PEC1 Preprocesamiento de los datos

Luis Manuel Martin Guerra

24/3/2019

Introducción

Los datos a tratar corresponden a un análisis sobre la satisfacción laboral de los trabajadores de una empresa con el objetivo de investigar si la satisfacción laboral de los trabajadores de la empresa está relacionada con la calificación del trabajo y con el nivel de estudios entre otras variables.

Los datos recogidos por cada trabajador son: la ciudad donde se ubica el puesto de trabajo, el sexo, el nivel de estudios (1: sin estudios, 2: estudios primarios, 3: educación secundaria o educación profesional, 4: universitarios), el tipo de trabajo que realiza (C: cualificado, PC: poco cualificado), la satisfacción laboral del trabajador (comprendida entre 0 y 10), la edad (años), la antiguedad (años), dias de baja en el último año laboral, si tuvo (1) o no baja (0) en el último año laboral (binaria), las horas que trabaja a la semana en promedio, nivel de colesterol en la penultima revisión médica (mmol/L), nivel de colesterol en la última revisión médica (mmol/L).

El archivo se denomina rawData.csv, contiene 1920 registros y 12 variables. Estas variables son: city, sex, educ_level, job_type, happiness, age, seniority, sick_leave, sick_leave_b, work_hours, Cho_initial, Cho_final

Preprocesamiento de los datos

El objetivo de la activdad es preparar el archivo para su posterior análisis. Para llevarlo a cabo se realizarán una serie de preprocesamientos:

1. Carga del archivo de datos.
2. Tipos de variables estadísticas presentes.
3. Correción de posibles errores de variables.
4. Normalizar / Estandarizar las variables cuantitativas.
5. Normalizar / Estandarizar las variables cualitativas.
6. Tratamiento de valores perdidos.
7. Búsqueda de valores atípicos.
8. Breve estudio descriptivo.
9. Creación del archivo corregido.

Criterios a aplicar en el preprocesamiento

Cuando se realiza un preprocesamiento, el analista decide como estandarizar/normalizar las variables. Para ello, establece unos criterios, que idealmente se escriben para homogeneizar versiones o para posteriores preprocesamientos que aparezcan sobre nuevos datos.

Los criterios que se seguirán para la realización del preprocesado son los siguentes:

- El punto (.) es el separador decimal de cualquier variable numérica.
- Los valores de la variable work_hours se estandarizan en una cifra decimal.
- Los nombres de las ciudades se estandarizan con la primera letra de cada palabra en mayúsculas (excepto preposiciones) y el resto en minúsculas. Por ejemplo: Barcelona, Las Palmas de Gran Canaria,...
- Los nombres de género se estandarizan como M y F.
- El nivel de estudios se estandariza con la siguiente abreviatura donde aparece el valor 1 cambiar por el valor N, donde aparece el valor 2 cambiar por el valor P, donde aparece el valor 3 cambiar por el valor S, donde aparece el valor 4 cambiar por el valor U. Siendo N: sin estudios, P: estudios primarios, S: educación secundaria o educación profesional, U: universitarios.
- El tipo de trabajo se estandariza con la siguiente abreviatura donde aparece el valor 1 cambiar por el valor C, donde aparece el valor 2 cambiar por el valor PC. Siendo C trabajo cualificado y PC poco cualificado.
- La inconsistencia entre las variables age vs seniority se produce cuando un trabajador tiene una diferencia entre edad (age) y antiguedad (seniority) menor de 18 años. El criterio de corrección es asignar a la variable antiguedad la edad del trabajador menos 18 años.
- Otra inconsistencia puede ser la provocada por valores negativos en el número de horas semanales promedio (work_hours). El criterio de correción es pasar el valor a positivo.

1. Carga del archivo de datos

Antes de decidir que función emplear para cargar el fichero de datos, inspeccionamos el archivo .csv para ver en que tipo de formato se encuentra.

En este caso concreto. observamos que se trata de un formato .csv inglés, en el que el separador de campos es la coma "," y el separador de decimales es el punto "."

Por lo tanto emplearemos la instrucción read.table para cargar el fichero csv, a la que le indicaremos que queremos que cargue las cabeceras, que los valores ausentes los trate como "NA" y que emplee como separador decimal el punto y como separador de columnas la coma.

```
trabajadores <- read.table(
  "/Users/manu/Documents/UOC - Ciencia de Datos/2 - Estadistica avanzada/PEC 1/rawData.csv",
  header=TRUE, sep=",", na.strings="NA", dec=".", strip.white=TRUE)
```

Una vez cargado el fichero, vamos a realiar una primera exploración de los datos, mediante la instrucción *summary()*

```
summary(trabajadores)
```

```
##                      city          sex        educ_level      job_type
##  Madrid                :280   M      :412   Min.   :1.00   Min.   :1.0
##  Santiago de Compostela:266   F      :399   1st Qu.:1.75   1st Qu.:1.0
##  Barcelona             :254   M      :149   Median :2.50   Median :1.5
##  barcelona             :109     F    :137   Mean   :2.50   Mean   :1.5
##    Madrid              : 96   F      :137   3rd Qu.:3.25   3rd Qu.:2.0
##  santiago de compostela: 96   f      :134   Max.   :4.00   Max.   :2.0
##  (Other)               :819   (Other):552
##    happiness         age          seniority       sick_leave
##  5.23   :  23   Min.   :18.00   Min.   : 0.00   Min.   : 0.000
##  6.23   :  23   1st Qu.:33.00   1st Qu.:17.00   1st Qu.: 0.000
```

```
##  5.33   :  19   Median :40.00   Median :20.00   Median : 0.000
##  6.33   :  19   Mean   :40.33   Mean   :19.99   Mean   : 6.667
##  5.13   :  18   3rd Qu.:47.00   3rd Qu.:24.00   3rd Qu.: 0.000
##  (Other):1806   Max.   :67.00   Max.   :35.00   Max.   :78.000
##  NA's   :  12
##   sick_leave_b      work_hours      Cho_initial     Cho_final
##  Min.   :0.0000   Min.   :-44.70   Min.   :0.95   Min.   :0.990
##  1st Qu.:0.0000   1st Qu.: 35.10   1st Qu.:1.15   1st Qu.:1.250
##  Median :0.0000   Median : 37.90   Median :1.25   Median :1.350
##  Mean   :0.2464   Mean   : 34.49   Mean   :1.20   Mean   :1.351
##  3rd Qu.:0.0000   3rd Qu.: 40.70   3rd Qu.:1.25   3rd Qu.:1.450
##  Max.   :1.0000   Max.   : 88.00   Max.   :1.45   Max.   :1.680
##
```

Seguidamente, vamos a confirmar que se han cargado todos los registros y que se han identificado todas las variables:

```
dim(trabajadores)
```

```
## [1] 1920   12
```

Con esto confirmamos que se han cargado los 1920 registros que contienen 12 variables, tal y como se nos indicaba en el enunciado de la práctica.

2. Tipos de variables estadísticas presentes

Para conocer el tipo de datos de las diferenes variables, empleamos la función *class*, aplicada a todo el dataset

```
sapply(trabajadores, class)
```

```
##         city          sex   educ_level     job_type    happiness
##     "factor"     "factor"    "integer"    "integer"     "factor"
##          age    seniority   sick_leave sick_leave_b   work_hours
##    "integer"    "integer"    "integer"    "integer"    "numeric"
##  Cho_initial    Cho_final
##    "numeric"    "numeric"
```

Vemos que R nos indica que *happines*, *city*, y *sex* son variables de tipo "factor", es decir variables de tipo cualitativos, mientras que el resto son de tipo cuantitativos.

De entre estos últimos nos encontramos con que *educ_level*, *job_tye* , *age*, *seniority*, *sick_leav* y *sick_leave_b* son discretas y que *work_hours*, *Cho_initial* y *Cho_final* son continuas.

3. Correción de errores de asignación de tipos

Vemos que R ha detectado que *job_type*, *educ_level*, *happiness* y *sick_leave_b* tienen un tipo de variable que no les corresponde, ya que según los criterios mencionados anteriormente, deberían ser de tipo factor, mientras que happiness debería ser de tipo numérico.

3.1 Nivel de estudios (educ_level)

La transformamos de integer a factor, para ello empleamos la función as.factor

```
trabajadores$educ_level <-as.factor(trabajadores$educ_level)
```

3.2 Tipo de trabajo(job_type) y Ha causado baja (sick_leave_b)

Hacemos lo mismo, con *job_type* y con *sick_leave_b*. En el caso de *sick_leave_b* esta variable puede tomar valores bien de 0, bien de 1, por lo tanto se comporta como una variable de tipo factor, con dos niveles, 0 y 1. Es por ello que vamos a convertir su tipo en factor.

```
trabajadores$job_type <- as.factor(trabajadores$job_type)

trabajadores$sick_leave_b<- as.factor(trabajadores$sick_leave_b)
```

3.3 Satisfacción laboral (happiness)

Por último, transformaremos la variable *happiness*, de factor a numeric.

Para ello, primero la convertiremos a tipo character:

```
trabajadores$happiness <- as.character(trabajadores$happiness)
```

Una vez la tenemos como tipo character, nos aseguraremos que los separadores decimales se han identificado correctamente, es decir, que todos los valores tienen como separador el punto decimal (.) y no la coma (,).

Cargamos las librerías stringi y SGP para tratar las cadenas de caracteres.

```
library(stringi)
library(SGP)
```

Ahora sustituiremos las posibles comas que pudieran aparecer en el conjunto de datos de la variable, por puntos.

```
trabajadores$happiness <-stri_replace_all_fixed(trabajadores$happiness,
                                                c(","),
                                                c("."), vectorize_all = FALSE)
```

Seguidamente, convertimos a numeric el tipo de datos de la variables *happiness*

```
trabajadores$happiness <- as.numeric(trabajadores$happiness)
```

3.4 Comprobaciones finales

Por último comprobamos que los tipos de datos de las variables se han convertido correctamente:

```
sapply(trabajadores, class)
```

```
##         city          sex   educ_level     job_type    happiness
##     "factor"     "factor"     "factor"     "factor"    "numeric"
##          age    seniority   sick_leave sick_leave_b   work_hours
##    "integer"    "integer"    "integer"     "factor"    "numeric"
##  Cho_initial    Cho_final
##    "numeric"    "numeric"
```

4. Normalizar / Estandarizar las variables cuantitativas.

En este punto vamos a estandarizar los valores de las variables cuantitativas del dataset, (*happiness*, *age*, *seniority*, *sick_leave*, *work_hours*, *Cho_initial* y *Cho_final*).

4.1 Happiness

Para eliminar inconsistencias con posibles valores negativos, vamos a aplicar el valor del cálculo absoluto a todos los valores de esta variable:

```
trabajadores$happiness <- sapply(trabajadores$happiness, abs)
```

Comprobamos como han quedado los valores:

```
head(trabajadores$happiness)
```

```
## [1] 7.23 5.93   NA 6.33 8.93 7.23
```

4.2 Edad (age)

Aplicamos el valor del cálculo del valor absoluto a los valores de esta variable, para eliminar la posible existencia de valores negativos:

```
trabajadores$age <- sapply(trabajadores$age, abs)
```

Comprobamos como han quedado los valores:

```
head(trabajadores$age)
```

```
## [1] 45 36 34 37 35 29
```

4.3 Antigüedad (seniority)

En esta variable pueden tener inconsistencia en los valores debido a la presencia de negativos. Para prevenir esta posible situación, asignaremos como valores de esta variable el resultado de calcular el valor del campo edad menos 18:

```
trabajadores$seniority <- trabajadores$age -18
```

Comprobamos como han quedado los valores:

```
head(trabajadores$seniority)
```

```
## [1] 27 18 16 19 17 11
```

4.4 Dias de baja (sick_leave)

Esta variable es de tipo numérico y por lo tanto debemos asegurarnos, que los separadores decimales se han detectado correctamtente.

Pasamos a tipo character los valores de la variable:

```
trabajadores$sick_leave <- as.character(trabajadores$sick_leave)
```

Reemplazamos las posibles comas que pudieran aparecer por puntos:

```
trabajadores$sick_leave <-stri_replace_all_fixed(trabajadores$sick_leave,
                                                 c(","),
                                                 c("."), vectorize_all = FALSE)
```

Convertimos nuevamente a tipo numeric:

```
trabajadores$sick_leave <- as.numeric(trabajadores$sick_leave)
```

Aplicamos el valor del cálculo del valor absoluto a los valores de esta variable, para eliminar la posible existencia de valores negativos:

```
trabajadores$sick_leave <-sapply(trabajadores$sick_leave, abs)
```

Comprobamos como han quedado los valores:

```
head(trabajadores$sick_leave)
```

```
## [1] 0 0 0 0 0 6
```

4.5 Horas de trabajo semanales (work_hours)

Esta variable es de tipo numérico y por lo tanto debemos asegurarnos, al igual que hicimos anteriormente con *happiness* que los separadores decimales se han detectado correctamtente.

Pasamos a tipo character los valores de la variable:

```
trabajadores$work_hours <- as.character(trabajadores$work_hours)
```

Reemplazamos las posibles comas que pudieran aparecer por puntos:

```
trabajadores$work_hours <-stri_replace_all_fixed(trabajadores$work_hours,
                                                 c(","),
                                                 c("."), vectorize_all = FALSE)
```

Convertimos nuevamente a tipo numeric:

```
trabajadores$work_hours <- as.numeric(trabajadores$work_hours)
```

Aplicamos el valor del cálculo del valor absoluto a los valores de esta variable, para eliminar la posible existencia de valores negativos:

```
trabajadores$work_hours <-sapply(trabajadores$work_hours, abs)
```

Además, estandarizamos los valores para que tengan una cifra decimal:

```
trabajadores$work_hours<-round(trabajadores$work_hours, 1)
```

Comprobamos como han quedado los valores:

```
head(trabajadores$work_hours)
```

```
## [1] 35.4 41.3 36.6 40.5 36.0 32.2
```

4.6 Nivel de colesterol en la penúltima revisión (cho_initial)

Esta variable es de tipo numérico y por lo tanto debemos asegurarnos, que los separadores decimales se han detectado correctamtente.

Pasamos a tipo character los valores de la variable:

```
trabajadores$Cho_initial <- as.character(trabajadores$Cho_initial)
```

Reemplazamos las posibles comas que pudieran aparecer por puntos:

```
trabajadores$Cho_initial <-stri_replace_all_fixed(trabajadores$Cho_initial,
                                                  c(","),
                                                  c("."), vectorize_all = FALSE)
```

Convertimos nuevamente a tipo numeric:

```
trabajadores$Cho_initial <- as.numeric(trabajadores$Cho_initial)
```

Aplicamos el valor del cálculo del valor absoluto a los valores de esta variable, para eliminar la posible existencia de valores negativos:

```
trabajadores$Cho_initial <-sapply(trabajadores$Cho_initial, abs)
```

Comprobamos como han quedado los valores:

```
head(trabajadores$Cho_initial)
```

```
## [1] 1.15 1.05 1.15 1.15 1.15 1.05
```

4.7 Niveles de colesterol en la última revisión (cho_final)

Esta variable es de tipo numérico y por lo tanto debemos asegurarnos, que los separadores decimales se han detectado correctamtente.

Pasamos a tipo character los valores de la variable:

```
trabajadores$Cho_final <- as.character(trabajadores$Cho_final)
```

Reemplazamos las posibles comas que pudieran aparecer por puntos:

```
trabajadores$Cho_final <-stri_replace_all_fixed(trabajadores$Cho_final,
                                                c(","),
                                                c("."), vectorize_all = FALSE)
```

Convertimos nuevamente a tipo numeric:

```
trabajadores$Cho_final <- as.numeric(trabajadores$Cho_final)
```

Aplicamos el valor del cálculo del valor absoluto a los valores de esta variable, para eliminar la posible existencia de valores negativos:

```
trabajadores$Cho_final <-sapply(trabajadores$Cho_final, abs)
```

Comprobamos como han quedado los valores:

```
head(trabajadores$Cho_final)
```

```
## [1] 1.33 1.10 1.25 1.21 1.31 1.18
```

Con esto, ya tenemos todas las variables cuantitativas estandarizadas según los criterios de preprocesamiento establecidos al principio de este documento.

5. Normalizar / Estandarizar las variables cualitativas.

A continuación vamos a aplicar las transformaciones correspondientes para estandarizar las variables cualitatitvas *city*, *sex*, *educ_level*, *job_type* y *sick_leave*, conforme a los criterios de preprocesamiento.

5.1 Ciudad (city)

Ahora mismo tenemos:

```
summary(trabajadores$city)
```

```
##                   barcelona                   Barcelona
##                          25                          94
##                barcelona                   Barcelona
##                           9                          30
##                      madrid                      Madrid
##                          30                          96
##                    madrid                      Madrid
##                          11                          29
##      santiago de compostela     Santiago de Compostela
##                          21                          90
##   santiago de compostela      Santiago de Compostela
##                          10                          35
##                   barcelona                   Barcelona
##                         109                         254
##                barcelona                   Barcelona
##                          27                          92
##                      madrid                      Madrid
##                          79                         280
##                    madrid                      Madrid
##                          29                          86
##      santiago de compostela     Santiago de Compostela
##                          96                         266
##   santiago de compostela      Santiago de Compostela
##                          34                          88
```

Vemos que hay tres ciudades, pero cuyos valores no están estandarizados. Hay nombres con espacios en blanco delante, todas las letras en minúsculas, etc.

Primero aplicamos la función *capwords*, para convertir la primera letra de cada palabra en mayúsculas y además eliminar los espacios en blanco que pudieran existir por delante y por detrás.

```
trabajadores$city<- sapply(trabajadores$city, capwords)
```

Además de todas las funcionalidades anteriormente descritas, *capwords* convierte el tipo de datos de los valores de la variable *city* a tipo character, con lo que nos facilita la siguiente transformación a realizar, que es sustituir la primera letra de las preposiciones que pudiera contener el nombre de la ciudad, de mayúsculas a minúsculas.

Para llevar a cabo el reemplazo, vamos a crear un vector de preposiciones, en el que todas las posibles preposiciones empleadas en nombres de ciudades, están en minúsculas y ademas tienen un espacio en blanco por delante y por detrás.

```
preposiciones <-c(" de ", " del ", " de la "," de los ")
```

Ahora crearemos otro vector, con igual que el anterior, pero con la primera letra de cada preposición en mayúsculas:

```
preposiciones_capital <-c(" De "," Del ", " De La ", " De Los ")
```

El siguiente paso es emplear la función stri_replace_all_fixed para sustituir las preposiciones que pudieran contener los valores de la variable, por sus correspondientes, pero en minúsculas:

```
trabajadores$city <-stri_replace_all_fixed(trabajadores$city,
                                           preposiciones_capital,
                                           preposiciones, vectorize_all = FALSE)
```

Finalmente, convertimos de nuevo el tipo de datos de la variable *city* de character a factor:

```
trabajadores$city <-as.factor(trabajadores$city)
class(trabajadores$city)
```

```
## [1] "factor"
```

Comprobamos que efectivamente hay 3 niveles para los valores de esta variable:

```
summary(trabajadores$city)
```

```
##              Barcelona                 Madrid Santiago de Compostela
##                    640                    640                    640
```

5.2 Género (sex)

Al igual que hemos hecho con la variable *city*, vamos a comprobar los valores introducidos para esta variable:

```
summary(trabajadores$sex)
```

```
##      f      F   f      F         m      M   m      M         f      F
##     45    137     16     41     60    126     16     39    134    399
##   f      F         m      M   m      M
##     51    137    127    412     31    149
```

Vemos que hay realmente solo existen dos niveles, F y M. Sin embargo los valores introducidos representan variaciones en mayúsculas/minúsculas, y en espacios en blanco por delante y por detrás.

Por lo tanto, aplicaremos la función capwords, para eliminar los espacios en blanco y transformar en mayúsuclas todas las letras:

```
trabajadores$sex <- sapply(trabajadores$sex, capwords)
```

Al igual que hicimos con *sex*, ahora volvemos a transformar el tipo de variable, de character a factor:

```
trabajadores$sex <- as.factor(trabajadores$sex)
```

Finalmente comprobamos que los valores se han transformado correctamente:

```
head(trabajadores$sex)
```

```
## [1] M M M M M M
## Levels: F M
```

5.3 Nivel de estudios (educ_level)

Esta variable se transformó a tipo factor en el apartado 3, así que en este punto, aplicaremos las etiquetas correspondientes a los diferentes niveles que puede tomar la variable.

```
trabajadores$educ_level <-factor(trabajadores$educ_level,
                          levels= c(1,2,3,4), labels =c("N", "P", "S", "U"))
```

Comprobamos que se han aplicado:

```
head(trabajadores$educ_level)
```

```
## [1] N N N N N N
## Levels: N P S U
```

5.4 Tipo de trabajo (job_level)

Al igual que con *educ_level*, ya lo transformamos en el apartado 3, así que aplicaremos las etiquetas correspondientes a los diferentes niveles que puede tomar:

```
trabajadores$job_type <- factor(trabajadores$job_type,
                                levels =c(1,2), labels=c("C","PC"))
```

Comprobamos los resultados:

```
head(trabajadores$job_type)
```

```
## [1] C C C C C C
## Levels: C PC
```

5.5 Baja laboral (sick_leave_b)

Al haber transformado el tipo de variable en el apartado 3, aplicaremos las etiquetas de 1=TRUE y 0=FALSE, como etiquetas de los niveles de esta variable:

```
trabajadores$sick_leave_b<- factor(trabajadores$sick_leave_b,
                                   levels=c(0,1), labels=c("F","T"))
```

Comprobamos los resultados:

```
head(trabajadores$sick_leave_b)
```

```
## [1] F F F F F T
## Levels: F T
```

6. Tratamiento de valores perdidos.

Una vez que tenemos los valores de las diferentes variables estandarizados, vamos a realizar el tratamiento de los valores nulos que pudieran existir.

Para ello vamos a obtener los posibles valores nulos, que en la carga del archivo, hemos indicado que se identificasen como NA.

Empleamos la función *is.na.data.frame()* para que nos devuelve un dataframe con los valores TRUE o FALSE para cada variable, dependiendo de si tiene valor NA (TRUE) o no (FALSE)

```
valores_na <-is.na.data.frame(trabajadores)
table(valores_na)
```

```
## valores_na
## FALSE  TRUE
## 23028    12
```

Vemos que efectivamente hay 12 registros en todo el dataset que contienen valores perdidos, identificados como NA.

Una primera opción que nos podríamos plantear sería la eliminación de dichos valores nulos, ya que únicamente representan el 0,0005% del conjunto de valores del archivo.

Sin embargo, tal y como se indica en el enunciado de este ejercicio, optaremos por imputar dichos valores perdidos a partir de los *k-vecinos* más cercanos a dichos valores, usando la distancia de Gower con la información de todas las variables.

6.1 Búsqueda y localización de valores perdidos

El dataset contine 12 registros identificados como NA's, por lo que el siguiente paso es averiguar que variables y registros son los que contienen esos valores perdidos.

Hacemos una primera aproximación con el resumen obtenido en el comando anterior:

```
summary(valores_na)
```

```
##     city            sex           educ_level       job_type
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:1920      FALSE:1920      FALSE:1920      FALSE:1920
##
##  happiness          age          seniority       sick_leave
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:1908      FALSE:1920      FALSE:1920      FALSE:1920
##  TRUE :12
##  sick_leave_b    work_hours      Cho_initial     Cho_final
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:1920      FALSE:1920      FALSE:1920      FALSE:1920
##
```

Esto nos muestra que los valores perdidos se encuentran únicamente en la variable happiness, ya que es la única que tiene registros TRUE.

Para conocer exactamente los registros con valores perdidos, emplearemos la función *which* que nos devolverá la posición de los registros que tienen el valor TRUE (es decir, está identificado como un NA):

```
happiness_nas<-is.na(trabajadores$happiness)
happines_nas_position <- which(happiness_nas, arr.ind = TRUE)
length((happines_nas_position))
```

```
## [1] 12
```

Comprobamos que efectivamente tenemos los 12 registros con valores perdidos o NA's de *happiness*. Si queremos conocer exactamente cuales son, basta con hacer:

```
happines_nas_position
```

```
##  [1]    3    7   12   55   66   76  100  200  300  800 1000 1050
```

Con esto obtenemos las posiciones correspondientes a 12 registros marcados como valores NA's.

6.2 Imputación de valores perdidos

Una vez que hemos detectado que nuestro dataset contiene elementos perdidos (NA's) vamos a proceder a imputar esos valores perdidos a partir de los *k-vecinos* más cercanos a dichos valores, usando la distancia de Gower con la información de todas las variables.

Para ello emplearemos la función *kNN()* de la libreria *VIM*

```
library(VIM)
```

A continuación aplicamos la función kNN() a nuestro dataset:

```
trabajadores_knn <-kNN(trabajadores)
```

Comprobamos los resultados:

```
summary(trabajadores_knn)
```

```
##                       city     sex     educ_level job_type   happiness
##  Barcelona              :640   F:960   N:480      C :960   Min.   : 1.930
##  Madrid                 :640   M:960   P:480      PC:960   1st Qu.: 4.930
##  Santiago de Compostela:640           S:480               Median : 5.930
##                                       U:480               Mean   : 5.891
##                                                           3rd Qu.: 6.830
##                                                           Max.   :10.000
##       age          seniority       sick_leave     sick_leave_b
##  Min.   :18.00   Min.   : 0.00   Min.   : 0.000   F:1447
##  1st Qu.:33.00   1st Qu.:15.00   1st Qu.: 0.000   T: 473
##  Median :40.00   Median :22.00   Median : 0.000
##  Mean   :40.33   Mean   :22.33   Mean   : 6.667
##  3rd Qu.:47.00   3rd Qu.:29.00   3rd Qu.: 0.000
##  Max.   :67.00   Max.   :49.00   Max.   :78.000
##    work_hours     Cho_initial     Cho_final      city_imp
##  Min.   :26.20   Min.   :0.95   Min.   :0.990   Mode :logical
##  1st Qu.:35.50   1st Qu.:1.15   1st Qu.:1.250   FALSE:1920
##  Median :38.20   Median :1.25   Median :1.350
##  Mean   :38.29   Mean   :1.20   Mean   :1.351
##  3rd Qu.:40.90   3rd Qu.:1.25   3rd Qu.:1.450
##  Max.   :88.00   Max.   :1.45   Max.   :1.680
##   sex_imp        educ_level_imp  job_type_imp    happiness_imp
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:1920      FALSE:1920      FALSE:1920      FALSE:1908
##                                                  TRUE :12
##
##
##
##   age_imp        seniority_imp   sick_leave_imp  sick_leave_b_imp
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:1920      FALSE:1920      FALSE:1920      FALSE:1920
##
##
##
##
##  work_hours_imp  Cho_initial_imp Cho_final_imp
##  Mode :logical   Mode :logical   Mode :logical
##  FALSE:1920      FALSE:1920      FALSE:1920
##
##
##
##
```

Vemos que la función ha imputados los valores que faltaban a la variable *happiness* y que además nos ha añadido al dataset el equivalente de las variables a imputar, identificadas por nombre_variable_imp.

Finalmente, cargamos en nuestro dataset definitivo las columnas con los valores normalizados, estandarizados e imputados:

```
trabajadores<-subset(trabajadores_knn, select=city:Cho_final)
```

Comprobamos que se han cargado correctamente los datos:

```
summary(trabajadores)
```

```
##                       city     sex     educ_level job_type   happiness
##  Barcelona              :640   F:960   N:480      C :960   Min.   : 1.930
##  Madrid                 :640   M:960   P:480      PC:960   1st Qu.: 4.930
##  Santiago de Compostela:640           S:480               Median : 5.930
##                                       U:480               Mean   : 5.891
##                                                           3rd Qu.: 6.830
##                                                           Max.   :10.000
##       age          seniority       sick_leave     sick_leave_b
##  Min.   :18.00   Min.   : 0.00   Min.   : 0.000   F:1447
##  1st Qu.:33.00   1st Qu.:15.00   1st Qu.: 0.000   T: 473
##  Median :40.00   Median :22.00   Median : 0.000
##  Mean   :40.33   Mean   :22.33   Mean   : 6.667
##  3rd Qu.:47.00   3rd Qu.:29.00   3rd Qu.: 0.000
##  Max.   :67.00   Max.   :49.00   Max.   :78.000
##    work_hours     Cho_initial     Cho_final
##  Min.   :26.20   Min.   :0.95   Min.   :0.990
##  1st Qu.:35.50   1st Qu.:1.15   1st Qu.:1.250
##  Median :38.20   Median :1.25   Median :1.350
##  Mean   :38.29   Mean   :1.20   Mean   :1.351
##  3rd Qu.:40.90   3rd Qu.:1.25   3rd Qu.:1.450
##  Max.   :88.00   Max.   :1.45   Max.   :1.680
```

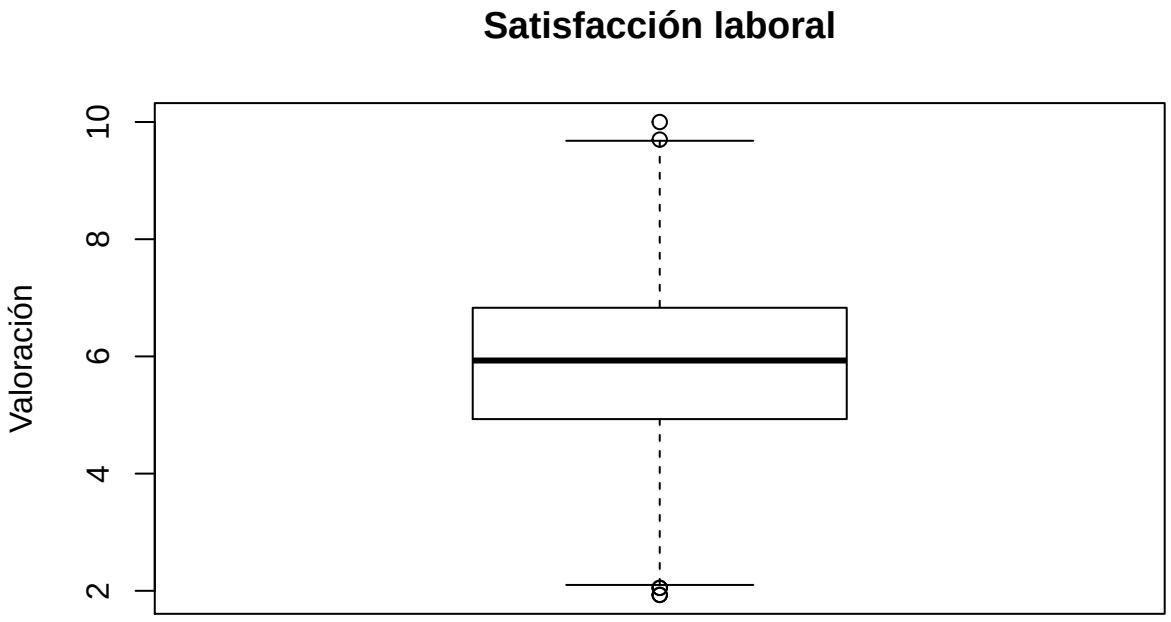
7. Búsqueda de valores atípicos.

En este apartado buscaremos, para de cada una de las variables cuantitativas del dataset, los valores atípicos si es que los tiene. Para ello realizaremos un boxplot y un cuadro con las estimaciones de los indicadores robustos y no robustos de tendecia central y dispersión, para cada una de estas variables.

7.1 Satisfacción laboral (happiness)

Dibujamos el boxplot:

```
boxplot(trabajadores$happiness, main="Satisfacción laboral",
        xlab="Nº de trabajadores", ylab="Valoración")
```



A continuación vamos a calcular los diferentes estimadores robustos y no robustos de tendencia central y de dispersión.

Para ello necesitamos cargar la librería pysch para el cálculo de la media winsorizada:

```
library(psych)
```

El siguiente paso es calcular los indicadores:

```
v_mean<-mean(trabajadores$happiness)
v_median <-median(trabajadores$happiness)
v_cropmean<- mean(trabajadores$happiness, trim=0.05)
v_winsor <- winsor.mean(trabajadores$happiness, trim=0.05)
v_sd <- sd(trabajadores$happiness)
v_IQR <- IQR(trabajadores$happiness)
v_mad <- mad(trabajadores$happiness)
```

Para crear la tabla, crearemos una cabecera, con el nombre de los indicadores, que reutilizaremos para las otras variables:

```
header <-c("Media aritmética", "Mediana", "Media recortada",
           "Media winsorizada", "Desviación estándar",
           "Rango intercuartílico (RIC)", "Desviación absoluta respecto de la mediana (DAM)")
```

Creamos un vector con los valores de las estimaciones:

```
values <- c(v_mean, v_median, v_cropmean, v_winsor, v_sd, v_IQR, v_mad)
```

Finalmente creamos un dataframe con las estimaciones y la cabecera:

```
estimaciones <- data.frame(Indicadores=header, Resultado=values)
```

Cargamos la librería knitr para crear la tabla:

```r
library(knitr)
```

Finalmente tenemos que las estimaciones para la variable *happiness* son:

```r
kable(estimaciones,
      caption="Estimaciones robustas y no robustas de tendencia central
      y dispersión para la variable happiness")
```

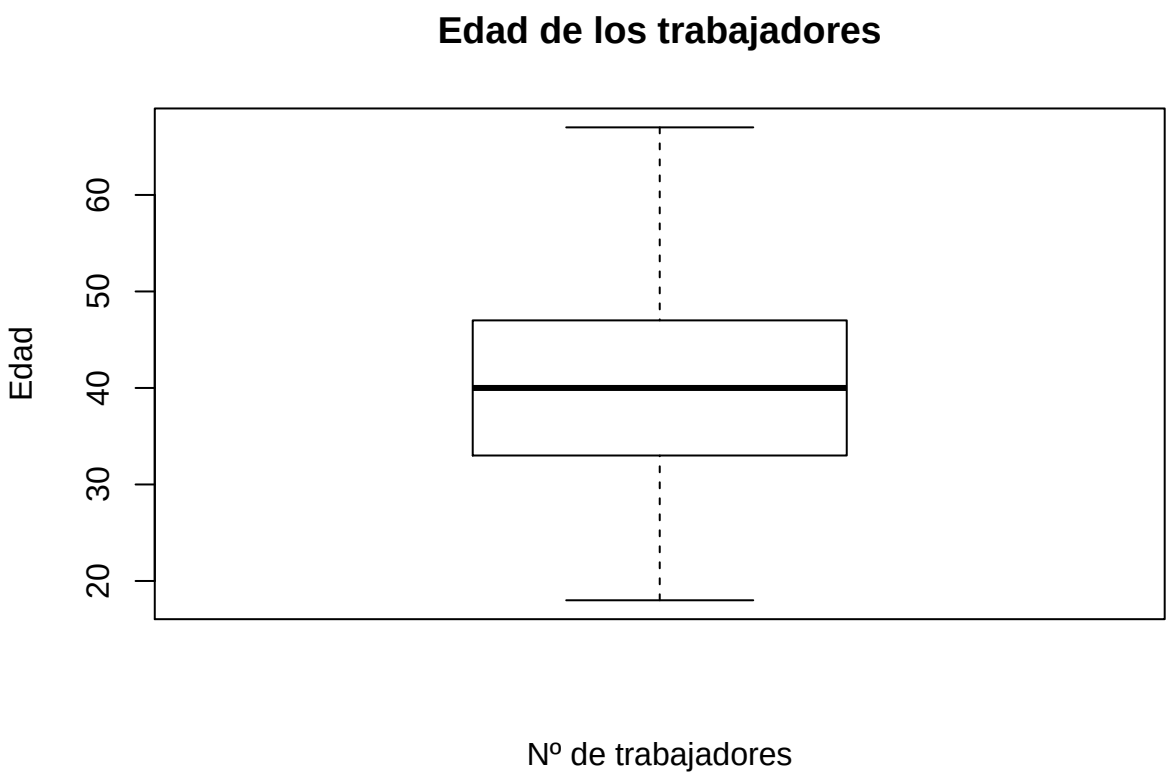
Table 1: Estimaciones robustas y no robustas de tendencia central y dispersión para la variable happiness

Indicadores	Resultado
Media aritmética	5.891219
Mediana	5.930000
Media recortada	5.892882
Media winsorizada	5.889019
Desviación estándar	1.345517
Rango intercuartílico (RIC)	1.900000
Desviación absoluta respecto de la mediana (DAM)	1.371405

7.2 Edad (age)

Dibujamos el boxplot:

```r
boxplot(trabajadores$age, main="Edad de los trabajadores",
        xlab="Nº de trabajadores", ylab="Edad")
```



Calculamos los diferentes estimadores robustos y no robustos de tendencia central y de dispersión:

```
v_mean<-mean(trabajadores$age)
v_median <-median(trabajadores$age)
v_cropmean<- mean(trabajadores$age, trim=0.05)
v_winsor <- winsor.mean(trabajadores$age, trim=0.05)
v_sd <- sd(trabajadores$age)
v_IQR <- IQR(trabajadores$age)
v_mad <- mad(trabajadores$age)
```

Cargamos los valores en el vector de valores que creamos en el apartado 7.1

```
values <- c(v_mean, v_median, v_cropmean, v_winsor, v_sd, v_IQR, v_mad)
```

Actualizamos el dataframe con las estimaciones y la cabecera correspondientes a este valor

```
estimaciones <- data.frame(Indicadores=header, Resultado=values)
```

Finalmente tenemos que las estimaciones para la variable *age* son:

```
kable(estimaciones,
      caption="Estimaciones robustas y no robustas de tendencia central
      y dispersión para la variable age")
```

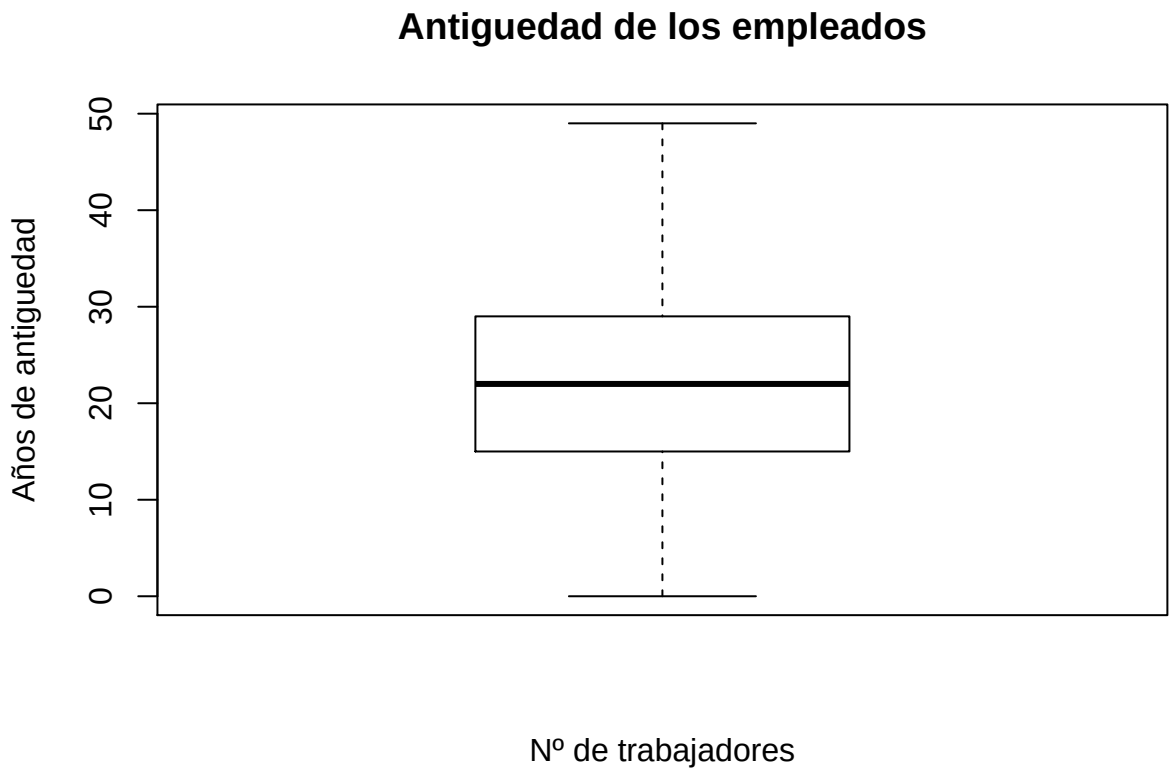
Table 2: Estimaciones robustas y no robustas de tendencia central y dispersión para la variable age

Indicadores	Resultado
Media aritmética	40.331250
Mediana	40.000000
Media recortada	40.247107
Media winsorizada	40.274896
Desviación estándar	9.879243
Rango intercuartílico (RIC)	14.000000
Desviación absoluta respecto de la mediana (DAM)	10.378200

7.3 Antiguedad (seniority)

Dibujamos el boxplot:

```
boxplot(trabajadores$seniority, main="Antiguedad de los empleados",
        xlab="Nº de trabajadores", ylab="Años de antiguedad")
```



Calculamos los diferentes estimadores robustos y no robustos de tendencia central y de dispersión:

```
v_mean<-mean(trabajadores$seniority)
v_median <-median(trabajadores$seniority)
v_cropmean<- mean(trabajadores$seniority, trim=0.05)
v_winsor <- winsor.mean(trabajadores$seniority, trim=0.05)
v_sd <- sd(trabajadores$seniority)
v_IQR <- IQR(trabajadores$seniority)
v_mad <- mad(trabajadores$seniority)
```

Cargamos los valores en el vector de valores que creamos en el apartado 7.1

```
values <- c(v_mean, v_median, v_cropmean, v_winsor, v_sd, v_IQR, v_mad)
```

Actualizamos el dataframe con las estimaciones y la cabecera correspondientes a este valor

```
estimaciones <- data.frame(Indicadores=header, Resultado=values)
```

Finalmente tenemos que las estimaciones para la variable *seniority* son:

```
kable(estimaciones,
      caption="Estimaciones robustas y no robustas de tendencia central
      y dispersión para la variable seniority")
```

Table 3: Estimaciones robustas y no robustas de tendencia central y dispersión para la variable seniority

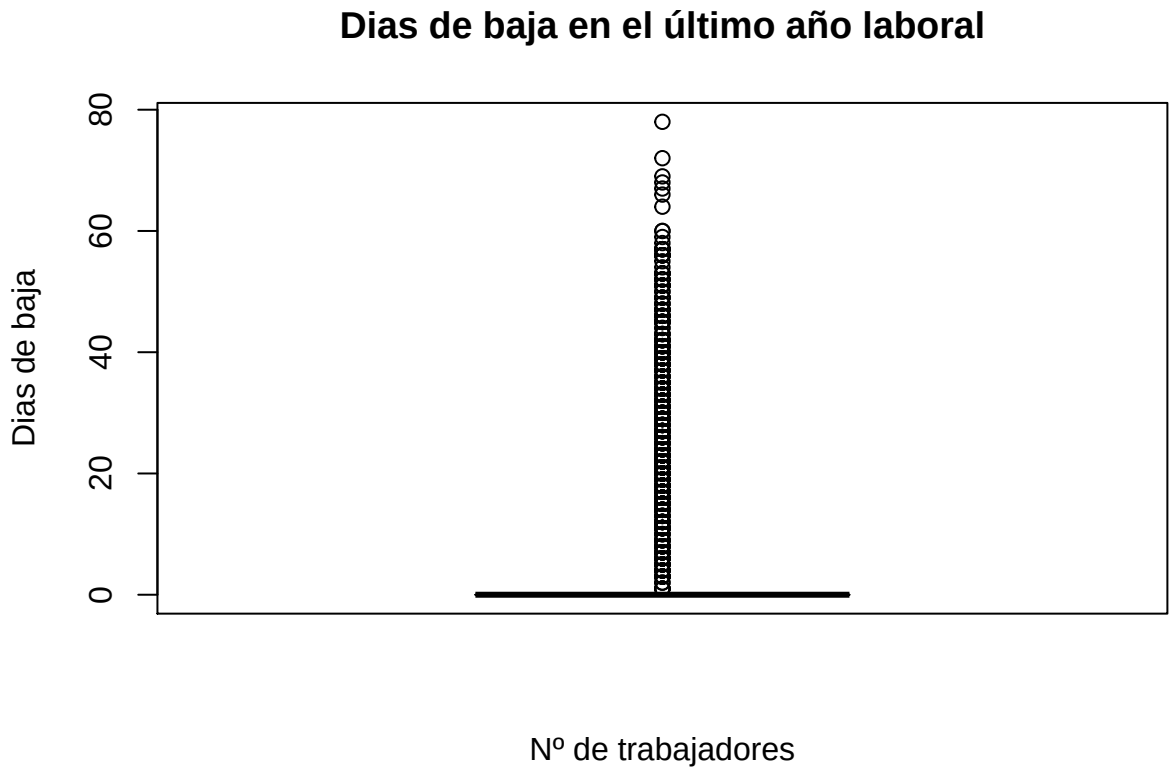
Indicadores	Resultado
Media aritmética	22.331250
Mediana	22.000000
Media recortada	22.247107
Media winsorizada	22.274896

Indicadores	Resultado
Desviación estándar	9.879243
Rango intercuartílico (RIC)	14.000000
Desviación absoluta respecto de la mediana (DAM)	10.378200

7.4 Dias de baja en el último año laboral (sick_leave)

Dibujamos el boxplot:

```
boxplot(trabajadores$sick_leave, main="Dias de baja en el último año laboral",
        xlab="Nº de trabajadores", ylab="Dias de baja")
```



Calculamos los diferentes estimadores robustos y no robustos de tendencia central y de dispersión:

```
v_mean<-mean(trabajadores$sick_leave)
v_median <-median(trabajadores$sick_leave)
v_cropmean<- mean(trabajadores$sick_leave, trim=0.05)
v_winsor <- winsor.mean(trabajadores$sick_leave, trim=0.05)
v_sd <- sd(trabajadores$sick_leave)
v_IQR <- IQR(trabajadores$sick_leave)
v_mad <- mad(trabajadores$sick_leave)
```

Cargamos los valores en el vector de valores que creamos en el apartado 7.1

```
values <- c(v_mean, v_median, v_cropmean, v_winsor, v_sd, v_IQR, v_mad)
```

Actualizamos el dataframe con las estimaciones y la cabecera correspondientes a este valor

```
estimaciones <- data.frame(Indicadores=header, Resultado=values)
```

Finalmente tenemos que las estimaciones para la variable *sick_leave* son:

```
kable(estimaciones,
      caption="Estimaciones robustas y no robustas de tendencia central
      y dispersión para la variable sick_leave")
```

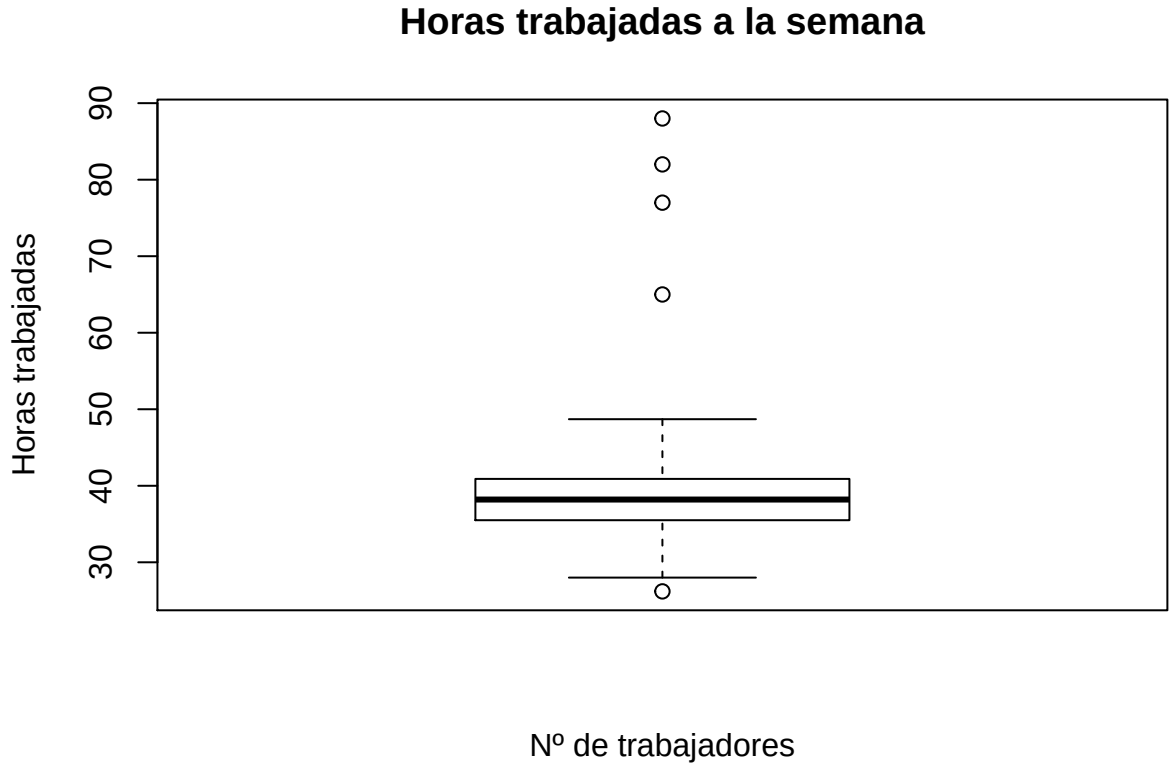
Table 4: Estimaciones robustas y no robustas de tendencia central y dispersión para la variable sick_leave

Indicadores	Resultado
Media aritmética	6.666667
Mediana	0.000000
Media recortada	4.640046
Media winsorizada	6.176042
Desviación estándar	13.899975
Rango intercuartílico (RIC)	0.000000
Desviación absoluta respecto de la mediana (DAM)	0.000000

7.5 Horas de trabajo a la semana (work_hours)

Dibujamos el boxplot:

```
boxplot(trabajadores$work_hours, main="Horas trabajadas a la semana",
        xlab="Nº de trabajadores", ylab="Horas trabajadas")
```



Calculamos los diferentes estimadores robustos y no robustos de tendencia central y de dispersión:

```
v_mean<-mean(trabajadores$work_hours)
v_median <-median(trabajadores$work_hours)
v_cropmean<- mean(trabajadores$work_hours, trim=0.05)
v_winsor <- winsor.mean(trabajadores$work_hours, trim=0.05)
```

```
v_sd <- sd(trabajadores$work_hours)
v_IQR <- IQR(trabajadores$work_hours)
v_mad <- mad(trabajadores$work_hours)
```

Cargamos los valores en el vector de valores que creamos en el apartado 7.1

```
values <- c(v_mean, v_median, v_cropmean, v_winsor, v_sd, v_IQR, v_mad)
```

Actualizamos el dataframe con las estimaciones y la cabecera correspondientes a este valor

```
estimaciones <- data.frame(Indicadores=header, Resultado=values)
```

Finalmente tenemos que las estimaciones para la variable *work_hours* son:

```
kable(estimaciones,
      caption="Estimaciones robustas y no robustas de tendencia central
      y dispersión para la variable work_hours")
```

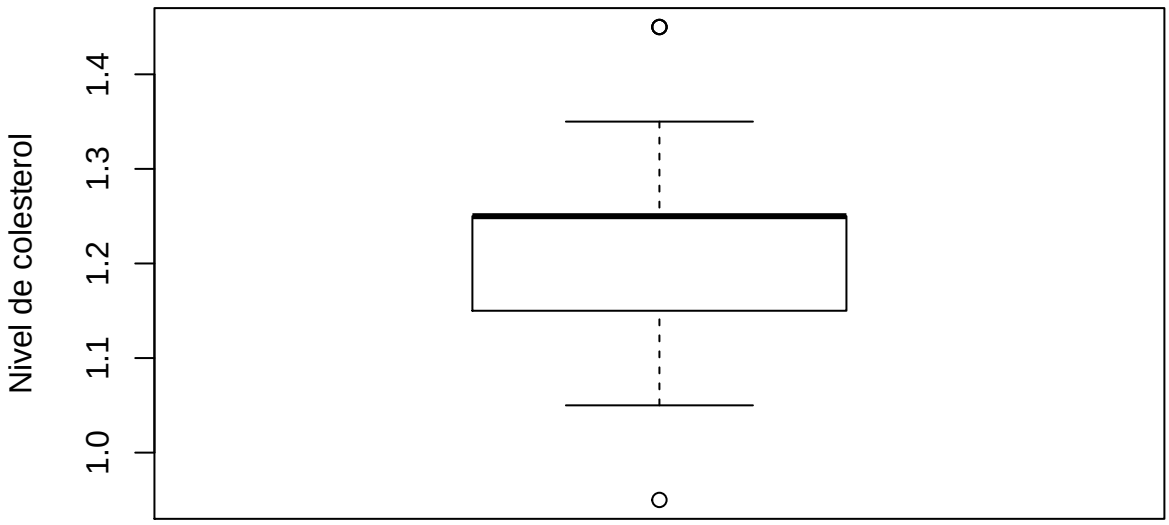
Table 5: Estimaciones robustas y no robustas de tendencia central y dispersión para la variable work_hours

Indicadores	Resultado
Media aritmética	38.286198
Mediana	38.200000
Media recortada	38.216493
Media winsorizada	38.209844
Desviación estándar	4.105982
Rango intercuartílico (RIC)	5.400000
Desviación absoluta respecto de la mediana (DAM)	4.003020

7.6 Nivel de colesterol en la penúltima revisión (Cho_initial)

Dibujamos el boxplot:

```
boxplot(trabajadores$Cho_initial,
        main="Nivel de colesterol de los trabajadores en la penúltima revisión",
        xlab="Nº de trabajadores", ylab="Nivel de colesterol")
```



Calculamos los diferentes estimadores robustos y no robustos de tendencia central y de dispersión:

```
v_mean<-mean(trabajadores$Cho_initial)
v_median <-median(trabajadores$Cho_initial)
v_cropmean<- mean(trabajadores$Cho_initial, trim=0.05)
v_winsor <- winsor.mean(trabajadores$Cho_initial, trim=0.05)
v_sd <- sd(trabajadores$Cho_initial)
v_IQR <- IQR(trabajadores$Cho_initial)
v_mad <- mad(trabajadores$Cho_initial)
```

Cargamos los valores en el vector de valores que creamos en el apartado 7.1

```
values <- c(v_mean, v_median, v_cropmean, v_winsor, v_sd, v_IQR, v_mad)
```

Actualizamos el dataframe con las estimaciones y la cabecera correspondientes a este valor

```
estimaciones <- data.frame(Indicadores=header, Resultado=values)
```

Finalmente tenemos que las estimaciones para la variable *Cho_initial* son:

```
kable(estimaciones,
      caption="Estimaciones robustas y no robustas de tendencia central
      y dispersión para la variable Cho_initial")
```

Table 6: Estimaciones robustas y no robustas de tendencia central y dispersión para la variable Cho_initial

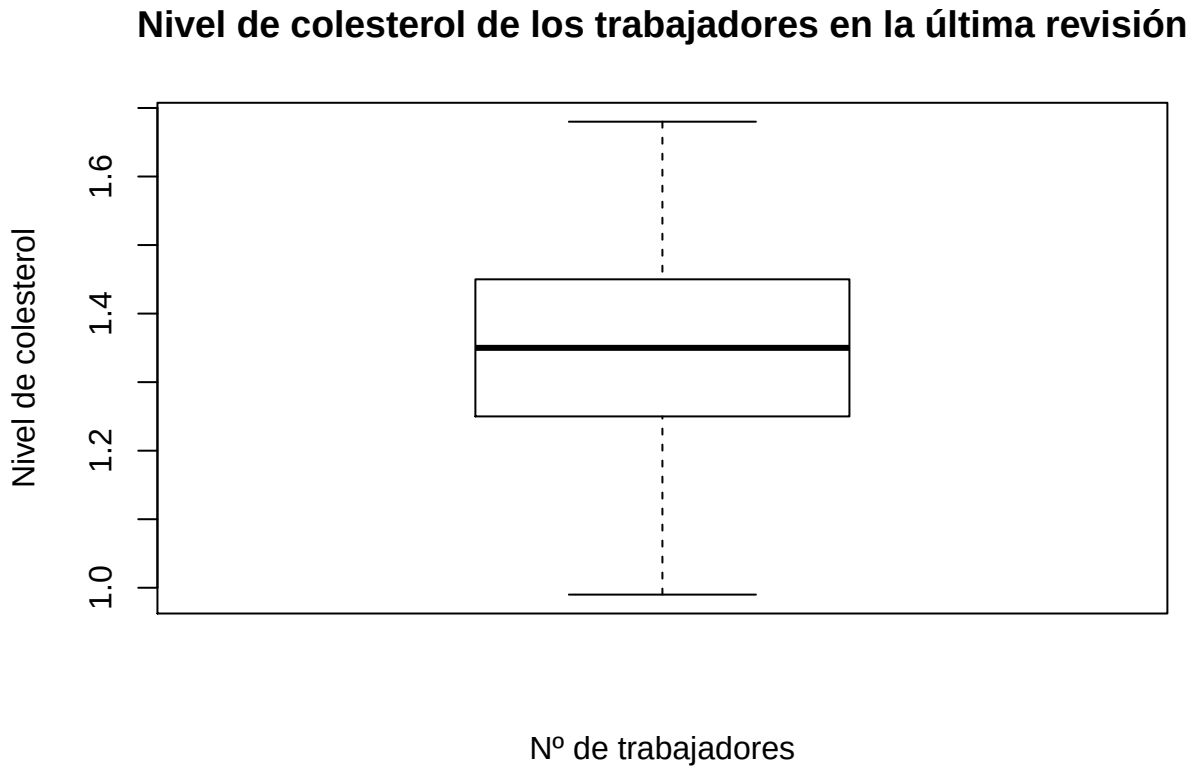
Indicadores	Resultado
Media aritmética	1.2002083
Mediana	1.2500000
Media recortada	1.2001157
Media winsorizada	1.2001042

Indicadores	Resultado
Desviación estándar	0.0771425
Rango intercuartílico (RIC)	0.1000000
Desviación absoluta respecto de la mediana (DAM)	0.1482600

7.7 Niveles de colesterol en la última revisión médica (Cho_final)

Dibujamos el boxplot:

```
boxplot(trabajadores$Cho_final,
        main="Nivel de colesterol de los trabajadores en la última revisión",
        xlab="Nº de trabajadores", ylab="Nivel de colesterol")
```



Calculamos los diferentes estimadores robustos y no robustos de tendencia central y de dispersión:

```
v_mean<-mean(trabajadores$Cho_final)
v_median <-median(trabajadores$Cho_final)
v_cropmean<- mean(trabajadores$Cho_final, trim=0.05)
v_winsor <- winsor.mean(trabajadores$Cho_final, trim=0.05)
v_sd <- sd(trabajadores$Cho_initial)
v_IQR <- IQR(trabajadores$Cho_final)
v_mad <- mad(trabajadores$Cho_final)
```

Cargamos los valores en el vector de valores que creamos en el apartado 7.1

```
values <- c(v_mean, v_median, v_cropmean, v_winsor, v_sd, v_IQR, v_mad)
```

Actualizamos el dataframe con las estimaciones y la cabecera correspondientes a este valor

```
estimaciones <- data.frame(Indicadores=header, Resultado=values)
```

Finalmente tenemos que las estimaciones para la variable *Cho_final* son:

```
kable(estimaciones,
      caption="Estimaciones robustas y no robustas de tendencia central
      y dispersión para la variable Cho_final")
```

Table 7: Estimaciones robustas y no robustas de tendencia central y dispersión para la variable Cho_final

Indicadores	Resultado
Media aritmética	1.3513229
Mediana	1.3500000
Media recortada	1.3512789
Media winsorizada	1.3516510
Desviación estándar	0.0771425
Rango intercuartílico (RIC)	0.2000000
Desviación absoluta respecto de la mediana (DAM)	0.1482600

8. Breve estudio descriptivo

A continuación, con los datos ya depurados vamos a realizar un breve estudio descriptivo sobre las variables del dataset.

Para las gráficas que vamos a elaborar, emplearemos la librería *ggplot2* que cargaremos a continuación

```
library(ggplot2)
```

8.1 Felicidad de los empleados

Vamos a empezar con la distribución de la felicidad en base a la edad y por sexos:

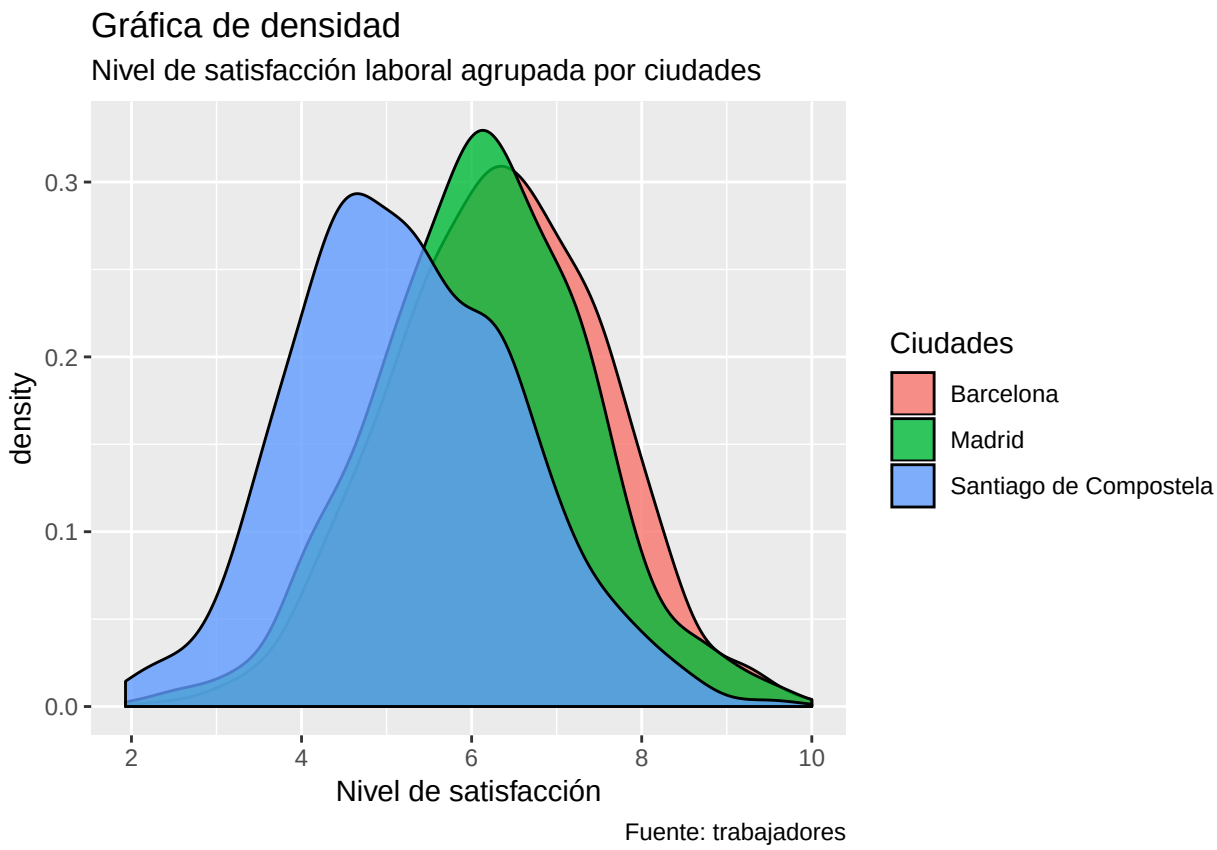
```
ggplot(trabajadores, aes(age, happiness)) + geom_bar(stat = "identity", aes(fill = sex))
```

La gráfica nos muestra que los niveles de felicidad se van incrementándo a la par que lo hace la edad del empleado, hasta un tope en torno a los 44 años, a partir del cual empieza a descender de manera progresiva hasta unos mínimos equiparabls a los de la gente más joven.

Con este primer vistazo podemos aventurar que los trabajadores más felices son aquellos cuya edad oscila entre los 35 y 45 años de edad.

Vamos a ver ahora como puede influir la ciudad en la que viven los trabajadores en la satisfacción laboral:

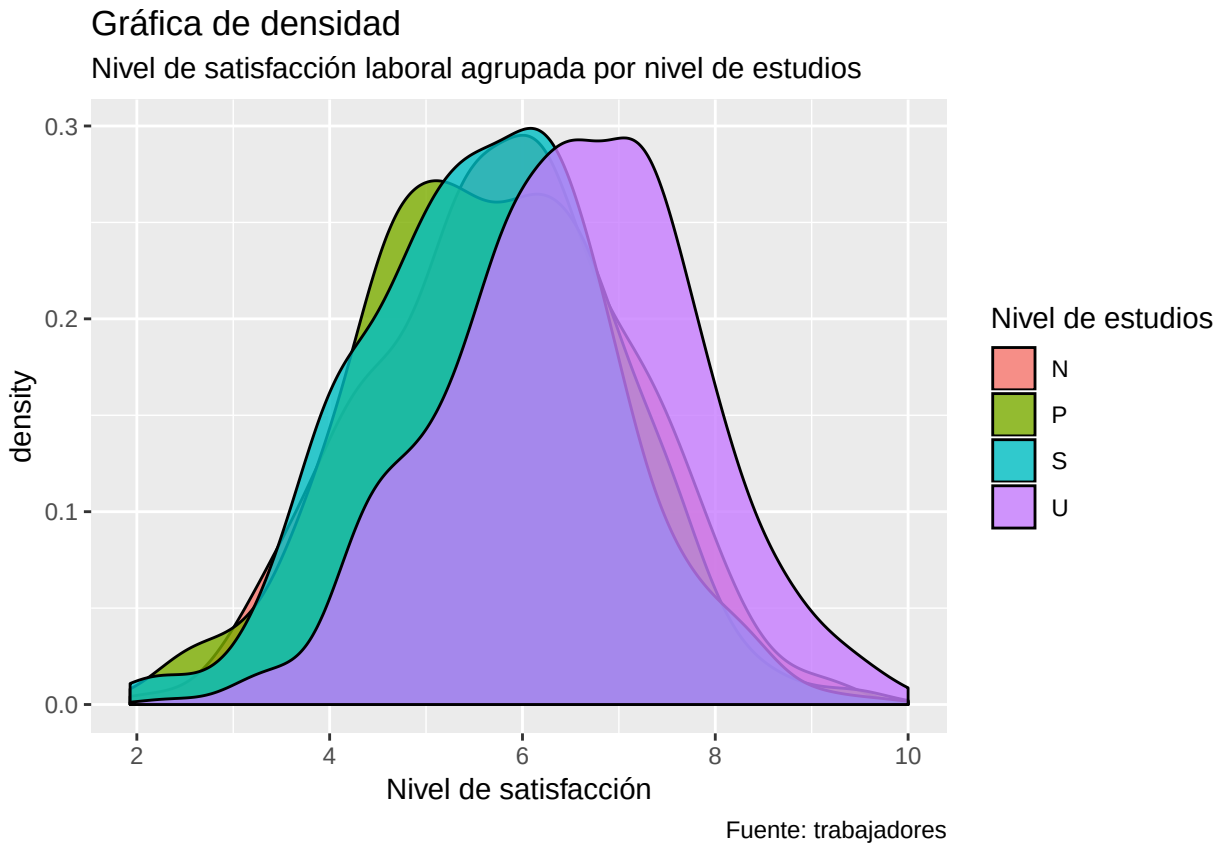
```
ggplot(trabajadores, aes(happiness)) + geom_density(aes(fill=factor(city)), alpha=0.8) +
labs(title = "Gráfica de densidad",
      subtitle = "Nivel de satisfacción laboral agrupada por ciudades",
      caption = "Fuente: trabajadores",
      x="Nivel de satisfacción",
     fill="Ciudades")
```



Con esto observamos que Madrid y Barcelona son las ciudades con un nivel de satisfacción mayor de sus empleados.

Por último, vamos a ver como distribuyen los niveles de satisfacción en función del nivel de estudios:

```
ggplot(trabajadores, aes(happiness)) + geom_density(aes(fill=factor(educ_level)), alpha=0.8) +
     labs(title = "Gráfica de densidad",
     subtitle = "Nivel de satisfacción laboral agrupada por nivel de estudios",
     caption = "Fuente: trabajadores",
     x="Nivel de satisfacción",
     fill="Nivel de estudios")
```



Los resultados nos muestran que los trabajadores con un nivel de estudios universitarios son los que mayor nivel de satisfacción tienen.

8.2 Horas de baja

A continuación vamos a ver como se distribuyen las horas de baja en el trabajo en función de la edad y por sexo:

```
ggplot(trabajadores, aes(age, sick_leave)) + geom_bar(stat = "identity", aes(fill = sex))
```

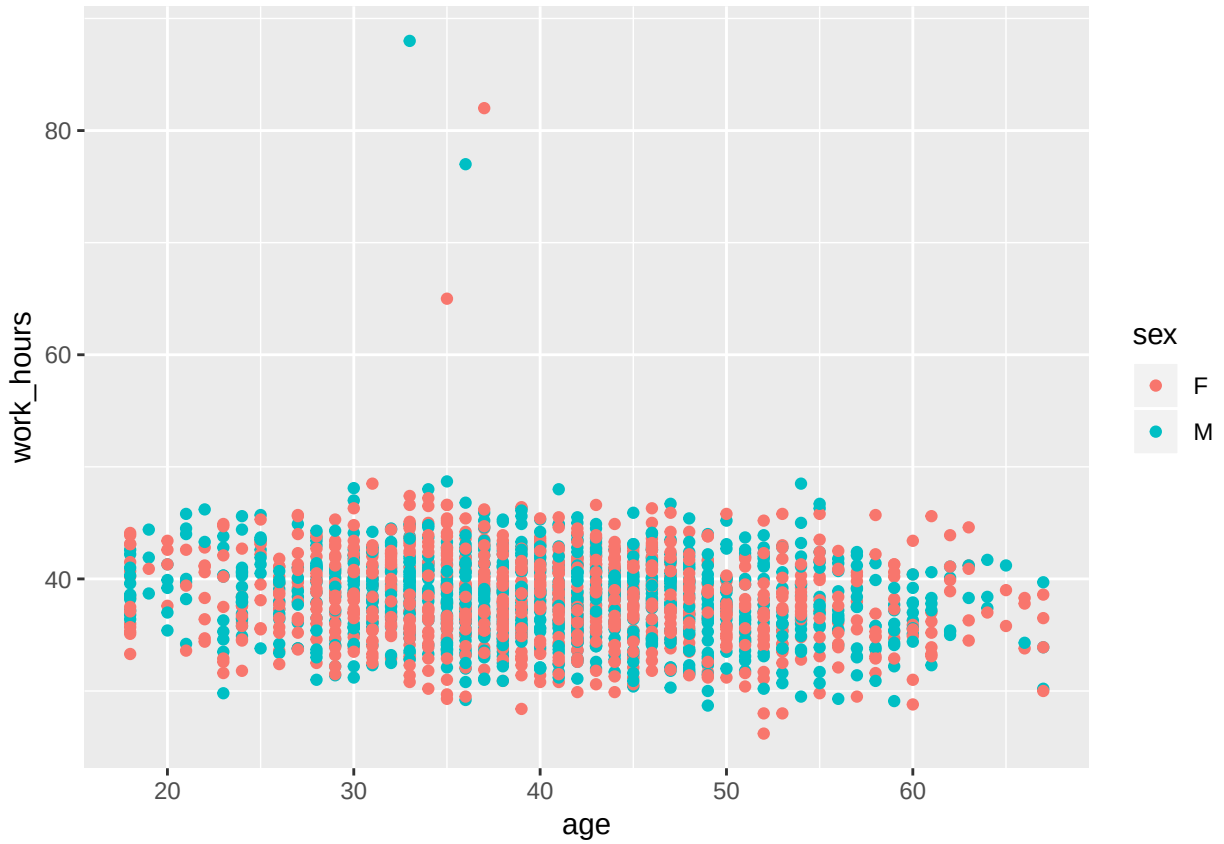
Vemos los hombres cogen más días de baja en el año laboral que las mujeres y que los picos de días se producen tanto a los 31, como a los 38 años, siendo esta última edad a más significativa. Esto nos hace pensar que la edad a la que los empleados deciden tener hijos se ha desplazado a los 38-39 años y por lo tanto las cifras coinciden con lo que se espera de horas de baja por motivos de paternidad/maternidad.

Vamos a explorar ahora

8.3 Horas de trabajo

Vamos a ver ahora si hay alguna diferencia entre el número de horas trabajadas:

```
ggplot(trabajadores, aes(age, work_hours)) + geom_point(aes(col = sex))
```

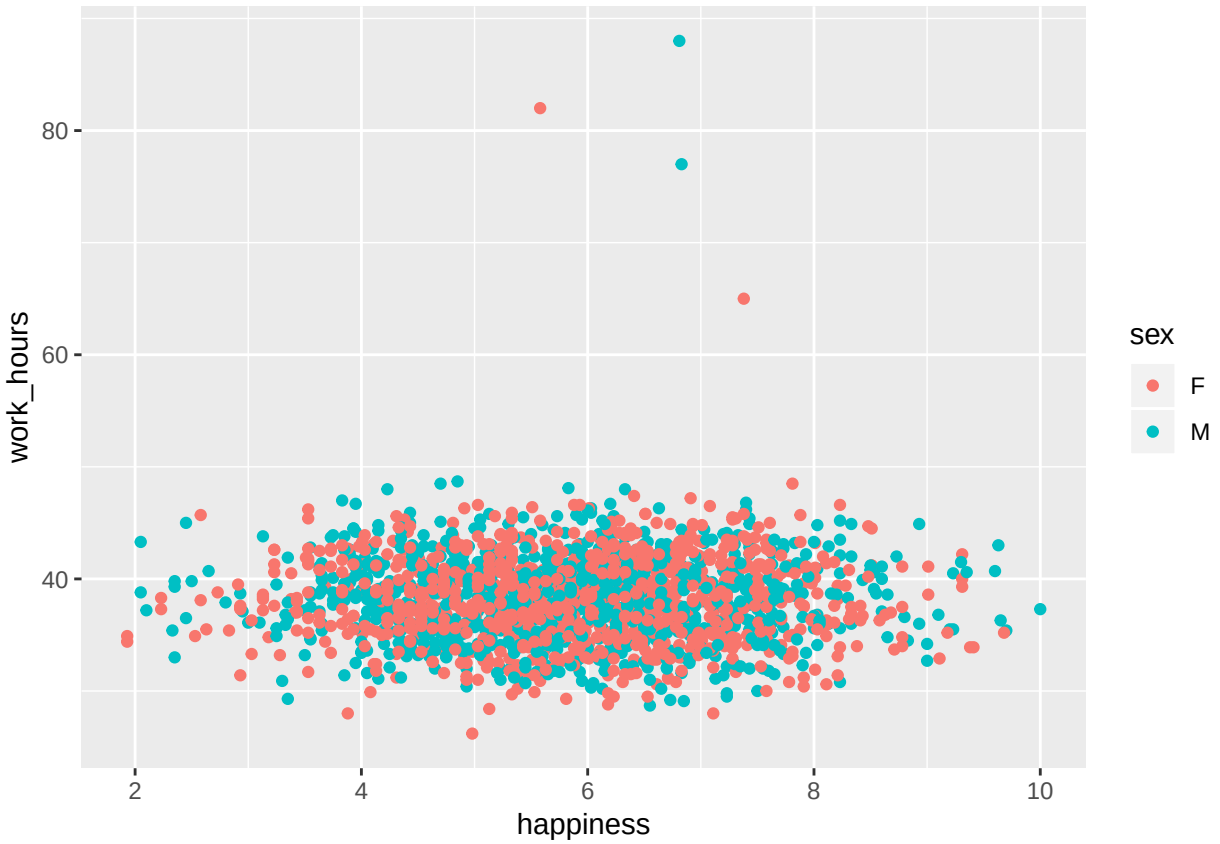


Vemos que el número de horas que se trabaja a la semana permanece casi constante entre las 30 y las 40 horas semanales, con la excepción de algunos casos que superan las 60 horas semanales.

8.4 Horas de trabajo vs Felicidad

Ya hemos visto anteriormente que los niveles de felicidad son mas elevados en cuanto a la edad de los trabajadores, ahora vamos a ver si hay algún tipo de relación entre las horas de trabajo y la felicidad:

```
ggplot(trabajadores, aes(happiness, work_hours)) + geom_point(aes(col = sex))
```

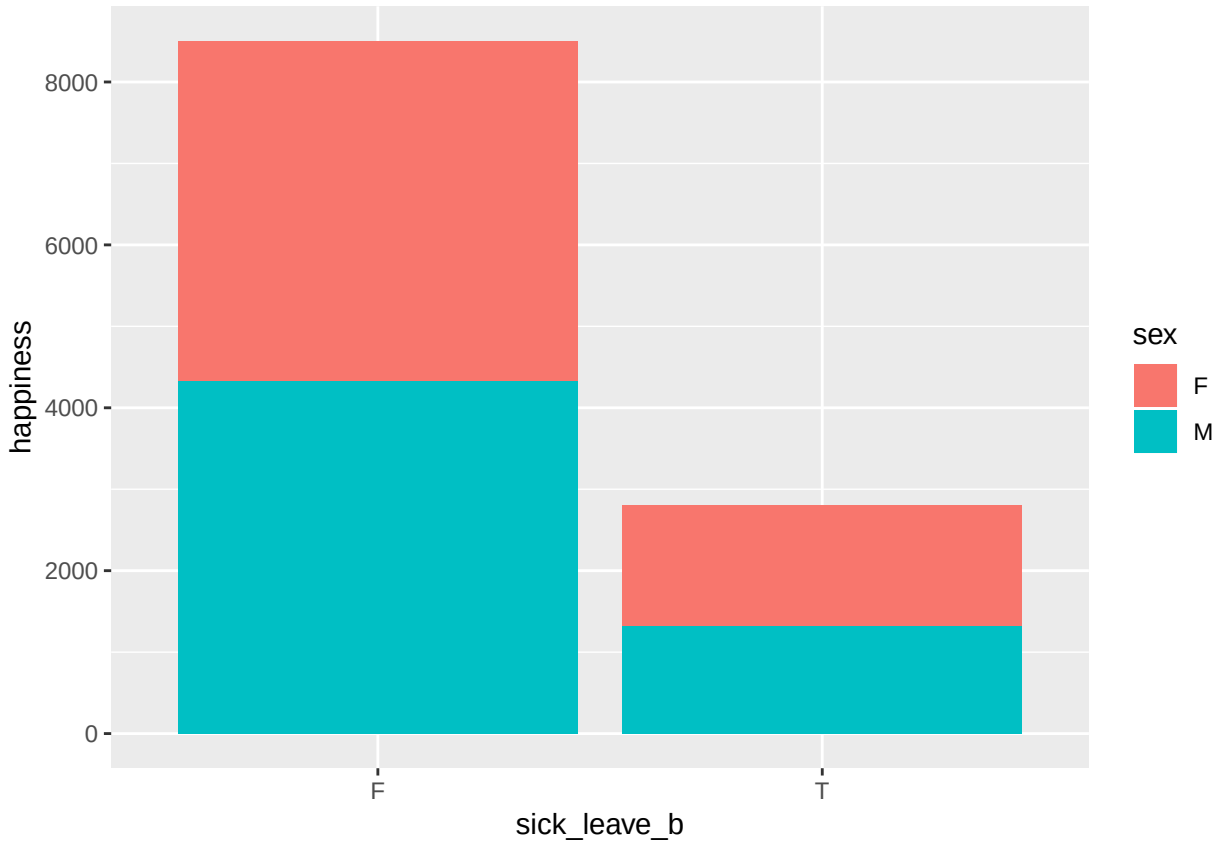


Vemos que los valores centrales de felicidad se distribuyen, salvo valores atípicos de manera uniforme en las horas de trabajo, luego el número de horas trabajadas no afecta de manera significativa a la valoración de la felicidad.

8.5 Baja y felicidad

Con la información que hemos obtenido en los apartados anteriores, vamos a ver los trabajadores que han cogido baja y sus niveles de felicidad:

```
ggplot(trabajadores, aes(sick_leave_b, happiness)) + geom_bar(stat = "identity", aes(fill = sex))
```



Los resultados nos muestran que los trabajadores que no han cogido baja son aquellos con una satisfacción laboral superior a los que si que la han cogido.

8.6 Niveles de colesterol

Vamos a ver ahora los valores de los niveles de colesterol de la penúnltima revisión médica en función de la edad y del sexo.

Primero los niveles iniciales:

```
ggplot(trabajadores, aes(age, Cho_initial)) + geom_bar(stat = "identity", aes(fill = sex))
```

Y ahora vamos a ver los niveles de colesterol de la última revisión médica

```
ggplot(trabajadores, aes(age, Cho_final)) + geom_bar(stat = "identity", aes(fill = sex))
```

Vemos que no hay cambios significativos en los niveles de colesterol de los empleados, de una revisión a otra.

8.7 Conclusiones

Tras este breve análisis podemos concluir que la satisfacción laboral puede estár relacionada con la edad de los empleados, la ciudad en la que estos trabajan y su nivel de estudios. Quizás esto se deba a que a mayor nivel de estudios, mayor categoría profesional y por lo tanto mayor salario. Sin embargo como no tenemos los datos relacionados con el salario, no podemos confirmar esta hipótesis.

También hemos observado que el tanto el número de horas trabajadas, como los niveles de colesterol no influyen de manera significativa en los niveles de satisfacción laboral de los empleados.

9. Creación del archivo corregido

Como último paso, después de haber transformado y estandarizado los tipos y los valores de las variables, creamos el archivo de datos corregido, indicando como separador de columnas la coma (,).

```
write.table(trabajadores,
  "/Users/manu/Documents/UOC - Ciencia de Datos/2 - Estadistica avanzada/PEC 1/martinguerra_luismanuel_
   sep=",", col.names=TRUE, row.names=FALSE, quote=TRUE, na="NA")
```

Con esto se da por concluida la etapa de preprocesamiento de los datos.