Preproceso de indemnizaciones otorgadas

Mohammed Makhfi Blulaich

9 de octubre, 2021

Introducción

El presente documento detalla el preprocesado de un fichero de datos que contiene información de una muestra de indemnizaciones otorgadas por una compañía de seguros, en función del tiempo de baja laboral del trabajador.

El objetivo de este proceso es preparar el fichero para su posterior análisis. Para ello, se examinará el fichero para detectar y corregir posibles errores, inconsistencias y valores perdidos. Además, se presentará una breve estadística descriptiva.

La preparación de los datos cada vez cobra más importancia, dado que la base de cualquier estudio estadístico es la información recogida, es decir, los datos disponibles para realizar el modelo y obtener las conclusiones.

La preparación de los datos se puede dividir en:

1. **Lectura de los datos.** Pasar la información bruta recogida y cargarla en el software estadístico.
2. **Limpieza de los datos.** Corregir errores, inconsistencias y otros problemas que pueden aparecer en los datos.
3. **Reducción de los datos.** En ocasiones, el exceso de información es contraproducente, y se requiere aplicar técnicas de optimización, eliminando datos que *a priori* no son útiles para el análisis.

En un proyecto de ciencias de datos, esta etapa ocupa el 80% del tiempo de trabajo, mientras que el 20% restante se divide en el desarrollo del modelo, la evaluación y la optimización del mismo.

1. Carga del archivo

El archivo a leer es un fichero de texto cuyo delimitador es el *punto y coma*, los campos están en formato tabular y el separador decimal es el *punto*:

```
data <- read.csv("train3.csv", header=TRUE, sep=";", dec=".")
```

Examinamos algunas variables utilizando la función *summary*, que nos resume el tipo de datos con el que R ha interpretado cada variable y nos muestra algunas medidas que nos permiten evaluar rápidamente la dispersión y la tendencia central de un conjunto de datos, que son los pasos iniciales importantes para comprender sus datos:

```
summary(data[c("ClaimNumber","DateTimeOfAccident","Gender","Age")])
```

```
##  ClaimNumber        DateTimeOfAccident    Gender              Age
##  Length:54000       Length:54000       Length:54000       Min.   : 13.00
##  Class :character   Class :character   Class :character   1st Qu.: 23.00
##  Mode  :character   Mode  :character   Mode  :character   Median : 32.00
##                                                           Mean   : 34.06
##                                                           3rd Qu.: 43.00
##                                                           Max.   :999.00
```

2. Duplicación de códigos

Verificamos la consistencia de la variable ClaimNumber y observamos que la clave primaria, que sirve para identificar de forma exclusiva un registro en la tabla, no es clave única:

```
# valores duplicados
data$ClaimNumber[duplicated(data$ClaimNumber) == TRUE]
```

```
## [1] "WC8668542" "WC8668542" "WC3501716" "WC6383678"
```

Por lo que procederemos a subsanarlo. Para ello, copiamos la clave primaria a una columna auxiliar, llevándonos solo los valores numéricos, los 7 caracteres de la derecha:

```
substrRight <- function(x, n){
  substr(x, nchar(x)-n+1, nchar(x))
}

data$ClaimNumberId <- substrRight(data$ClaimNumber, 7)
data$ClaimNumberId <- as.integer(data$ClaimNumberId)
```

Aquellos valores duplicados los sustituimos por valores nuevos partiendo del número siguiente al máximo identificado en orden:

```
maxClaimNumberId <- summary(data$ClaimNumberId)[["Max."]]
nduplicated <- length(data$ClaimNumber[duplicated(data$ClaimNumber) == TRUE])
for (i in 1:nduplicated) {
  data$ClaimNumberId[duplicated(data$ClaimNumberId) == TRUE][1] <- maxClaimNumberId + i
}
```

Y los incorporamos de vuelta a la variable original:

```
# pegar 'WC' de vuelta al número como paso previo a incorporarlo a la variable original
data$ClaimNumber[duplicated(data$ClaimNumber) == TRUE] <- paste(
  "WC",
  data$ClaimNumberId[duplicated(data$ClaimNumber) == TRUE],
  sep=""
)
# borramos la variable auxiliar
data <- subset(data, select = -c(ClaimNumberId))
```

3. Nombres de las variables

Simplificamos el nombre de algunas variables para hacer más fácil su manejo:

```
names(data)[names(data) == "InitialIncurredCalimsCost"] <- "IniCost" #typo in colname
names(data)[names(data) == "UltimateIncurredClaimCost"] <- "UltCost"
names(data)[names(data) == "HoursWorkedPerWeek"] <- "HoursWeek"
names(data)[names(data) == "DaysWorkedPerWeek"] <- "DaysWeek"
```

4. Normalización de los datos cualitativos

4.1. Marital Status

Los valores posibles de MaritalStatus son: M (married), S (single), U (unknown), D (divorced), W (widowed). Pero observamos que los valores de la variable no son consistentes:

```
unique(data$MaritalStatus)
```

```
##  [1] "M"       "m"       "U"       "S"       "married" "W"       ""
##  [8] "d"       "D"       "w"
```

Por un lado, unificamos el formato de los valores, capitalizando el texto y realizando las sustituciones oportunas. Y, por otro, informamos los valores vacíos con la U, dando a entender que son valores desconocidos:

```
# unificamos el formato de los valores, capitalizando el texto y sustituyendo MARRIED por M
data$MaritalStatus <- toupper(data$MaritalStatus)
data["MaritalStatus"][data["MaritalStatus"] == "MARRIED"] <- "M"
# los valores vacíos los informamos con la U, dando a entender que son desconocidos
data["MaritalStatus"][data["MaritalStatus"] == ""] <- "U"
# convertimos la variable a tipo factor, indicando que es una variable categórica
data$MaritalStatus <- factor(data$MaritalStatus)
levels(data$MaritalStatus)
```

```
## [1] "D" "M" "S" "U" "W"
```

4.2. Gender

Los valores posibles de Gender son: F (femenino), M (masculino), U (unknown). Pero observamos que los valores de la variable no son consistentes:

```
unique(data$Gender)
```

```
## [1] "M"  "F"  "Fm" "f"  "U"
```

Por lo que, al igual que hicimos en el apartado anterior, unificamos el formato de la variable:

```
# sustituir por F todo valor que comience por una F, sea mayúscula o minúscula
data$Gender[grepl("[F|f].*", data$Gender)] <- "F"
# convertimos la variable a tipo factor, indicando que es una variable categórica
data$Gender <- factor(data$Gender)
levels(data$Gender)
```

```
## [1] "F" "M" "U"
```

5. Normalización de los datos cuantitativos

5.1. IniCost y UltCost

La variable IniCost no parece que requiera alguna transformación:

```
# el resumen nos adelanta que la variable IniCost es entera (sin decimales)
summary(data$IniCost)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     700    2000    7841    9500 2000000
```

```
class(data$IniCost)
```

```
## [1] "integer"
```

Mientras que la variable UltCost sí. Como podemos observar, algunos de los valores que toma la variable están expresados en miles. A estos hay que aplicarles una transformación tal que pasen a las unidades:

```
# observamos que algunos valores están expresados en miles
head(data$UltCost[grepl(".*[a-zA-Z]", data$UltCost)])
```

```
## [1] "1.277718363K"  "0.1077643807K" "1.445204861K"  "8.60810336K"
## [5] "2.597486704K"  "2.005016752K"
```

```
# unificamos el formato expresándolos en unidades
data$UltCost[grepl(".*[a-zA-Z]", data$UltCost)] <- as.numeric(
    sub("K", "", data$UltCost[grepl(".*[a-zA-Z]", data$UltCost)], fixed = TRUE)
  )*1000
# transformamos la variable en entera (sin decimales)
data$UltCost <- as.integer(data$UltCost)
# el resumen nos sugiere que la transformación se ha realizado correctamente
summary(data$UltCost)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       5    1128    3179   10195    8900 1570535
```

5.2. Age

Transformamos Age en una variable entera:

```
data$Age <- as.integer(data$Age)
class(data$Age)
```

```
## [1] "integer"
```

5.3. WeeklyWages, HoursWeek, DaysWeek

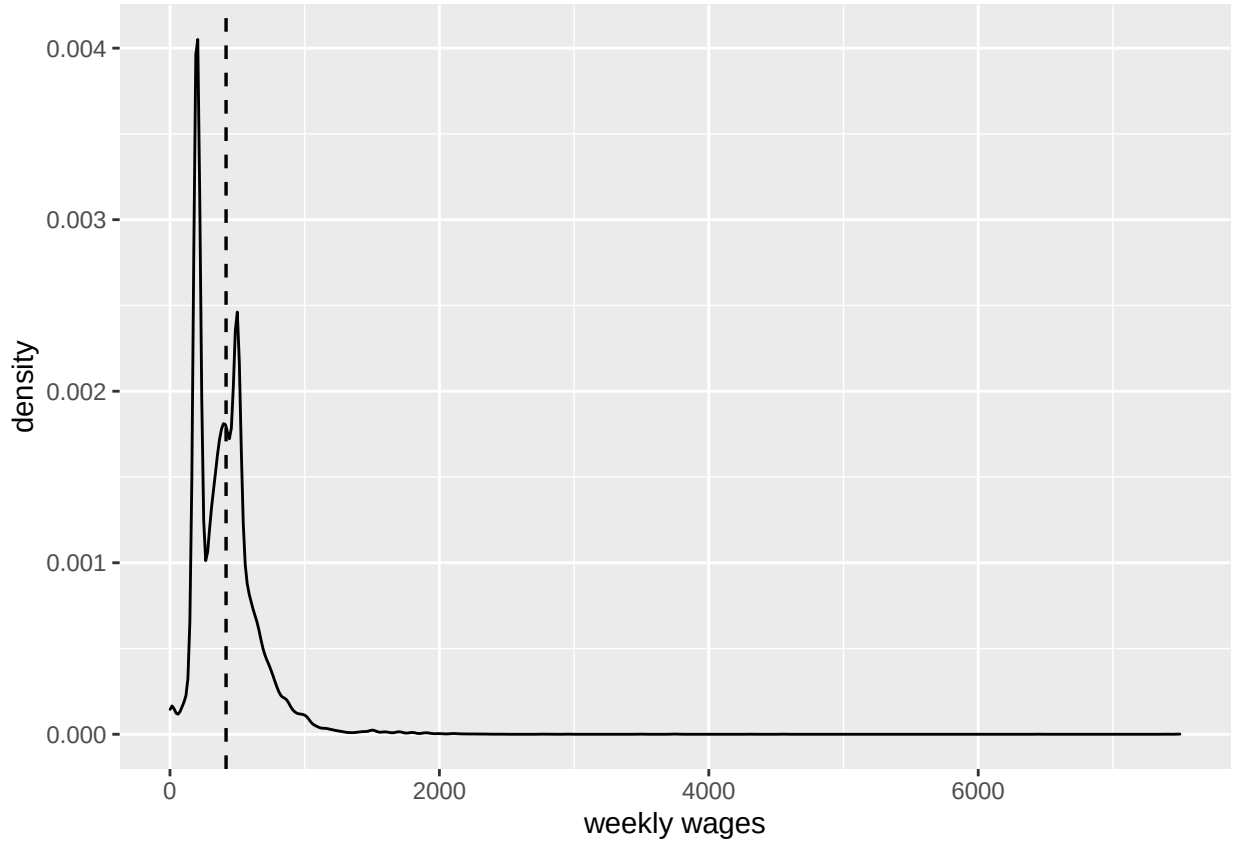
Aplicamos el formato numérico a la variable WeeklyWages: el símbolo de separador decimal es el punto y no la coma:

```
# sustituir la coma por el punto donde corresponda
data$WeeklyWages <- as.numeric(sub(",", ".", data$WeeklyWages, fixed = TRUE))
summary(data$WeeklyWages)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   200.0   392.2   416.4   500.0  7497.0
```

Visualizamos cómo está distribuida la variable mediante un gráfico de densidad:

```
# la línea vertical indica la media de la variable
ggplot(data, aes(x = WeeklyWages)) +
  geom_density() +
  geom_vline(aes(xintercept = mean(WeeklyWages)), linetype = "dashed", size = 0.6) +
  xlab("weekly wages")
```



Hacemos lo mismo con la variable HoursWeek. Primero le aplicamos el formato numérico:

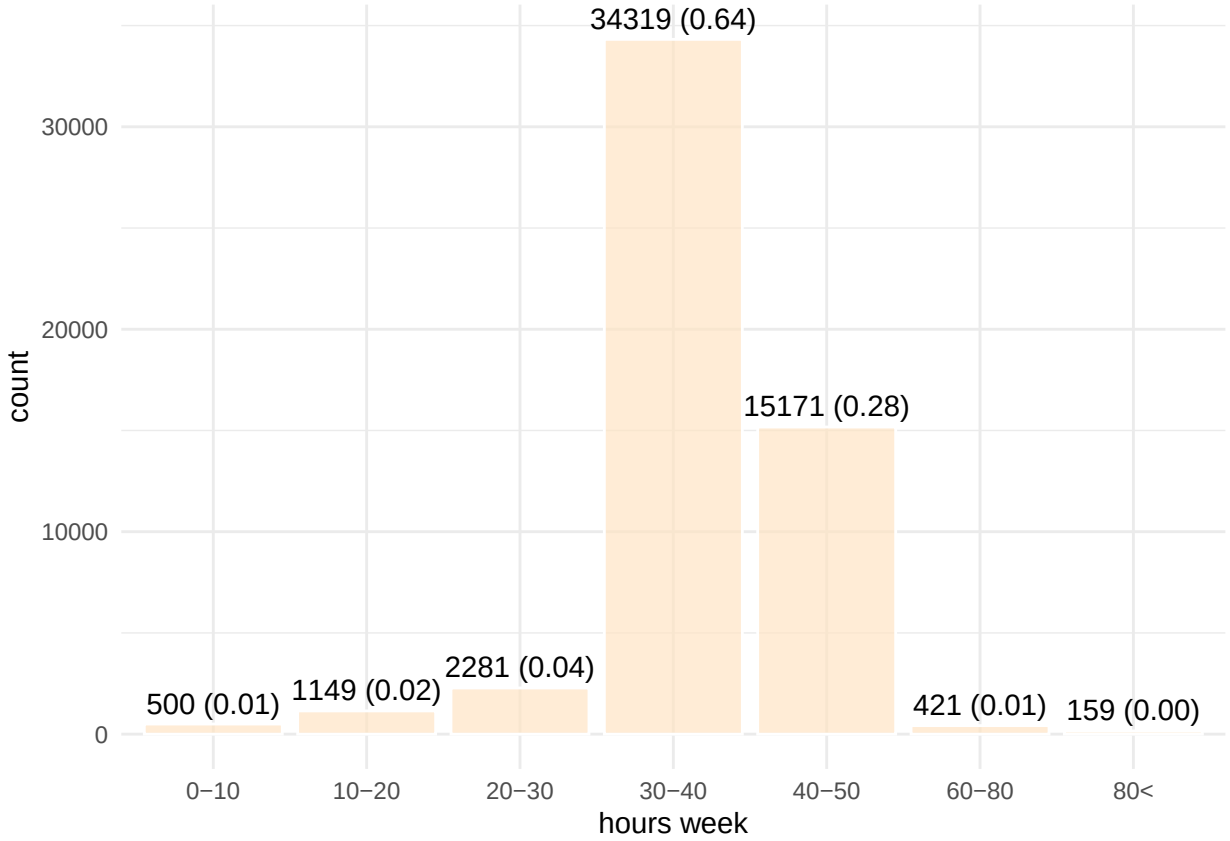
```
data$HoursWeek <- as.numeric(sub(",", ".", data$HoursWeek, fixed = TRUE))
summary(data$HoursWeek)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   38.00   38.00   37.74   40.00  640.00
```

E inmediatamente después estudiamos el patrón que sigue la variable, mediante el agrupamiento de datos (data binning):

```
# necesitamos tener la variable ordenada
HoursWeek <- sort(data$HoursWeek)
# especificamos los cortes que queremos realizar
breaks <- c(0,10,20,30,40,60,80,max(HoursWeek))
# especificamos las etiquetas de los intervalos
tags <- c("0-10","10-20", "20-30", "30-40", "40-50", "60-80","80<")
# agrupamos los valores bajo sus correspondientes intervalos
group_tags <- cut(HoursWeek,
                  breaks=breaks,
                  include.lowest=TRUE,
                  right=FALSE,
                  labels=tags)
# transformamos la variable en una variable categórica ordinal
group_tags <- factor(group_tags, levels=tags, ordered=TRUE)

ggplot(data = as_tibble(group_tags), mapping = aes(x=value)) +
  geom_bar(fill="bisque",color="white",alpha=0.7) +
  stat_count(
    geom="text",
    aes(label=sprintf("%i (%.2f)", ..count.., ..count../length(group_tags))),
    vjust=-0.5
  ) +
  labs(x='hours week') +
  theme_minimal()
```

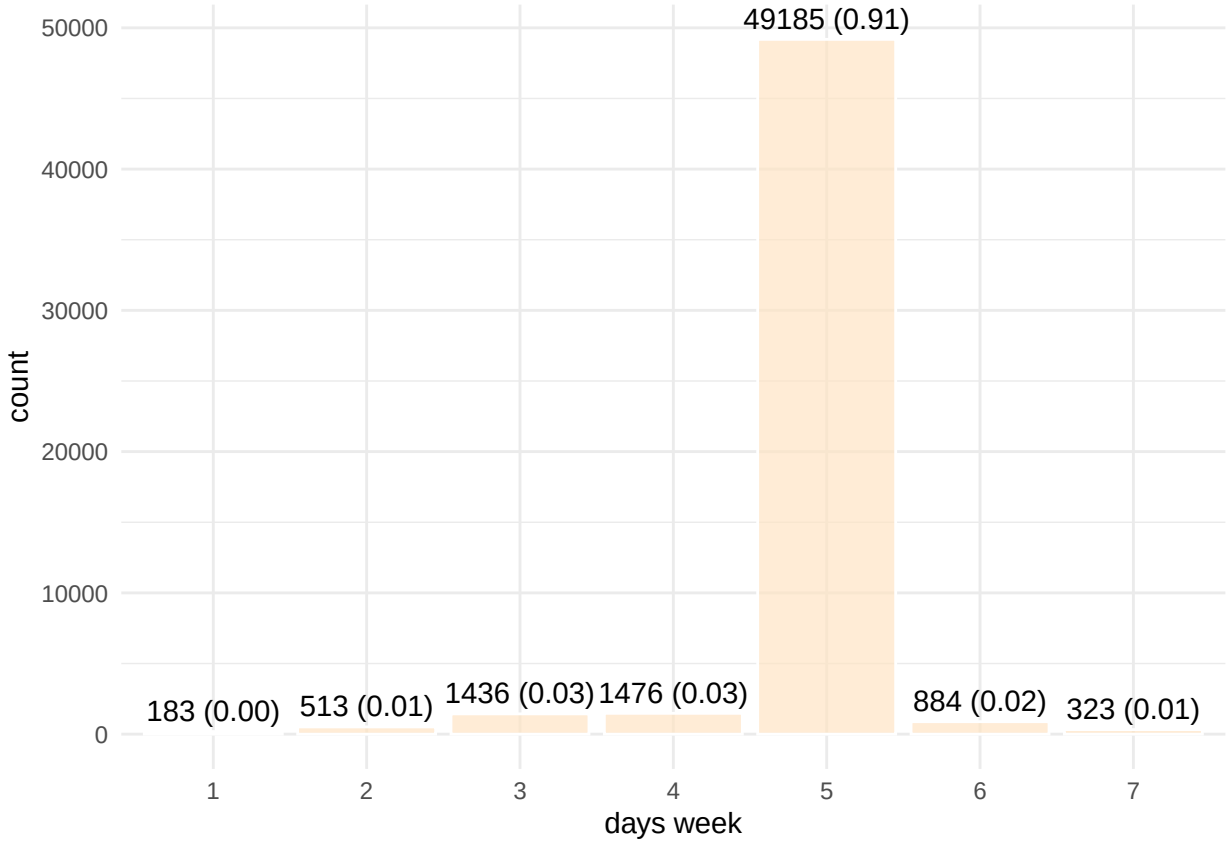


En cuanto a la variable DaysWeek, podemos decir que es una variable categórica ordinal oculta como variable cuantitativa, pudiendo tomar valores enteros entre el rango 1 y 7:

```
data$DaysWeek <- factor(data$DaysWeek)
levels(data$DaysWeek)
```

```
## [1] "1" "2" "3" "4" "5" "6" "7"
```

```
ggplot(data, mapping = aes(x=DaysWeek)) +
  geom_bar(fill="bisque",color="white",alpha=0.7) +
  stat_count(geom="text", aes(label=sprintf("%i (%.2f)", ..count.., ..count../length(data$DaysWeek))), 
  labs(x='days week') +
  theme_minimal()
```



Siendo 5 el número de días que trabaja el indemnizado en general, como era de esperar.

6. Valores atípicos

Clasificaremos los valores atípicos mediante el test de Tukey, por su sencillez y buenos resultados:

```
out_tukey <- function(x){
  q1 <- quantile(x, 0.25)
  q3 <- quantile(x, 0.75)

  iqr <- q3-q1 # rango = IQR(x)

  res <- x<(q1-1.5*iqr) | x>(q3+1.5*iqr)
  res
}
```

Listamos los valores atípicos de la variable Age y sustituimos 999 por NA:

```
# listar valores atípicos
data$Age[out_tukey(data$Age)]
```

```
##  [1]  74  74 999 999 999  74  75 999  75  79 999  74  75  74 999  74 999  76  75
## [20]  76  76  75  74  80 999  76  79 999  78  78  81 999 999 999
```

```r
# sustituimos 999 por NA
data$Age[data$Age == 999] <- NA
```

Comprobamos que la proporción de valores atípicos de WeeklyWages sea baja, para evitar perder información relevante:

```r
# proporción de valores atípicos
length(data$WeeklyWages[out_tukey((data$WeeklyWages))])/length(data$WeeklyWages)
```

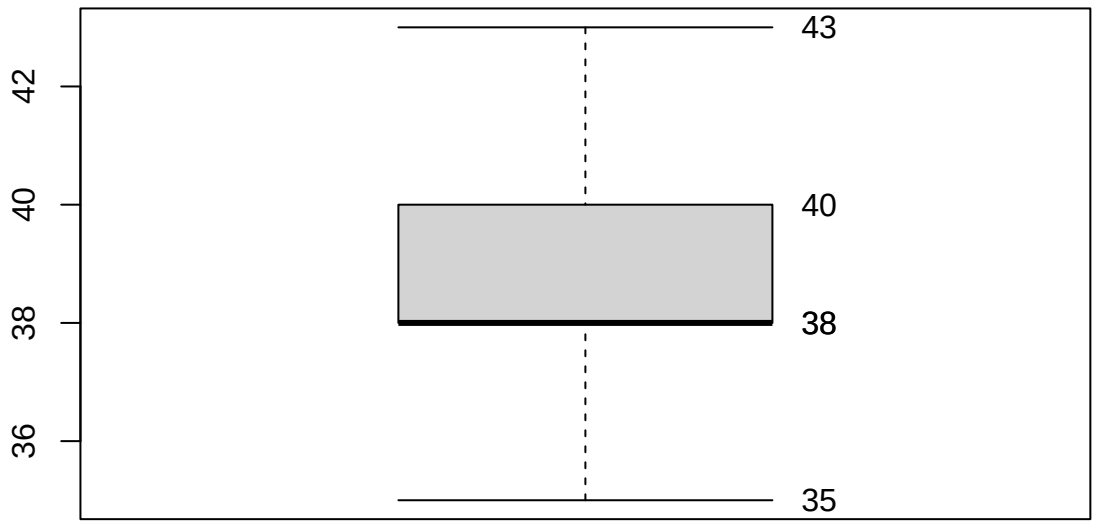
```
## [1] 0.02737037
```

Y los sustituimos por NA:

```r
# sustituir
data$WeeklyWages[out_tukey((data$WeeklyWages))] <- NA
```

Sustituimos los valores atípicos de HoursWeek por NA y visualizamos el boxplot resultante:

```r
data$HoursWeek[out_tukey(data$HoursWeek)] <- NA
boxplot(data$HoursWeek, staplewex = 1)
text(
  y=boxplot.stats(data$HoursWeek)$stats,
  labels = boxplot.stats(data$HoursWeek)$stats,
  x = 1.25
)
```



7. Imputación de valores

Listamos el nombre de las variables que tienen valores sin informar:

```
# columnas con NAs
colnames(data)[apply(data, 2, anyNA)]
```

```
## [1] "Age"         "WeeklyWages" "HoursWeek"
```

Imputamos la variable Age por la media aritmética y verificamos que se ha realizado correctamente:

```
mean_missing <- mean(data$Age, na.rm = TRUE)
# para Age, aplicamos la imputación por la media aritmética
data <- data %>%
  mutate(Age_imp  = ifelse(is.na(Age), mean_missing, Age))

head(data[is.na(data$Age),c("Age", "Age_imp")])
```

```
##       Age  Age_imp
## 5218   NA 33.84161
## 6850   NA 33.84161
## 7495   NA 33.84161
## 11184  NA 33.84161
## 17520  NA 33.84161
## 24946  NA 33.84161
```

En el resto de variables aplicaremos la imputación por vecinos más cercanos (kNN). En el cómputo de los vecinos más cercanos consideraremos Age, IniCost y UltCost. Además, la imputación se hará agrupada por género:

```
# agrupamos por género
by_gender <- data %>% group_by(Gender)
# imputamos por género teniendo en cuenta las variables cuantitativas mencionadas
by_gender_imp <- kNN(
  by_gender,
  c("WeeklyWages", "HoursWeek"),
  dist_var = c("Age", "IniCost", "UltCost")
)
# desagrupamos el resultado
data_imp <- by_gender_imp %>% ungroup()
# eliminamos las columnas que indican la imputación con TRUE o FALSE
data_imp <- subset(data_imp, select = -c(WeeklyWages_imp, HoursWeek_imp))
# renombramos las columnas para indicar que han sido imputadas
names(data_imp)[names(data_imp) == "WeeklyWages"] <- "WeeklyWages_imp"
names(data_imp)[names(data_imp) == "HoursWeek"] <- "HoursWeek_imp"
data_imp <- subset(data_imp, select =
                     c("ClaimNumber", "WeeklyWages_imp", "HoursWeek_imp"))
# añadimos las columnas imputadas al set de datos original
data <- merge(data, data_imp, by="ClaimNumber")
# verificamos que la imputación se ha realizado correctamente
sample_n(data[
  is.na(data$Age) | is.na(data$WeeklyWages) | is.na(data$HoursWeek),
```

```
  c(
    "ClaimNumber", "Age", "WeeklyWages", "HoursWeek",
    "Age_imp", "WeeklyWages_imp", "HoursWeek_imp"
    )
  ], 5)
```

```
##   ClaimNumber Age WeeklyWages HoursWeek Age_imp WeeklyWages_imp HoursWeek_imp
## 1   WC3446086  63      443.30        NA      63          443.30            38
## 2   WC9838520  44          NA        40      44          726.00            40
## 3   WC6352169  31      646.82        NA      31          646.82            38
## 4   WC9173809  53      500.00        NA      53          500.00            40
## 5   WC9234157  49      665.00        NA      49          665.00            40
```

Antes de continuar, realizamos una limpieza del entorno, borrando aquellas variables que no nos hacen falta de aquí en adelante:

```
# eliminamos las columnas con valores perdidos
data <- subset(data, select = -c(Age, WeeklyWages, HoursWeek))
# sustituimos por las columnas imputadas
names(data)[names(data) == "Age_imp"] <- "Age"
names(data)[names(data) == "WeeklyWages_imp"] <- "WeeklyWages"
names(data)[names(data) == "HoursWeek_imp"] <- "HoursWeek"
# limpiamos las variables de entorno salvando únicamente el set de datos
envars <- ls()
rm(list = envars[envars != "data"])
```

8. Preparación de los datos

8.1. Tiempo de abertura del expediente

Calculamos el tiempo que se tarda en abrir el expediente desde el suceso del accidente, a partir de las variables DateTimeOfAccident y DateReported.

```
# convertimos las variables en formato Date
data$DateReported <- as.POSIXct(
  data$DateReported,
  format="%Y-%m-%dT%H:%M:%SZ",
  tz=Sys.timezone()
)
data$DateTimeOfAccident <- as.POSIXct(
  data$DateTimeOfAccident,
  format="%Y-%m-%dT%H:%M:%SZ",
  tz=Sys.timezone()
)
# realizamos el cálculo y almacenamos el resultado en una nueva variable del conjunto de datos (Time)
data$Time <- as.numeric(difftime(data$DateReported, data$DateTimeOfAccident), units = "days")
```

8.2. Diferencia entre IniCost y UltCost

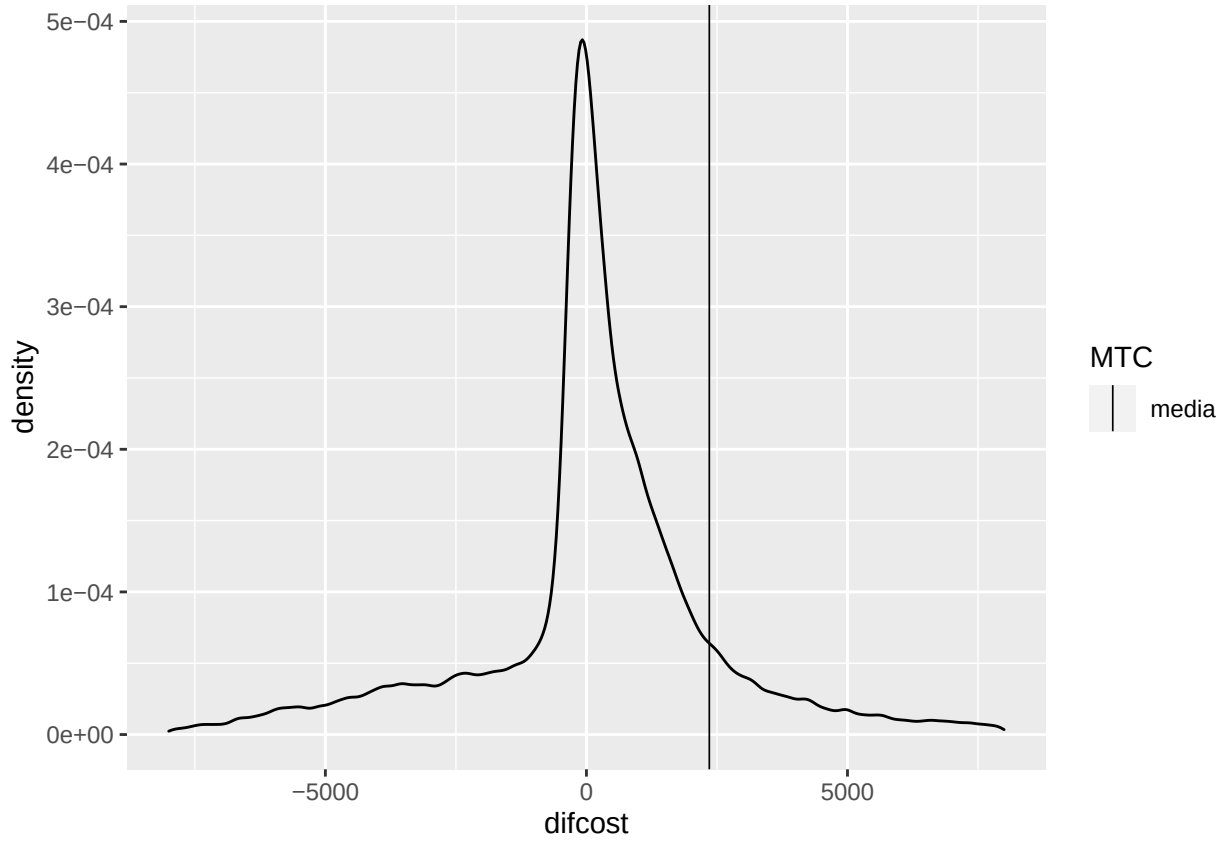
Calculamos la diferencia entre el coste final y el coste inicial estimado:

```
data$DifCost <- data$UltCost - data$IniCost
summary(data$DifCost)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1933783     -449      190     2354     1435  1527535
```

Y visualizamos cómo está distribuida la variable mediante un gráfico de densidad:

```
ggplot(data, aes(x = DifCost)) +
  geom_density() +
  geom_vline(aes(xintercept = mean(DifCost), linetype = "media"),
             size = 0.3, show.legend = TRUE) +
  scale_linetype_manual(name = "MTC", values = c("media" = "solid")) +
  scale_x_continuous(limits = c(-8000, 8000)) +
  xlab("difcost")
```



9. Estudio descriptivo

9.1. Funciones de media robustas

Para el estudio descriptivo haremos uso de las medidas de tendencia central y dispersión. A continuación las definimos:

```r
media <- function(x) {
  mean(x)
}

media.recortada <- function(x, perc=0.05) {
  mean(x, trim=perc)
}

media.winsor <- function (x, perc=0.05) {
  lim <- quantile(x, probs=c(perc, 1-perc))
  x[ x < lim[1] ] <- lim[1]
  x[ x > lim[2] ] <- lim[2]
  mean(x)
}
```

También implementamos la mediana y la desviación típica para poder referenciarlas en lengua castellana:

```r
mediana <- function(x) {
  median(x)
}

desviacion.tipica <- function(x) {
  sd(x)
}
```

La validación de estas funciones se encuentran en el script adjunto.

9.2. Estudio descriptivo de las variables cuantitativas

Definimos una función auxiliar que confecciona una tabla resumen con medidas de tendencia central y dispersión, robustas y no robustas:

```r
resumen.tcd <- function(df, cols, fns) {
  tmp.df <- data.frame(matrix(ncol = length(cols), nrow = 0))
  x <- cols
  colnames(tmp.df) <- x

  for (fun in fns) {
    tmp.row <- df %>%
      select(all_of(cols)) %>%
      summarise(across(.fns=get(fun)))
    row.names(tmp.row) <- fun
    tmp.df <- rbind(tmp.df, tmp.row)
  }

  tmp.df
}
```

Especificamos las variables objeto de estudio, las medidas que queremos analizar y realizamos la llamada a la función, obteniendo la tabla resumen en cuestión:

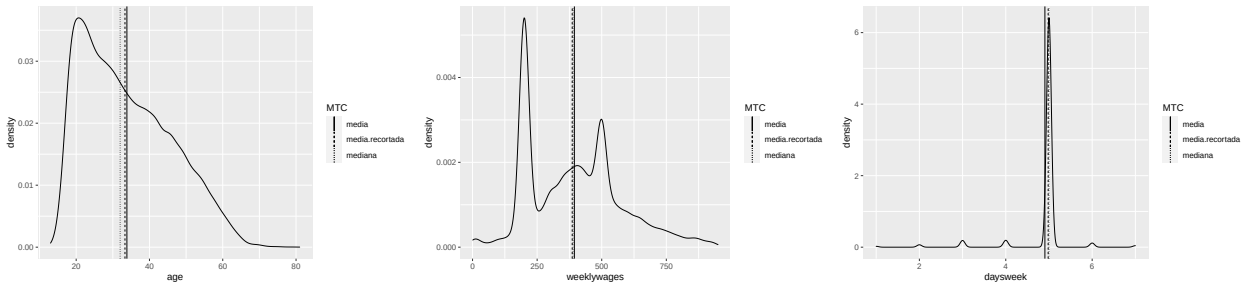
```r
# aplicamos el formato numérico para incluirla en el análisis de manera temporal
data$DaysWeek <- as.integer(data$DaysWeek)
resumen.cols <- c("Age", "WeeklyWages", "DaysWeek", "HoursWeek", "IniCost", "UltCost")
resumen.fns <- c("media", "mediana", "media.recortada", "media.winsor",
                 "desviacion.tipica", "IQR", "mad")
# muestra tabla resumen con medidas de tendencia central y dispersión, robustas y no robustas
resumen.tcd(data, resumen.cols, resumen.fns)
```

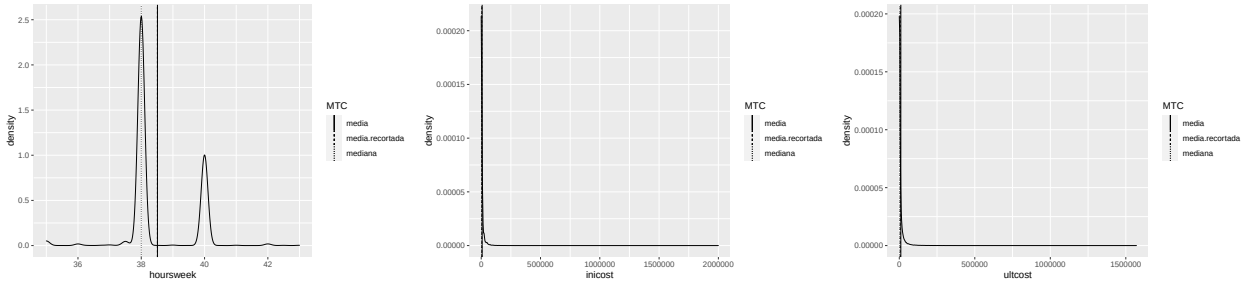
```
##                       Age WeeklyWages  DaysWeek HoursWeek   IniCost   UltCost
## media             33.84161    394.1509 4.9057593 38.511706  7841.146 10194.962
## mediana           32.00000    388.8900 5.0000000 38.000000  2000.000  3179.000
## media.recortada   33.31739    385.7556 4.9813169 38.509723  5102.874  6195.801
## media.winsor      33.68565    393.7702 4.9331852 38.558750  6108.337  7538.201
## desviacion.tipica 12.12035    182.6745 0.5521291  1.050354 20584.075 29023.515
## IQR               20.00000    300.0000 0.0000000  2.000000  8800.000  7772.000
## mad               13.34340    210.3216 0.0000000  0.000000  2223.900  3755.426
```

También disponemos de la función análoga, en formato gráfico:

```r
# resumen gráfico de las medidas de tendencia central (MTC)
resumen.grf <- function(df, cols) {
  for (col in cols) {
    print(
      ggplot(df, aes(x = get(col))) +
        geom_density() +
        geom_vline(aes(xintercept = media(get(col)), linetype="media"),
                   size = 0.3) +
        geom_vline(aes(xintercept = media.recortada(get(col)), linetype="media.recortada"),
                   size = 0.3) +
        geom_vline(aes(xintercept = mediana(get(col)), linetype="mediana"),
                   size = 0.3) +
        scale_linetype_manual(
          name = "MTC",
          values = c("media" = "solid", "media.recortada" = "dashed", "mediana" = "dotted")
        ) +
        labs(x=tolower(col))
    )
  }
}
```

```r
# resumen gráfico de las MTC por variable
resumen.grf(data, resumen.cols)
```






```
# revertimos el formato a variable categórica una vez terminado el análisis
data$DaysWeek <- factor(data$DaysWeek)
levels(data$DaysWeek)
```

```
## [1] "1" "2" "3" "4" "5" "6" "7"
```

Una vez finalizado el preprocesamiento sobre el archivo, copiamos el resultado de los datos en un archivo llamado *train_clean.csv*:

```
write.csv(data, file = 'train_clean.csv', row.names = FALSE)
```

Conclusión

Ya tenemos el dataset listo para pasar a la siguiente etapa.

Hemos verificado que la lectura de los datos se ha realizado de forma correcta, revisando si se ha asignado a cada variable el tipo de variable estadística adecuada: cuantitativa discreta o continua, o cualitativa nominal u ordinal.

La limpieza que hemos llevado a cabo ha consistido en tratar:

- errores sintácticos y de normalización en variables,
- inconsistencias entre variables,
- valores atípicos y
- valores perdidos o no observados (NAs).

Este es un proceso básico de limpieza y tratamiento de los datos. Existe una gran cantidad de técnicas para poder manipularlos. Las técnicas que usemos van a depender de nuestro set de datos y el objetivo que buscamos.