 8: (2611/3, lift 1.7)  
## amount > 106460.7  
## Old\_Balance\_of\_Origin <= 106566  
## isZero\_Old\_Balance\_of\_Or <= 0  
## -> class No\_Fraud [0.998]  
##   
## Rule 9: (936/1, lift 1.7)  
## isZero\_New\_Balance\_of\_Or <= 0  
## isZero\_New\_Balance\_of\_Destina <= 0  
## -> class No\_Fraud [0.998]  
##   
## Rule 10: (1477/4, lift 1.7)  
## type <= 0  
## isZero\_New\_Balance\_of\_Or > 0  
## isZero\_Old\_Balance\_of\_Destina <= 0  
## -> class No\_Fraud [0.997]  
##   
## Rule 11: (1968/4, lift 1.7)  
## type > 0  
## amount > 126375.2  
## amount <= 233123.6  
## Old\_Balance\_of\_Origin <= 125863  
## -> class No\_Fraud [0.997]  
##   
## Rule 12: (1144/3, lift 1.7)  
## amount > 447637.7  
## Old\_Balance\_of\_Origin <= 436757.8  
## -> class No\_Fraud [0.997]  
##   
## Rule 13: (1638/6, lift 1.7)  
## amount > 1946.74  
## amount <= 106460.7  
## Old\_Balance\_of\_Origin <= 2231  
## isZero\_New\_Balance\_of\_Destina <= 0  
## -> class No\_Fraud [0.996]  
##   
## Rule 14: (1665/5, lift 1.7)  
## amount > 9475.21  
## amount <= 106460.7  
## Old\_Balance\_of\_Origin <= 9426  
## -> class No\_Fraud [0.996]  
##   
## Rule 15: (1744/6, lift 1.7)  
## amount > 11738.48  
## amount <= 106460.7  
## Old\_Balance\_of\_Origin <= 12494  
## isZero\_New\_Balance\_of\_Destina <= 0  
## -> class No\_Fraud [0.996]  
##   
## Rule 16: (2185/8, lift 1.7)  
## type > 0  
## amount > 155963.9  
## amount <= 301841.4  
## Old\_Balance\_of\_Origin <= 157029  
## -> class No\_Fraud [0.996]  
##   
## Rule 17: (3100/10, lift 1.7)  
## amount > 233123.6  
## Old\_Balance\_of\_Origin <= 233622.1  
## -> class No\_Fraud [0.996]  
##   
## Rule 18: (2203/7, lift 1.7)  
## amount > 301841.4  
## Old\_Balance\_of\_Origin <= 303147  
## -> class No\_Fraud [0.996]  
##   
## Rule 19: (4425/29, lift 1.7)  
## Old\_Balance\_of\_Origin <= 129  
## isZero\_New\_Balance\_of\_Destina <= 0  
## -> class No\_Fraud [0.993]  
##   
## Rule 20: (10083/2173, lift 1.3)  
## isZero\_Old\_Balance\_of\_Destina <= 0  
## -> class No\_Fraud [0.784]  
##   
## Rule 21: (1549/1, lift 2.5)  
## type > 0  
## Old\_Balance\_of\_Origin > 436757.8  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.999]  
##   
## Rule 22: (3061/1, lift 2.5)  
## type <= 0  
## isZero\_Old\_Balance\_of\_Destina > 0  
## isZero\_New\_Balance\_of\_Destina > 0  
## -> class Fraud [0.999]  
##   
## Rule 23: (1716/3, lift 2.5)  
## type <= 0  
## Old\_Balance\_of\_Origin > 281347.3  
## isZero\_New\_Balance\_of\_Or > 0  
## isZero\_Old\_Balance\_of\_Destina > 0  
## -> class Fraud [0.998]  
##   
## Rule 24: (191, lift 2.5)  
## isRound > 0  
## isZero\_New\_Balance\_of\_Destina > 0  
## -> class Fraud [0.995]  
##   
## Rule 25: (233/4, lift 2.4)  
## amount <= 155963.9  
## Old\_Balance\_of\_Origin > 125863  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.979]  
##   
## Rule 26: (46, lift 2.4)  
## amount <= 56847.17  
## Old\_Balance\_of\_Origin > 52663  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.979]  
##   
## Rule 27: (134/2, lift 2.4)  
## amount <= 31358.72  
## Old\_Balance\_of\_Origin > 21785.61  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.978]  
##   
## Rule 28: (2104/52, lift 2.4)  
## type > 0  
## Old\_Balance\_of\_Origin > 205701  
## isZero\_New\_Balance\_of\_Or > 0  
## isZero\_New\_Balance\_of\_Destina <= 0  
## -> class Fraud [0.975]  
##   
## Rule 29: (338/9, lift 2.4)  
## amount <= 205628.6  
## Old\_Balance\_of\_Origin > 157029  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.971]  
##   
## Rule 30: (31, lift 2.4)  
## amount <= 11738.48  
## Old\_Balance\_of\_Origin > 9426  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.970]  
##   
## Rule 31: (208/6, lift 2.4)  
## amount <= 126375.2  
## Old\_Balance\_of\_Origin > 106566  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.967]  
##   
## Rule 32: (141/4, lift 2.4)  
## amount <= 9475.21  
## Old\_Balance\_of\_Origin > 2231  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.965]  
##   
## Rule 33: (345/15, lift 2.4)  
## amount <= 89388.81  
## Old\_Balance\_of\_Origin > 57461  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.954]  
##   
## Rule 34: (185/8, lift 2.4)  
## amount <= 21990.52  
## Old\_Balance\_of\_Origin > 12494  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.952]  
##   
## Rule 35: (157/8, lift 2.3)  
## amount <= 106460.7  
## Old\_Balance\_of\_Origin > 89209  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.943]  
##   
## Rule 36: (48/2, lift 2.3)  
## amount <= 1946.74  
## Old\_Balance\_of\_Origin > 129  
## Old\_Balance\_of\_Origin <= 2231  
## -> class Fraud [0.940]  
##   
## Rule 37: (276/16, lift 2.3)  
## amount <= 52661.86  
## Old\_Balance\_of\_Origin > 31856  
## isZero\_New\_Balance\_of\_Or > 0  
## -> class Fraud [0.939]  
##   
## Default class: No\_Fraud  
##   
##   
## Evaluation on training data (15399 cases):  
##   
## Rules   
## ----------------  
## No Errors  
##   
## 37 119( 0.8%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 9132 77 (a): class No\_Fraud  
## 42 6148 (b): class Fraud  
##   
##   
## Attribute usage:  
##   
## 91.50% Old\_Balance\_of\_Origin  
## 85.46% isZero\_Old\_Balance\_of\_Destina  
## 85.40% isZero\_New\_Balance\_of\_Destina  
## 68.44% amount  
## 61.40% type  
## 53.96% isZero\_New\_Balance\_of\_Or  
## 16.96% isZero\_Old\_Balance\_of\_Or  
## 1.24% isRound  
##   
##   
## Time: 0.2 secs

El árbol obtenido tiene una tasa de error del 0.6%, con lo que podemos decir que se ha obtenido una muy buena clasificación.

Como contrapartida observamos que se han generado 44 reglas, por eso se explicarán las que se han considerado más relevantes y han obtenido una mayor validez:

**Regla 23:**

* type = TRANSFER
* isZero\_Old\_Balance\_of\_Destina > 0
* isZero\_New\_Balance\_of\_Destina > 0

Fraude 100%

**Regla 1:** - step <= 405 - amount > 30548.46 - Old\_Balance\_of\_Origin > 145 - Old\_Balance\_of\_Origin <= 30827 - isZero\_New\_Balance\_of\_Destina <= 0

No Fraude 99%

**Regla 2:** - amount > 38180.99 - Old\_Balance\_of\_Origin > 145 - Old\_Balance\_of\_Origin <= 37323.15

No Fraude 99%

**Regla 3:** - amount > 51856.4 - Old\_Balance\_of\_Origin > 145 - Old\_Balance\_of\_Origin <= 51711

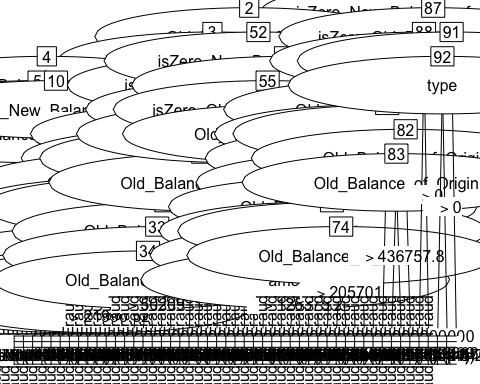
No Fraude 99%

**Regla 4:** - amount > 22695.74 - Old\_Balance\_of\_Origin > 145 - Old\_Balance\_of\_Origin <= 22807

No Fraude 99%

A continuación se mostrará el árbol obtenido, que tendrá muchas ramas debido a todas las reglas generadas.

set.seed(123)  
model <- C50::C5.0(trainX, trainy)  
plot(model)

 Como podemos ver, debido a las diversas ramas que tenemos, resulta imposible mostrar un gráfico legible.

Una vez tenemos el modelo, podemos comprobar su calidad prediciendo la clase para los datos de prueba que nos hemos reservado al principio.

testX$isZero\_Old\_Balance\_of\_Or = as.numeric(testX$isZero\_Old\_Balance\_of\_Or)  
testX$isZero\_New\_Balance\_of\_Or = as.numeric(testX$isZero\_New\_Balance\_of\_Or)  
testX$isZero\_Old\_Balance\_of\_Destina = as.numeric(testX$isZero\_Old\_Balance\_of\_Destina)  
testX$isZero\_New\_Balance\_of\_Destina = as.numeric(testX$isZero\_New\_Balance\_of\_Destina)

set.seed(123)  
predicted\_model <- predict( model, testX, type="class" )  
print(sprintf("La precisión del árbol es: %.4f %%",100\*sum(predicted\_model == testy) /  
 length(predicted\_model)))

## [1] "La precisión del árbol es: 98.7921 %"

Cuando existen pocas clases, la calidad de la predicción se puede analizar mediante una matriz de confusión que identifica los tipos de errores cometidos.

mat\_conf<-table(testy,Predicted=predicted\_model)  
mat\_conf

## Predicted  
## testy No\_Fraud Fraud  
## No\_Fraud 3071 39  
## Fraud 23 2000

Por lo tanto vemos que el porcentaje de precisión del árbol C5.0 es ligeramente mejor que el árbol rpart utilizado anteriormente.

### 5.1.2 Random Forest

El algorismo **Random Forest** consiste en una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de ellos. Es una modificación sustancial de bagging que construye una larga colección de árboles no correlacionados y después los promedia.

Bagging es una técnica usada para reducir la varianza de las predicciones a través de la combinación de los resultados de varios clasificadores, cada uno moldeados con diferentes subconjuntos presos de la misma población. Random forest permite trabajar en base de datos muy grandes, donde permite crear múltiples subconjuntos de datos, construir múltiples modelos y combinar estos modelos. Es muy útil para problemas de clasificación y regresión. Al igual que con el árbol, usaremos los mismos parámetros de validación cruzada.

Esta vez, en lugar de tuneLength que conecta los valores predeterminados para los hiperparámetros del modelo, usaremos un TuneGrid donde tenemos estos parámetros predeterminados. Estos hiperparámetros incluyen:

**Ntree:** Número de árboles por crear. Esta variable no debe ser un número demasiado pequeño, para garantizar que cada fila de entrada se predice al menos algunas veces.

**Mtry:** número de variables mostradas al azar como candidatos a cada división.

Como hicimos con el árbol, entrenaremos nuestro bosque usando el conjunto de datos balanceado. Luego, evaluaremos su éxito en la misma muestra aleatoria que generamos anteriormente.

Utilizaremos libreria random forest, que nos permitirá calcular el modelo Random Forest:

if(!require(randomForest)){  
 install.packages('randomForest', repos='http://cran.us.r-project.org')  
 library(randomForest)  
}

Generamos el modelo random forest con hiperparametros por defecto. Por defecto, el número de árboles es 500 (ntree) y el número de variables probadas en cada división es 3 (mtry) en ese caso.

start\_time <- Sys.time()  
set.seed(123)  
default\_RFmodel <- randomForest(isFraud~ ., data = train\_set\_b[,1:12], importance = TRUE)  
end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 20.11276 secs

print(default\_RFmodel)

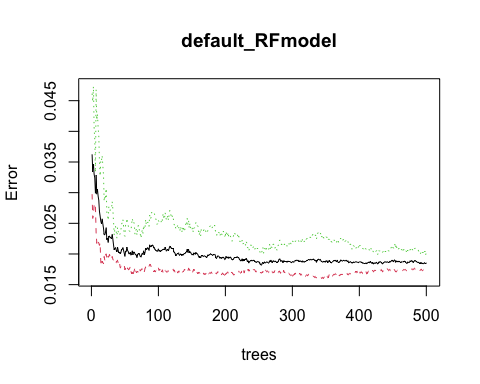
##   
## Call:  
## randomForest(formula = isFraud ~ ., data = train\_set\_b[, 1:12], importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 1.86%  
## Confusion matrix:  
## No\_Fraud Fraud class.error  
## No\_Fraud 9080 160 0.01731602  
## Fraud 126 6034 0.02045455

Por defecto, el número de árboles es 500 y el número de variables probadas en cada división es 3 en ese caso. Se ha obtenido una tasa de error de 7.44%.

Al crearse distintos número de árboles (500) es muy difícil representar el resultado en un único árbol como se ha realizado en el caso anterior, por lo que en este caso se graficará en forma de gráfica

A continuación vamos a graficar el modelo generado previamente:

plot(default\_RFmodel)



En este gráfico se observa la tasa de error por el número de árboles. Puede observarse que a medida que aumentan el número de árboles, la tasa de error disminuye.

A continuación se intenta mejorar la tasa de error modificando el parámetro mtry, dónde ahora será 1 y manteniendo el número de árboles utilizados para el caso anterior.

start\_time <- Sys.time()  
set.seed(123)  
RFmodel\_mtry1 <- randomForest(isFraud~ ., data = train\_set\_b[,1:11], ntree = 500, mtry = 1 , importance = TRUE)  
end\_time <- Sys.time()  
end\_time - start\_time

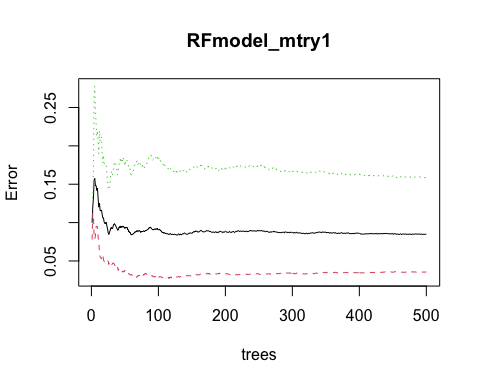
## Time difference of 10.14425 secs

print(RFmodel\_mtry1)

##   
## Call:  
## randomForest(formula = isFraud ~ ., data = train\_set\_b[, 1:11], ntree = 500, mtry = 1, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 1  
##   
## OOB estimate of error rate: 8.48%  
## Confusion matrix:  
## No\_Fraud Fraud class.error  
## No\_Fraud 8912 328 0.03549784  
## Fraud 978 5182 0.15876623

Modificando el parámetro mtry a 1 no se ha logrado reducir la tasa de error, no obstante el tiempo de ejecución ha disminuido notablemente a 9.1 segundos.

plot(RFmodel\_mtry1)

 Finalmente, probaremos a aumentar el valor de mtry para ver como se comporta el algoritmo. El numero de predictors incluidos en la división de nodos será 7.

set.seed(123)  
start\_time <- Sys.time()  
RFmodel\_mtry7 <- randomForest(isFraud~ ., data = train\_set\_b[,1:11], ntree = 500, mtry = 7 , importance = TRUE)  
end\_time <- Sys.time()  
end\_time - start\_time

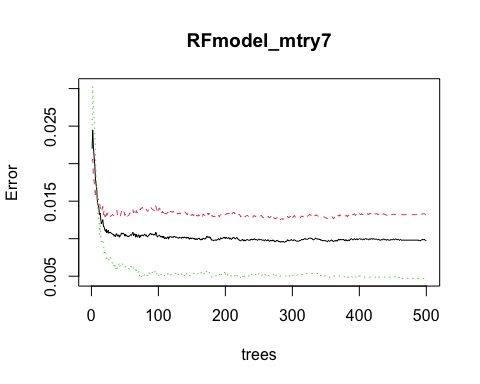
## Time difference of 21.16824 secs

print(RFmodel\_mtry7)

##   
## Call:  
## randomForest(formula = isFraud ~ ., data = train\_set\_b[, 1:11], ntree = 500, mtry = 7, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 7  
##   
## OOB estimate of error rate: 0.97%  
## Confusion matrix:  
## No\_Fraud Fraud class.error  
## No\_Fraud 9119 121 0.013095238  
## Fraud 29 6131 0.004707792

Modificando el parámetro mtry a 7 se ha logrado reducir la tasa de error a 0.79%, no obstante el tiempo de ejecución ha aumentado notablemente a 17 segundos. Lo cual es normal puesto que usando más variables predictoras en el cálculo.

plot(RFmodel\_mtry7)

 Como en el caso anterior, se observa que a medida que se incrementa el número de árboles, la tasa de error disminuye.

A continuación obtendremos una tabla mostrando, en primer lugar, la clasificación real del dataset y en segundo lugar cómo la ha clasificado nuestro algoritmo. También se extraerá el porcentaje de fiabilidad de nuestro algoritmo y podremos comparar si ha tenido mejores resultados que con el algoritmo C50.

Obtenemos la tabla de clasificación de nuestro juego de datos

default\_RFmodel$confusion

## No\_Fraud Fraud class.error  
## No\_Fraud 9080 160 0.01731602  
## Fraud 126 6034 0.02045455

Lo aplicamos a nuestros datos de prueba y obtenemos la tabla de clasificación y la tasa de fiabilidad.

test\_set\_b$step <- as.numeric(as.character(test\_set\_b$step))  
predValid <- predict(default\_RFmodel, test\_set\_b, type = "class")  
mean(predValid == test\_set\_b$isFraud)

## [1] 0.9863601

table(predValid,test\_set\_b$isFraud)

##   
## predValid No\_Fraud Fraud  
## No\_Fraud 3038 29  
## Fraud 41 2024

Observamos que hemos obtenido un porcentaje de fiabilidad del 98% en cuanto a la classificación de transacciones.

Finalmente vamos a obtener la tabla de clasificación y la tasa de fiabilidad para el random forest creado cuyo mtry era 7 y daba mejor tasa de fiabilidad.

predValid7 <- predict(RFmodel\_mtry7, test\_set\_b, type = "class")  
mean(predValid7 == test\_set\_b$isFraud)

## [1] 0.9939595

table(predValid7,test\_set\_b$isFraud)

##   
## predValid7 No\_Fraud Fraud  
## No\_Fraud 3054 6  
## Fraud 25 2047

Observamos que para **mtry = 7**, hemos obtenido un porcentaje de fiabilidad del 99% en cuanto a la classificación de transacciones.Es ligeramente mejor a la precisión encontrada en el algoritmo C5.0 calculada antes, la cual era 98%.

## 5.2 Algoritmos no supervisados

**El aprendizaje no supervisado** utiliza algoritmos de aprendizaje automático para analizar y agrupar conjuntos de **datos sin etiquetar**. Estos algoritmos descubren patrones ocultos en los datos sin necesidad de intervención humana (por tanto, están “sin supervisar”).

Los modelos de aprendizaje no supervisado se utilizan para tres tareas principales: *agrupación, asociación y reducción de la dimensionalidad.*

La principal diferencia es el uso de conjuntos de datos. Por decirlo simplemente, el aprendizaje supervisado utiliza datos de entrada y salida etiquetados, mientras que un algoritmo de aprendizaje no supervisado no.

### 5.2.1 K-Means

Durante este aparatdo deberemos aplicar un modelo **no supervisado** y basado en el concepto de distancia, sobre el juego de datos. El algoritmo escogido para realizar un modelo no supervisado es el **k-means** basado en la métrica euclídea, cuyo objetivo es conocer y agrupar los datos en grupos (clustering). Por eso lo primero que hay que hacer es un análisis de los posibles clusters a los que pueden pertenecer las transacciones. Por eso generaremos un nuevo juego de datos eliminando la variable isfraud. De esta forma conseguimos un juego de datos idóneo para aplicar un algoritmo no supervisado.

Utilizaremos un conjunto de datos normalizado donde también se tendra que eliminar la variable isFraud.

#Comenzamos por pasar las variables booleanas a intengers con 0s y 1s  
transacciones\_bal$isZero\_Old\_Balance\_of\_Or = as.numeric(transacciones\_bal$isZero\_Old\_Balance\_of\_Or)  
transacciones\_bal$isZero\_New\_Balance\_of\_Or = as.numeric(transacciones\_bal$isZero\_New\_Balance\_of\_Or)  
transacciones\_bal$isZero\_Old\_Balance\_of\_Destina = as.numeric(transacciones\_bal$isZero\_Old\_Balance\_of\_Destina)  
transacciones\_bal$isZero\_New\_Balance\_of\_Destina = as.numeric(transacciones\_bal$isZero\_New\_Balance\_of\_Destina)  
transacciones\_bal$step <- as.numeric(as.character(transacciones\_bal$step))

A continuación vamos a normalizar las demás columnas para asegurarnos de que cada variable contribuye por igual a nuestro análisis.

# Definim la funció de normalització  
 nor <-function(x) { (x -min(x))/(max(x)-min(x))}  
# Guardem un nou dataset normalitzat  
r = c(1,3,4,5,6,9,10,11,12)  
transacciones\_bal2<- transacciones\_bal %>% select(all\_of(r))  
#transacciones\_bal\_NOR <- as.data.frame(lapply(transacciones\_bal2[, c(1,3,4,5,6,8,9,10,11)], nor))  
transacciones\_bal2$isFraud <- NULL  
n = c(1,2,3,4,5,6,7,8,9)  
transacciones\_bal2<- transacciones\_bal2 %>% select(all\_of(n))  
transacciones\_bal\_NOR <- as.data.frame(lapply(transacciones\_bal2, nor))

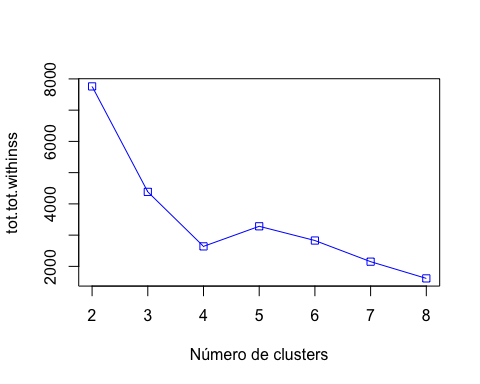
head(transacciones\_bal\_NOR)

## step amount Old\_Balance\_of\_Origin New\_Balance\_of\_Origin  
## 1 0 5.985775e-06 3.037675e-06 0  
## 2 0 5.985775e-06 3.037675e-06 0  
## 3 0 9.279604e-05 4.709236e-05 0  
## 4 0 9.279604e-05 4.709236e-05 0  
## 5 0 6.656446e-04 3.378029e-04 0  
## 6 0 6.656446e-04 3.378029e-04 0  
## Old\_Balance\_of\_Destination isZero\_Old\_Balance\_of\_Or isZero\_New\_Balance\_of\_Or  
## 1 0.000000e+00 0 1  
## 2 8.966665e-05 0 1  
## 3 0.000000e+00 0 1  
## 4 1.109171e-04 0 1  
## 5 0.000000e+00 0 1  
## 6 2.653341e-05 0 1  
## isZero\_Old\_Balance\_of\_Destina isZero\_New\_Balance\_of\_Destina  
## 1 1 1  
## 2 0 1  
## 3 1 1  
## 4 0 1  
## 5 1 1  
## 6 0 0

A continuación se aplicará la métrica Euclidiana que consiste en encontrar la menor suma de los cuadrados de las distancias de los puntos de cada grupo respecto a su centro (withinss), con la mayor separación entre centros de grupos (betweenss). Este método es conocido como Elbow (codo), que no es más que la selección del número de clusters sobre la base de la inspección de la gráfica que se obtiene al iterar con el mismo conjunto de datos para diferentes valores del número de clusters . Se seleccionará el valor que se encuentra en el codo de la curva.

library(usethis)   
usethis::edit\_r\_environ()

set.seed(666)  
library(cluster)  
library(factoextra)  
library(NbClust)  
d <- daisy(transacciones\_bal\_NOR)   
resultados <- rep(0, 8)  
for (i in c(2,3,4,5,6,7,8))  
{  
 fit <- kmeans(transacciones\_bal\_NOR, i)  
 resultados[i] <- fit$tot.withinss  
}  
plot(2:8,resultados[2:8],type="o",col="blue",pch=0,xlab="Número de clusters",ylab="tot.tot.withinss")

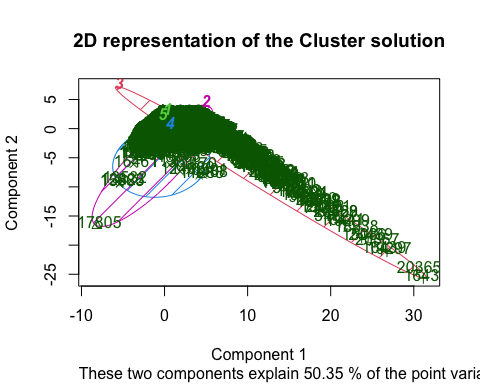


En este caso se observa que la curva comienza a estabilizarse a partir del valor 5, por lo que el valor óptimo para aplicar el algoritmo no supervisado K-means con esta métrica es 5.

##Para tener el mismo resultado cada vez que ejecutamos el código, hacemos set.seed(4).  
##Esto significa que R probará 4 tareas iniciales aleatorias diferentes y  
#después seleccionará la que tenga la variación más baja dentro del clúster  
set.seed(123)  
transacciones5<- kmeans(transacciones\_bal\_NOR, 5)

Para visualizar el clúster podemos utilizar la función clusplot. Miremos la agrupación con 5 clusters que son los que ha encontrado con el método elbow.

clusplot(transacciones\_bal\_NOR, transacciones5$cluster, main='2D representation of the Cluster solution',  
 color=TRUE, shade=TRUE,  
 labels=2, lines=0)

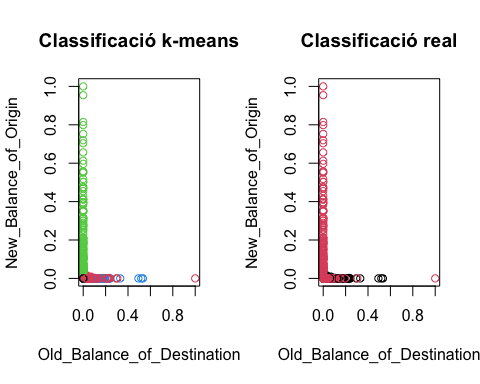


Utilizaremos ahora las variables que tienen mayor correlación como hemos visto en apartados anteriores y compararemos dos a dos:

head(transacciones\_bal\_NOR)

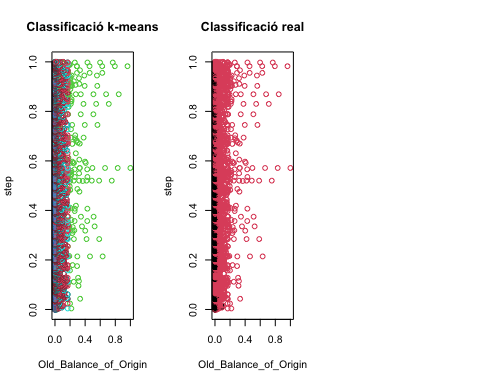
## step amount Old\_Balance\_of\_Origin New\_Balance\_of\_Origin  
## 1 0 5.985775e-06 3.037675e-06 0  
## 2 0 5.985775e-06 3.037675e-06 0  
## 3 0 9.279604e-05 4.709236e-05 0  
## 4 0 9.279604e-05 4.709236e-05 0  
## 5 0 6.656446e-04 3.378029e-04 0  
## 6 0 6.656446e-04 3.378029e-04 0  
## Old\_Balance\_of\_Destination isZero\_Old\_Balance\_of\_Or isZero\_New\_Balance\_of\_Or  
## 1 0.000000e+00 0 1  
## 2 8.966665e-05 0 1  
## 3 0.000000e+00 0 1  
## 4 1.109171e-04 0 1  
## 5 0.000000e+00 0 1  
## 6 2.653341e-05 0 1  
## isZero\_Old\_Balance\_of\_Destina isZero\_New\_Balance\_of\_Destina  
## 1 1 1  
## 2 0 1  
## 3 1 1  
## 4 0 1  
## 5 1 1  
## 6 0 0

# Old balance of destination y new balance of origin  
old.par <- par(mfrow=c(1, 2))  
plot(transacciones\_bal\_NOR[c(5,4)], col=transacciones5$cluster, main="Classificació k-means")  
plot(transacciones\_bal\_NOR[c(5,4)], col=as.factor(transacciones\_bal$isFraud), main="Classificació real")



par(old.par)

# Step y old balance of origin  
old.par <- par(mfrow=c(1, 3))  
plot(transacciones\_bal\_NOR[c(3,1)], col=transacciones5$cluster, main="Classificació k-means")  
plot(transacciones\_bal\_NOR[c(3,1)], col=as.factor(transacciones\_bal$isFraud), main="Classificació real")  
par(old.par)



A continuación mostramos la matriz de confusión para cada cluster.

d=table(transacciones5$cluster,transacciones\_bal$isFraud)  
d

##   
## No\_Fraud Fraud  
## 1 9 3919  
## 2 4772 2836  
## 3 279 158  
## 4 5812 35  
## 5 1447 1265

Los resultados de la matriz nos dicen que el cluster 1 la gran mayoria de las transaccions a classificado como No\_Fraud. Este compartoamiento es parecido en los clusters 2 y 5. En el cluster 3, las transacciones clasifican como No\_Fraud y Fraud casi igual y en el cluster 4 la mayoría clasifica como Fraud.

A partir de la matriz de confusión extraeremos el número de casos correctamente clasificados y el porcentaje de precisión del modelo.

cat("Total de casos correctamente clasificados: 2803+25+3896+5341+1167 =",  
 2803+25+3896+5341+1167 , "\n")

## Total de casos correctamente clasificados: 2803+25+3896+5341+1167 = 13232

cat("Total de casos incorrectamente clasificados: 10+2968+2835+1485+2=",  
 10+2968+2835+1485+2, "\n")

## Total de casos incorrectamente clasificados: 10+2968+2835+1485+2= 7300

cat("Precisión del modelo: (13232/(13232+7300))\*100= ",  
 (13232/(13232+7300))\*100, "%\n")

## Precisión del modelo: (13232/(13232+7300))\*100= 64.44574 %

Aplicando una métrica de distancia euclídea para calcular el número de clusters, se ha obtenido una precisión del modelo del 64.4% (k=5), por lo que se puede afirmar que esta métrica no es muy óptima para aplicar sobre nuestro juego de datos .

### 5.2.2 Distancia de manhattan

La segunda métrica seleccionada para aplicar el modelo no supervisado es con la distancia de manhattan que consiste en calcular las diferencias absolutas entre coordenadas de par de objetos. La **distancia de gauss** hemos decidido no usarla ya que esta distancia no tiene en cuenta la correlación entre variables. Además también tenemos la **distancia de mahalanobis** es una métrica de distancia multivariante eficaz que mide la distancia entre un punto y una distribución.

Hemos escogido la distancia de manhattan para ver la diferencia con la distancia euclídea. En teoría, la distancia de manhttan funcionaría bien con datos con una dimensionalidad grande, por tanto queremos ver la poca eficacia de este modelo dado que nuestro juego de datos tiene una dimensionalidad pequeña.

Para averiguar el número de clusters utilizaremos el algoritmo aglomerativo jerárquico. En este método, cada observación se asigna a nuestro cluster. Después, se calcula la similitud (o distancia) entre cada uno de los clusters y los dos clusters más similares se fusionan en uno.

Cada observación comienza en su propio cluster y los clusters se fusionan sucesivamente. Los criterios de enlace determinan la métrica utilizada para la estrategia de combinación:

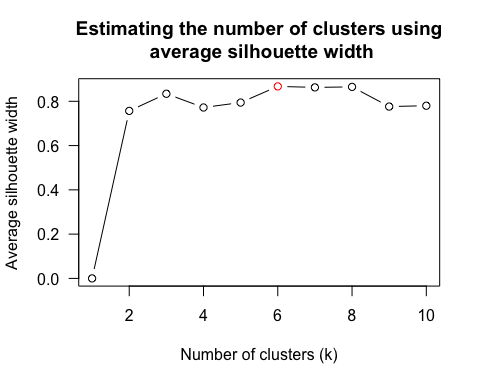
Ward minimiza la suma de las diferencias al cuadrado dentro de todos los clusters. Es un enfoque que minimiza la varianza y, en este sentido, es similar a la función objetivo k-means, pero abordado con un enfoque jerárquico aglomerativo.

Para poder aplicar este algoritmo deberemos escalar los datos y posteriormente mostrar el dendograma para saber dónde cortarlo y poder escoger el número de clusters.

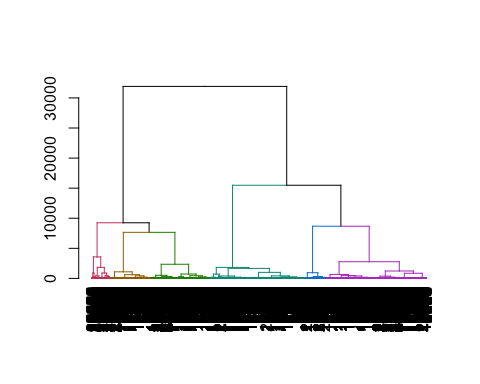
# Escalamos los datos  
transacciones\_scale <- scale(transacciones\_bal\_NOR)  
head(transacciones\_scale)

## step amount Old\_Balance\_of\_Origin New\_Balance\_of\_Origin  
## [1,] -1.568334 -0.4395164 -0.2878658 -0.06881908  
## [2,] -1.568334 -0.4395164 -0.2878658 -0.06881908  
## [3,] -1.568334 -0.4380312 -0.2867628 -0.06881908  
## [4,] -1.568334 -0.4380312 -0.2867628 -0.06881908  
## [5,] -1.568334 -0.4282307 -0.2794837 -0.06881908  
## [6,] -1.568334 -0.4282307 -0.2794837 -0.06881908  
## Old\_Balance\_of\_Destination isZero\_Old\_Balance\_of\_Or  
## [1,] -0.3263969 -0.6314376  
## [2,] -0.3208965 -0.6314376  
## [3,] -0.3263969 -0.6314376  
## [4,] -0.3195930 -0.6314376  
## [5,] -0.3263969 -0.6314376  
## [6,] -0.3247693 -0.6314376  
## isZero\_New\_Balance\_of\_Or isZero\_Old\_Balance\_of\_Destina  
## [1,] 0.2683201 1.3777772  
## [2,] 0.2683201 -0.7257714  
## [3,] 0.2683201 1.3777772  
## [4,] 0.2683201 -0.7257714  
## [5,] 0.2683201 1.3777772  
## [6,] 0.2683201 -0.7257714  
## isZero\_New\_Balance\_of\_Destina  
## [1,] 1.9888782  
## [2,] 1.9888782  
## [3,] 1.9888782  
## [4,] 1.9888782  
## [5,] 1.9888782  
## [6,] -0.5027715

#Utilizaremos el dataset escalado previamente   
#Calculamos la distancia manhattan del fecha set  
d\_manh <- dist(transacciones\_scale, method = 'manhattan')  
  
#Utilizaremos la variable d\_manh  
hc<- hclust(d\_manh, method = "ward")  
library(dendextend)  
dend <- as.dendrogram(hc)  
dend\_k<- find\_k(dend)  
plot(dend\_k)



plot(color\_branches(dend, k = dend\_k$nc))



Obtenemos que el nombre de clusters será:

groups <- cutree(hc, k = 3)  
  
table(groups, transacciones\_bal$isFraud)

##   
## groups No\_Fraud Fraud  
## 1 1473 5776  
## 2 4994 2376  
## 3 5852 61

A partir de la matriz de confusión extraeremos el número de casos correctamente clasificados y el porcentaje de precisión del modelo.

cat("Total de casos correctamente clasificados: 1536+2479+5673 =",  
 1536+2479+5673, "\n")

## Total de casos correctamente clasificados: 1536+2479+5673 = 9688

cat("Total de casos incorrectamente clasificados: 5706+5110+25=",  
 5706+5110+25, "\n")

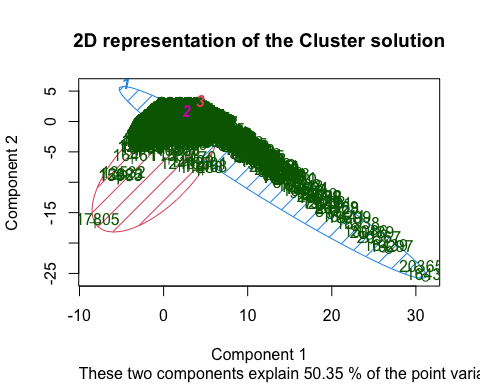
## Total de casos incorrectamente clasificados: 5706+5110+25= 10841

cat("Precisión del modelo: (4912/(1551+4912))\*100= ",  
 (9688/(9688+10841))\*100, "%\n")

## Precisión del modelo: (4912/(1551+4912))\*100= 47.19178 %

Para visualizar el clúster podemos utilizar la función clusplot. Miremos la agrupación con 3 clusters:

clusplot(transacciones\_scale, groups, main='2D representation of the Cluster solution',  
 color=TRUE, shade=TRUE,  
 labels=2, lines=0)



Como puede apreciarse a simple vista en la tabla, después de aplicar la métrica de manhattan a través del algoritmo de aglomeración se ha obtenido una precisión del 48%, la cual ha realizado una agrupación correcta pero inferior a la vista cuando se ha aplicado la distancia euclídea sobre el algoritmo de k-means. Esto se debe a que nuestro juego de datos presenta un mejor comportamiento cuando se calculan las distancias aplicando el cuadrado de la diferencia, donde permite realizar un agrupamiento más exacto. Por tanto, en este caso, nos quedaríamos con la primera métrica aplicada para realizar una correcta clasificación de las diferentes transacciones.

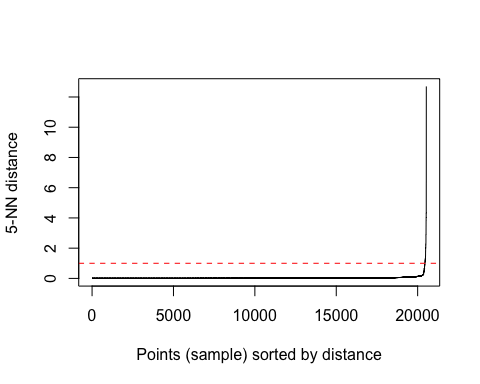
A continuación trabajaremos el algoritmo **DBSCAN** como métodos de clustering que permiten la generación de grupos no radiales a diferencia de k-means. Veremos que su parámetro de entrada más relevante es **minPts** que define la mínima densidad aceptada en torno a un centroide.

### 5.2.3 DBSCAN

El método **DBSCAN** es un algoritmo de agrupación basado en la densidad que forma grupos de regiones densas de puntos de datos ignorando las áreas de baja densidad (considerándolas como ruido).

Por este modelo tomamos las 3 variables que tienen mejor correlación y que extraímos en los apartados anteriores.

if (!require('dbscan')) install.packages('dbscan'); library('dbscan')  
  
n = c(1,4,5)  
transaccionesScan= transacciones\_bal %>% select(all\_of(n))  
transacciones\_scan\_nor <- as.data.frame(lapply(transaccionesScan, nor))  
transacciones\_scan\_scale <- scale(transacciones\_scan\_nor)  
kNNdistplot(transacciones\_scan\_scale, k = 5)  
abline(h = 1, col = "red", lty = 2)

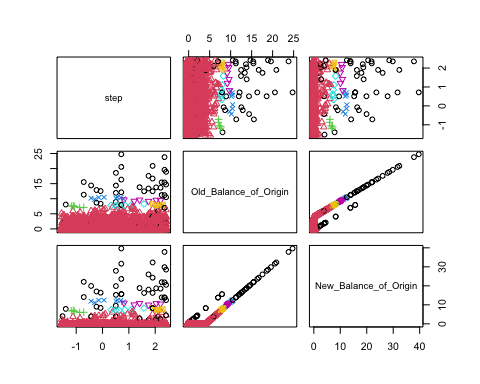
 La gráfica nos dice que el codo es visible en torno a una 5-NN distancia de 1.

res\_1<- dbscan(transacciones\_scan\_scale, eps = 1 , minPts = 5)  
res\_1

## DBSCAN clustering for 20532 objects.  
## Parameters: eps = 1, minPts = 5  
## The clustering contains 6 cluster(s) and 49 noise points.  
##   
## 0 1 2 3 4 5 6   
## 49 20454 5 8 6 6 4   
##   
## Available fields: cluster, eps, minPts

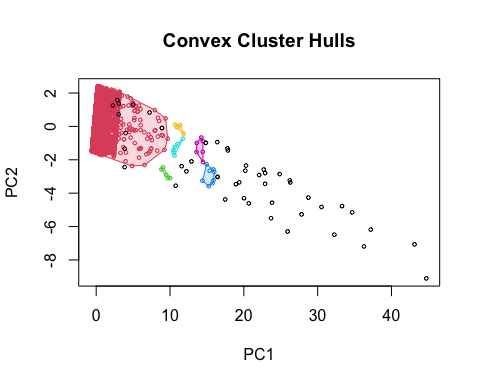
Hemos obtenido 6 clusters y 49 outliers que se encuentran en el cluster 0. Ahora mostraremos los clusters con los puntos ruidosos.

plot(as.data.frame(transacciones\_scan\_scale), col = res\_1$cluster + 1L, pch = res\_1$cluster + 1L)



Seguimos adelante con una representación gráfica que nos muestra los clusters mediante formas convexas.

hullplot(transacciones\_scan\_scale, res\_1)

 Vemos que los clusters están bien identificados y separados. Ahora mostraremos la matriz de confusión para explicar el resultado de la agrupación.

d0=table(res\_1$cluster,transacciones\_bal$isFraud)  
d0

##   
## No\_Fraud Fraud  
## 0 1 48  
## 1 12318 8136  
## 2 0 5  
## 3 0 8  
## 4 0 6  
## 5 0 6  
## 6 0 4

En el cluster 0 tenemos los 49 outliers, es decir, casos que no están asignados a ningún cluster. Tenemos 1 cluster identificado, en el primer cluster tenemos 12318 transacciones no fraudulentas y 8136 fraudulentas. En los otros 2-6 clusters las transacciones sólo clasifican como fraudes.

A partir de la matriz extraeremos el número de casos correctamente clasificados y el porcentaje de precisión.

cat("Total de casos correctamente classificados: 12318 =",  
 12318, "\n")

## Total de casos correctamente classificados: 12318 = 12318

cat("Total de casos incorrectamente clasificados: 8136+5+8+6+6+4=",  
 8136+5+8+6+6+4, "\n")

## Total de casos incorrectamente clasificados: 8136+5+8+6+6+4= 8165

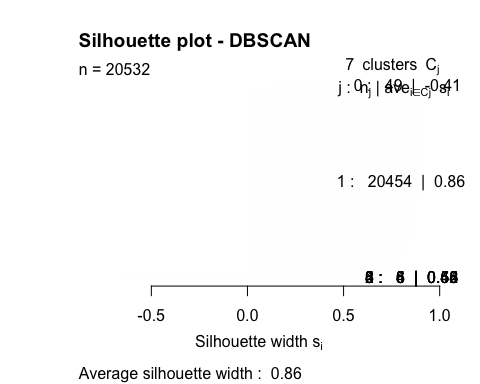
cat("Precisión: (14844/(4844+1619))\*100= ",  
 ( 12318/(12318+8165))\*100, "%\n")

## Precisión: (14844/(4844+1619))\*100= 60.13768 %

El **coeficiente silhouette** se calcula utilizando la distancia media del cluster entre puntos y la distancia media del cluster más cercano. Por ejemplo, un clúster con muchos puntos de datos muy cerca unos de otros (alta densidad) y que está lejos del siguiente clúster más cercano, tendrá un coeficiente de silhouette cercano a 1.

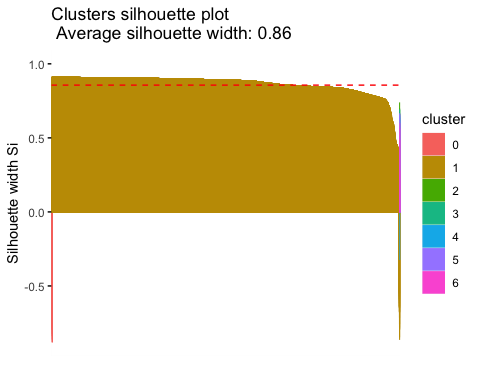
Por las gráficas de abajo vemos que el cluster 0 como era obvio tiene coeficientes negativos ya que se tratan de los outliers y por lo general los coeficientes de los otros clusters han obtenido buenos coeficientes.

aux = daisy(transacciones\_scan\_scale)  
sil = silhouette(res\_1$cluster, aux)  
plot(sil, main = "Silhouette plot - DBSCAN")



fviz\_silhouette(sil)

## cluster size ave.sil.width  
## 0 0 49 -0.41  
## 1 1 20454 0.86  
## 2 2 5 0.66  
## 3 3 8 0.54  
## 4 4 6 0.42  
## 5 5 6 0.55  
## 6 6 4 0.48

 Vemos que obtenido un coeficiente de 0.86.

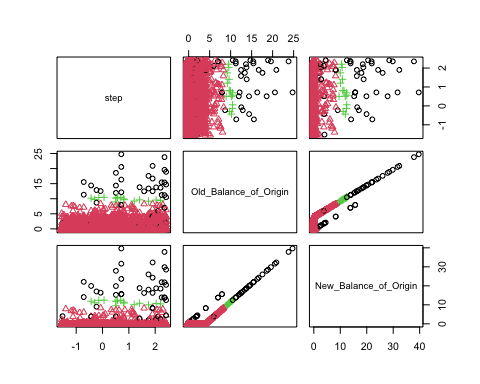
**Que pasaría si incrementamos el parametro eps ?**

res\_2<- dbscan(transacciones\_scan\_scale, eps = 1.2, minPts = 5)  
res\_2

## DBSCAN clustering for 20532 objects.  
## Parameters: eps = 1.2, minPts = 5  
## The clustering contains 2 cluster(s) and 46 noise points.  
##   
## 0 1 2   
## 46 20471 15   
##   
## Available fields: cluster, eps, minPts

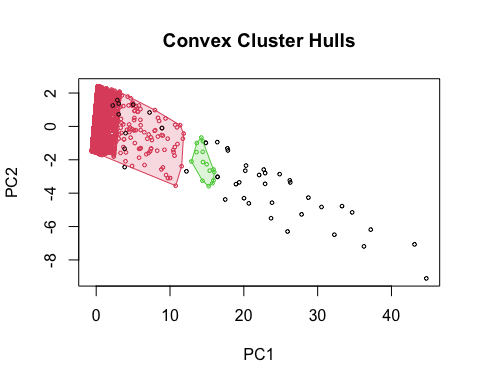
Hemos obtenido 2 clusters y 45 outliers que se encuentran en el cluster 0. Ahora mostramos los clusters con los puntos ruidososo.

plot(as.data.frame(transacciones\_scan\_scale), col = res\_2$cluster + 1L, pch = res\_2$cluster + 1L)



Seguimos adelante con una representación gráfica que nos muestra los clusters mediante formas convexas.

hullplot(transacciones\_scan\_scale, res\_2)



Vemos que los clusters están bien identificados y separados. Ahora mostraremos la matriz de confusión para explicar el resultado de la agrupación.

d1=table(res\_2$cluster,transacciones\_bal$isFraud)  
d1

##   
## No\_Fraud Fraud  
## 0 1 45  
## 1 12318 8153  
## 2 0 15

En el cluster 0 tenemos los 45 outliers, es decir, casos que no están asignados a ningún cluster. Tenemos 1-2 clusters identificados, en el primer cluster tenemos 12319 transacciones clasificados como No Fraude y 8153 como Fraude. En el otro clusters sólo clasifica como Fraude 15 transacciones.

A partir de la matriz extraeremos el número de casos correctamente clasificados y el porcentaje de precisión.

cat("Total de casos correctamente clasificados: 12319+15 =",  
 12319+15, "\n")

## Total de casos correctamente clasificados: 12319+15 = 12334

cat("Total de casos incorrectamentee clasificados: 8153=",  
 8153, "\n")

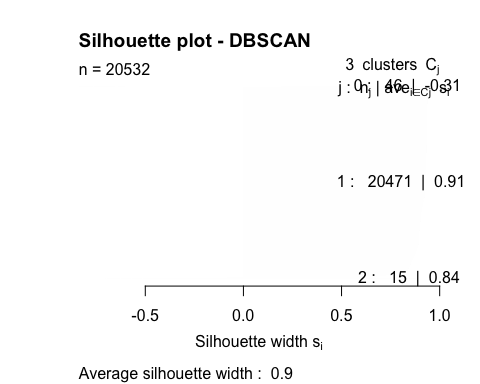
## Total de casos incorrectamentee clasificados: 8153= 8153

cat("Precissió: (14844/(4844+1619))\*100= ",  
 ( 12334/(12334+8153))\*100, "%\n")

## Precissió: (14844/(4844+1619))\*100= 60.20403 %

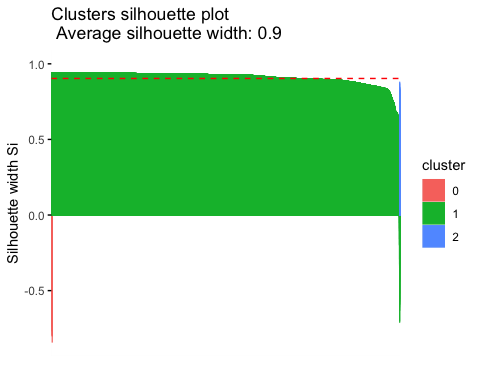
Por las gráficas de abajo vemos que el cluster 0 como era obvio tiene coeficientes negativos ya que se tratan de los outliers y por lo general los coeficientes de los otros clusters han obtenido buenos coeficientes.

aux = daisy(transacciones\_scan\_scale)  
sil = silhouette(res\_2$cluster, aux)  
plot(sil, main = "Silhouette plot - DBSCAN")



fviz\_silhouette(sil)

## cluster size ave.sil.width  
## 0 0 46 -0.31  
## 1 1 20471 0.91  
## 2 2 15 0.84



Vemos que el coeficiente de silhouete por un eps = 1.2 no mejora y es casi el mismo.

Tenemos pues que con el algoritmo DBSCAN por diferentes eps, obteniendo 2 y 6 clusters, siendo la precisión la misma. El porcentaje está alrededor de 60% y es mejor que el obtenido con k means con distancia euclídea que está por encima del 64% y peor al modelo obtenido con el algoritmo de aglomeración y distancia de manhattan con 47%.

#6 Limitaciones de datos en modelos supervisado y no supervisados

**El aprendizaje supervisado**  es un enfoque de aprendizaje automático que se define mediante el uso de conjuntos de **datos etiquetados**. Estos conjuntos de datos están diseñados para entrenar o “supervisar” algoritmos para clasificar los datos o predecir los resultados con precisión. Usando entradas y salidas etiquetadas, el modelo puede medir su precisión y aprender con el tiempo.

El aprendizaje supervisado se puede separar en dos tipos de problemas en la minería de datos: *clasificación y regresión*

**El aprendizaje no supervisado** utiliza algoritmos de aprendizaje automático para analizar y agrupar conjuntos de **datos sin etiquetar**. Estos algoritmos descubren patrones ocultos en los datos sin necesidad de intervención humana (por tanto, están “sin supervisar”).

Los modelos de aprendizaje no supervisado se utilizan para tres tareas principales: *agrupación, asociación y reducción de la dimensionalidad.*

La principal diferencia es el uso de conjuntos de datos. Por decirlo simplemente, el aprendizaje supervisado utiliza datos de entrada y salida etiquetados, mientras que un algoritmo de aprendizaje no supervisado no.

En nuestro caso tenemos un conjunto de datos referente a las transacciones bancarias. Por cada row es un data point , es decir, cada fila representa una transaccion diferente que está definido por el conjunto de características como el balance de origen, la cantidad, el balance de destino, etc. Las características son las columnas del juego de datos. Para hablar de etiquetas, deberemos contextualizar nuestro problema. Si predecimos una feature o columna basada en las demás, entonces la etiqueta es el feature o columna. Si predecimos si nuestra transacción es fraude o no (en nuestro caso), basándonos en la información de la transacción, entonces ésta es la etiqueta.

Por tanto, nuestro juego de datos esta etiquetado ya que queremos predecir si la transacción es fraudulenta o no a partir del conjunto de features. Por tanto nuestro juego de datos está orientado al aprendizaje supervisado. Imaginemos que tenemos un nueva transacción es decir no sabemos si es fraudulenta o no, el modelo entonces haría una predicción si esta transacción es fraudulenta o no.

Supongamos que nuestro data set no está etiquetado, entonces nuestro modelo todavía podría decirnos si dos data points son similares entre ellos o differentes. Los podría agrupar por similitud aunque no sabemos que representa a cada grupo.

En nuestro data set no hay demasiadas limitaciones. Por un lado, la dimensionalidad es pequeña con 11 variables y alrededor de 6 millones de filas.

Estos son los **retos a los que se enfrenta el aprendizaje automático supervisado**:

* Los datos de entrenamiento irrelevantes de la función de entrada pueden dar resultados inexactos
* La preparación y el preprocesamiento de datos es siempre un reto
* La precisión sufre cuando se han introducido valores imposibles, improbables e incompletos como datos de entrenamiento
* Si el experto en cuestión no está disponible, el otro enfoque es “fuerza bruta”. Significa que debemos pensar las featuras adecuadas (variables de entrada) para entrenar la máquina. Puede ser inexacto.
* Clasificar gran cantidad de datos puede ser un gran reto
* Entrenar el modelo supervisado necesita tiempo computacional

Estos son los **retos a los que se enfrenta el aprendizaje automático no supervisado**:

* No podemos obtener información precisa sobre la ordenación de datos, y la salida como datos utilizados en el aprendizaje no supervisado está etiquetada y no se conoce.
* Menos precisión de los resultados se debe a que los datos de entrada no son conocidas y no etiquetados por la gente de antemano. Esto significa que la máquina debe hacerlo ella misma.
* Las clases no siempre corresponden a clases informativas.
* El usuario debe dedicar tiempo a interpretar y etiquetar las clases que siguen esta clasificación.