

Activitat 1: Exploració i preprocés de dades

Marc Cervera Rosell

05-04-2024

1: Lectura de dades i examinació del tipus de variable

1.1 Carregar el fitxer de dades

Llegir el fitxer de dades i consultar el nom de les columnes del fitxer.

```
tryCatch({  
  data <- read.csv("pisa2009-1.csv", header = TRUE)  
  print("El fitxer s'ha llegit correctament")  
}, error = function(e){  
  cat("ERROR en el moment de llegir el document:", conditionMessage(e), "\n")  
})
```

```
## [1] "El fitxer s'ha llegit correctament"
```

Si per alguna raó el fitxer no es pot llegir, en haver-hi un block tryCatch, es traurà per pantalla un missatge d'error. En cas de poder-se llegir sense problema i que tot vagi bé, a més de llegir-se el fitxer es traurà un missatge per pantalla indicant que el fitxer s'ha llegit correctament.

```
columns <- names(data)  
print(columns)
```

```
## [1] "grade"           "male"           "raceeth"  
## [4] "preschool"      "expectBachelors" "motherHS"  
## [7] "motherBachelors" "motherWork"      "fatherHS"  
## [10] "fatherBachelors" "fatherWork"      "selfBornUS"  
## [13] "motherBornUS"    "fatherBornUS"    "englishAtHome"  
## [16] "computerForSchoolwork" "read30MinsADay" "minutesPerWeekEnglish"  
## [19] "studentsInEnglish" "schoolHasLibrary" "publicSchool"  
## [22] "urban"           "schoolSize"      "readingScore"
```

1.2 Examinar el tipus de variables

Indicar quines variables són de naturalesa numèrica, caràcter i categòrica. En cas que el tipus de variable que ha atorgat R no coincideixi amb el tipus que li correspondria, indicar de quines variables es tracta. Considereu que les variables binàries prenen valors 1 o 0. La transformació corresponent, si és necessària, s'aplicarà en els apartats següents, una vegada normalitzades les variables.

```
type <- sapply(data, class)  
  
for (i in seq_along(columns)) {  
  cat("La columna", columns[i], "és de tipus", type[i], "\n")  
}
```

```
## La columna grade és de tipus integer
## La columna male és de tipus integer
## La columna raceeth és de tipus character
## La columna preschool és de tipus integer
## La columna expectBachelors és de tipus integer
## La columna motherHS és de tipus integer
## La columna motherBachelors és de tipus integer
## La columna motherWork és de tipus integer
## La columna fatherHS és de tipus integer
## La columna fatherBachelors és de tipus integer
## La columna fatherWork és de tipus integer
## La columna selfBornUS és de tipus integer
## La columna motherBornUS és de tipus integer
## La columna fatherBornUS és de tipus integer
## La columna englishAtHome és de tipus integer
## La columna computerForSchoolwork és de tipus integer
## La columna read30MinsADay és de tipus character
## La columna minutesPerWeekEnglish és de tipus integer
## La columna studentsInEnglish és de tipus integer
## La columna schoolHasLibrary és de tipus integer
## La columna publicSchool és de tipus integer
## La columna urban és de tipus integer
## La columna schoolSize és de tipus integer
## La columna readingScore és de tipus character
```

La funció *sapply* ens permet obtenir el tipus de dades que hi ha en cada columna. Per a fer més senzilla la visualització de la categoria i el seu tipus, s'utilitza un bucle de tipus *for* per a treure per pantalla una frase on es relaciona cada categoria amb el seu tipus.

2: Normalització de variables qualitatives (text)

2.1 Variable raceeth

Mostreu les categories de la variable *raceeth*. En cas d'inconsistències o errors, corregiu la informació. A continuació, mostreu el percentatge d'estudiants a cada categoria i dibuixeu un gràfic circular (pie chart).

```
categories_raceeth <- unique(data$raceeth)
print(categories_raceeth)
```

```
## [1] NA
## [2] "White"
## [3] "Black"
## [4] "Hispanic"
## [5] "Assian"
## [6] "white"
## [7] "More than one race"
## [8] "Asian"
## [9] "American Indian/Alaska Native"
## [10] "Native Hawaiian/Other Pacific Islander"
## [11] "Asiann"
## [12] "whit"
```

La funció *unique()* ens permet obtenir els diferents valors únics que hi ha en el conjunt de dades que estem estudiant, en aquest cas, la columna *raceeth*.

```
percentages_rounded <- round(prop.table(table(data$raceeth)) * 100, 2)
for (i in seq_along(categories_raceeth)) {
  cat("La categoria", categories_raceeth[i], "té un percatatge de ",
      percentages_rounded[i], "%", "\n")
}

## La categoria NA té un percatatge de 1.02 %
## La categoria White té un percatatge de 3.75 %
## La categoria Black té un percatatge de 0.14 %
## La categoria Hispanic té un percatatge de 0.06 %
## La categoria Assian té un percatatge de 12.24 %
## La categoria white té un percatatge de 22.99 %
## La categoria More than one race té un percatatge de 3.42 %
## La categoria Asian té un percatatge de 0.85 %
## La categoria American Indian/Alaska Native té un percatatge de 0.08 %
## La categoria Native Hawaiian/Other Pacific Islander té un percatatge de 0.17 %
## La categoria Asiann té un percatatge de 55.29 %
## La categoria whit té un percatatge de NA %
```

En aquest últim apartat cal posar èmfasi en el fet que s'han arrodonit els percentatges i, per tant, no són exactes.

Com s'ha fet anteriorment, per tal de facilitar la visualització dels percentatges, s'ha tret per pantalla una frase amb cada categoria i el tant per cent que representa.

Abans de representar les dades en el diagrama de pastís, es pot observar que en el conjunt de dades hi ha alguns errors. Per exemple: "White", "white", "whit", per tant, abans de representar les dades en el diagrama s'han de rectificar els errors del *dataset*

```
tryCatch({
  data$raceeth <- gsub("\\bwhite\\b", "White", data$raceeth, ignore.case = TRUE)
  data$raceeth <- gsub("\\bwhit\\b", "White", data$raceeth, ignore.case = TRUE)
  data$raceeth <- gsub("\\bAssian\\b", "Asian", data$raceeth, ignore.case = TRUE)
  data$raceeth <- gsub("\\bAsiann\\b", "Asian", data$raceeth, ignore.case = TRUE)
  write.csv(data, "pisa_clean.csv", row.names = FALSE)
  print("File has been corrected succesfully")
}, error = function(e){
  cat("ERROR en aplicar les correccions.", conditionMessage(e), "\n")
})
```

```
## [1] "File has been corrected succesfully"
```

Totes les correccions de les dades qualitatives es guardaran en el nou fitxer corregit en el qual es guardaran totes les correccions que anirem fent. Aquest nou fitxer s'anomena *pisa_clean.csv* Tornem a repetir els passos de lectura i consulta de les categories però ara amb el fitxer corregit.

```
tryCatch({
  data2 <- read.csv("pisa_clean.csv", header = TRUE)
  print("El fitxer amb les dades corregides s'ha llegit correctament")
}, error = function(e){
  print("ERROR en el moment de llegir el document:", conditionMessage(e), "\n")
})
```

```
## [1] "El fitxer amb les dades corregides s'ha llegit correctament"
```

```
categories_raceeth2 <- unique(data2$raceeth)
print(categories_raceeth2)
```

```
## [1] NA
## [2] "White"
## [3] "Black"
## [4] "Hispanic"
## [5] "Asian"
## [6] "More than one race"
## [7] "American Indian/Alaska Native"
## [8] "Native Hawaiian/Other Pacific Islander"
```

Com es pot observar, un cop corregides les categories de la variable qualitativa *raceeth*, apareixen menys categories a representar en el diagrama de pastís.

```
percentages_rounded2 <- round(prop.table(table(data2$raceeth)) * 100, 2)
for (i in seq_along(categories_raceeth2)) {
  cat("La categoria", categories_raceeth2[i], "té un percatatge de ",
      percentages_rounded2[i], "%", "\n")
}
```

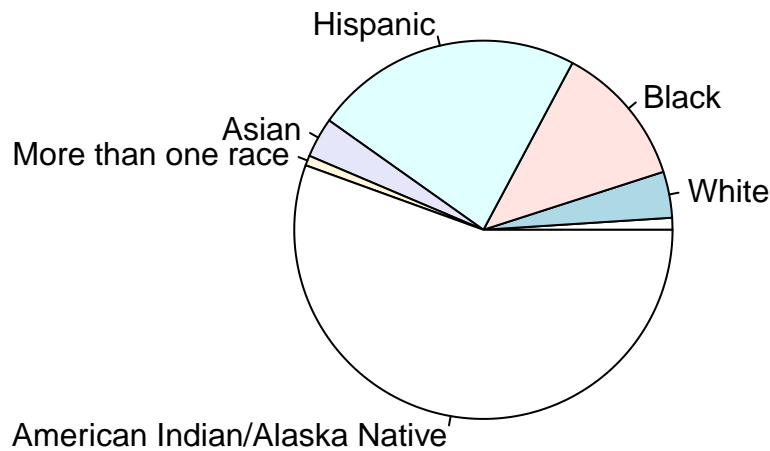
```
## La categoria NA té un percatatge de  1.02 %
## La categoria White té un percatatge de  3.94 %
## La categoria Black té un percatatge de 12.24 %
## La categoria Hispanic té un percatatge de 22.99 %
## La categoria Asian té un percatatge de  3.42 %
## La categoria More than one race té un percatatge de  0.85 %
## La categoria American Indian/Alaska Native té un percatatge de 55.54 %
## La categoria Native Hawaiian/Other Pacific Islander té un percatatge de  NA %
```

```
percentages_not_rounded_aux <- prop.table(table(data2$raceeth)) * 100
print(percentages_not_rounded_aux)
```

```
##
##          American Indian/Alaska Native          Asian
##                1.0198456                3.9415656
##                Black                Hispanic
##                12.2381477                22.9878721
##          More than one race Native Hawaiian/Other Pacific Islander
##                3.4178611                0.8544653
##                White
##                55.5402426
```

En aquesta última cel·la es veuen els percentatges exactes de les categories un cop corregits els noms mal escrits

```
percentages_not_rounded <- prop.table(table(data2$raceeth)) * 100
pie(percentages_not_rounded, labels=categories_raceeth2)
```



En el diagrama anterior es representen les categories corregides. Com s'observa hi ha una categoria del diagrama que no té etiqueta, això és degut al fet que en el fitxer csv aquesta categoria correspon a NA.

3: Normalització i descripció de variables binàries

El conjunt de dades conté un nombre elevat de variables binàries. Revisen els seus valors i en cas d'errors o inconsistències, corregiu els valors a partir dels criteris indicats. A continuació, resumeu en una taula la proporció d'estudiants per als valors positius (1) i els valors negatius (0) d'aquestes variables. Interpreteu breument.

Requisits:

- La taula ha de contenir una variable a cada fila i quatre columnes: nombre d'estudiants amb valor 0 a la variable, nombre d'estudiants amb valor 1, proporció d'estudiants amb valor 0 i proporció d'estudiants amb valor 1.
- Es recomana generar la taula de forma automàtica, sense haver de fer el càlcul manualment per a cada variable. Podeu fer servir funcions de la família *apply* per automatitzar aquest càlcul.

```
total_data <- nrow(data)
calculate_stats <- function(col_name){
  number_of_1 <- sum(col_name == 1, na.rm = TRUE)
  number_of_0 <- sum(col_name == 0, na.rm = TRUE)
  ratio_1 <- number_of_1 / total_data
  ratio_0 <- number_of_0 / total_data
  c(number_of_1, number_of_0, ratio_1, ratio_0)
```

```
}
```

En la cel·la anterior, es mostra l'obtenció del nombre de files que hi ha en el fitxer csv i la definició d'una funció que ens permetrà realitzar el càlcul de les diferents estadístiques. El paràmetre *na.rm* s'estableix a *TRUE*, ja que en les dades tenim valors del tipus *NA* i, per tant, si no establim aquest paràmetre a cert, els càlculs donaran com a resultat *NA*. Aquest paràmetre establert a *TRUE* fa que s'ignorin els valors *NA*.

```
binary_columns <- data[, c("male", "preschool", "expectBachelors", "motherHS",  
                           "motherBachelors", "motherWork", "fatherHS",  
                           "fatherBachelors", "fatherWork", "selfBornUS",  
                           "motherBornUS", "fatherBornUS", "englishAtHome",  
                           "computerForSchoolwork", "schoolHasLibrary",  
                           "publicSchool", "urban")]
```

Un cop executada l'última cel·la, ja tenim seleccionades les columnes amb variables binàries a les quals volem aplicar els diferents càlculs.

```
calculations <- t(apply(binary_columns, 2, calculate_stats))  
# 2 = Apply function in each column
```

Ara ja tenim els càlculs realitzats i en una taula, però si traiem per pantalla la taula, tal com es mostra en la següent cel·la, les columnes no tenen un nom que permeti identificar correctament el càlcul realitzat.

```
print(calculations)
```

```
##           [,1] [,2]      [,3]      [,4]  
## male      1872 1791 0.5110565 0.48894349  
## preschool 2607 1000 0.7117117 0.27300027  
## expectBachelors 2830 771 0.7725908 0.21048321  
## motherHS   3138 428 0.8566749 0.11684412  
## motherBachelors 1137 2129 0.3104013 0.58121758  
## motherWork 2622 948 0.7158067 0.25880426  
## fatherHS   2937 481 0.8018018 0.13131313  
## fatherBachelors 1027 2067 0.2803713 0.56429156  
## fatherWork 2926 504 0.7987988 0.13759214  
## selfBornUS 3347 247 0.9137319 0.06743107  
## motherBornUS 2775 817 0.7575758 0.22304122  
## fatherBornUS 2722 828 0.7431067 0.22604423  
## englishAtHome 3131 461 0.8547639 0.12585313  
## computerForSchoolwork 3236 362 0.8834289 0.09882610  
## schoolHasLibrary 3406 114 0.9298389 0.03112203  
## publicSchool 3421 242 0.9339339 0.06606607  
## urban      1410 2253 0.3849304 0.61506962
```

```
colnames(calculations) <- c("Number of 1", "Number of 0", "Poportion 1", "Proportion 0")
```

Si tornem a treure per pantalla la taula amb els càlculs, podrem observar que, ara sí, les columnes apareixen amb el nom que els hi pertoca.

```
print(calculations)
```

```
##           Number of 1 Number of 0 Poportion 1 Proportion 0  
## male      1872      1791    0.5110565    0.48894349  
## preschool 2607      1000    0.7117117    0.27300027  
## expectBachelors 2830      771    0.7725908    0.21048321  
## motherHS   3138      428    0.8566749    0.11684412  
## motherBachelors 1137      2129    0.3104013    0.58121758
```

## motherWork	2622	948	0.7158067	0.25880426
## fatherHS	2937	481	0.8018018	0.13131313
## fatherBachelors	1027	2067	0.2803713	0.56429156
## fatherWork	2926	504	0.7987988	0.13759214
## selfBornUS	3347	247	0.9137319	0.06743107
## motherBornUS	2775	817	0.7575758	0.22304122
## fatherBornUS	2722	828	0.7431067	0.22604423
## englishAtHome	3131	461	0.8547639	0.12585313
## computerForSchoolwork	3236	362	0.8834289	0.09882610
## schoolHasLibrary	3406	114	0.9298389	0.03112203
## publicSchool	3421	242	0.9339339	0.06606607
## urban	1410	2253	0.3849304	0.61506962

4: Normalització de variables quantitatives

4.1 Variable readingScore

Revisen els valors de la variable `readingScore` i verifiquen que estiguin dins dels marges esperats. Si hi ha algun valor erroni o molt extrem, substituir per NA. Mostreu un gràfic de tipus `boxplot` per visualitzar la distribució d'aquesta variable. Interpreteu el resultat.

En primer lloc, s'ha de seleccionar la columna en la qual volem realitzar la comprovació del rang de valors. Tot seguit aplicar la correcció que es demana tot especificant que la correcció s'aplicarà a aquells valors menors a 0 o a aquells valors majors a 1000. Abans de fer cap correcció, però, cal transformar els valors a valors numèrics, ja que segons consta a la tercera cel·la de codi (on s'han consultat els tipus), els valors d'aquesta variable són de tipus *character*. Un cop fetes les correccions, es guardaran en el fitxer de correccions que hem creat anteriorment anomenat *pisa_clean.csv*.

```
data$readingScore <- as.numeric(data$readingScore) # Conversio str-int
```

```
## Warning: NAs introduced by coercion
```

```
data$readingScore[data$readingScore < 0 | data$readingScore > 1000] <- NA
cat("Correccions aplicades correctament")
```

```
## Correccions aplicades correctament
```

Un cop fetes les correccions, cal guardar-les al fitxer de correccions.

```
tryCatch({
  write.csv(data, "pisa_clean.csv", row.names = FALSE)
  cat("Correccions aplicades al fitxer final")
}, error = function(e){
  cat("ERROR en escriure els canvis.", conditionMessage(e), "\n")
})
```

```
## Correccions aplicades al fitxer final
```

En aquest punt, les correccions han estat guardades.

Finalment, solament queda treure per pantalla el gràfic de tipus `boxplot`, prèvia lectura del fitxer amb les noves actualitzacions.

```
tryCatch({
  data2 <- read.csv("pisa_clean.csv", header = TRUE)
  print("El fitxer amb les dades corregides s'ha llegit correctament")
}, error = function(e){
```

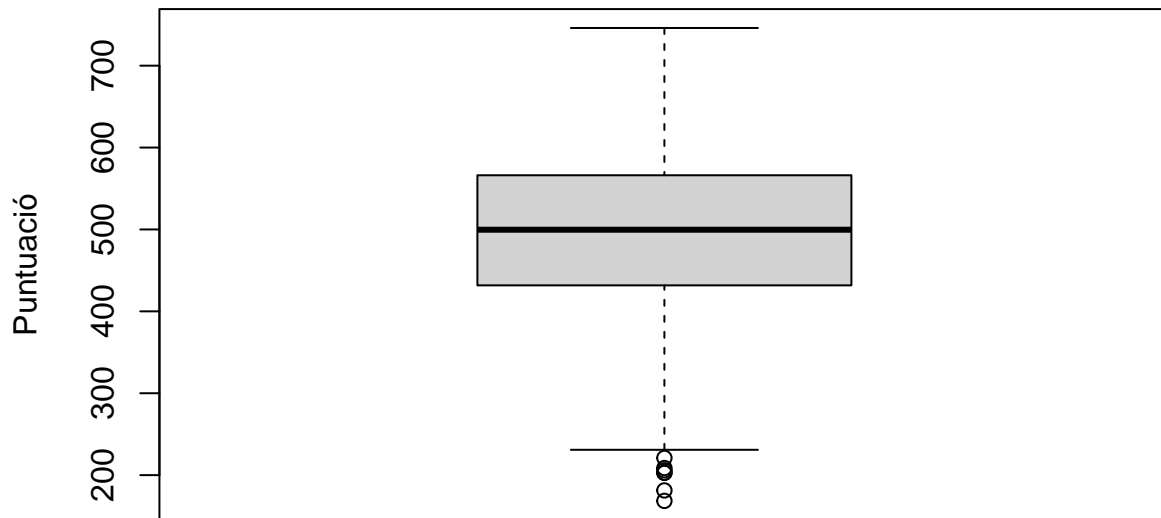
```

print("ERROR en el moment de llegir el document:",conditionMessage(e), "\n")
})

## [1] "El fitxer amb les dades corregides s'ha llegit correctament"
boxplot(data2$readingScore, main = "Diagrama de caixa de les puntuacions de lectura",
        ylab = "Puntuació", na.action = na.pass)

```

Diagrama de caixa de les puntuacions de lectura



4.2 Variable grade

Mostreu visualment la distribució de la variable grade (curs). A continuació, reviseu si els valors de la variable grade estan dins dels marges raonables. Per a la mostra d'estudi, composta per estudiants de 15 anys, es correspondria al desè curs. Hi poden haver casos d'estudiants que estiguin en cursos més avançats o en cursos inferiors. Si hi ha un valor extrem o erroni, s'ha de substituir per NA.

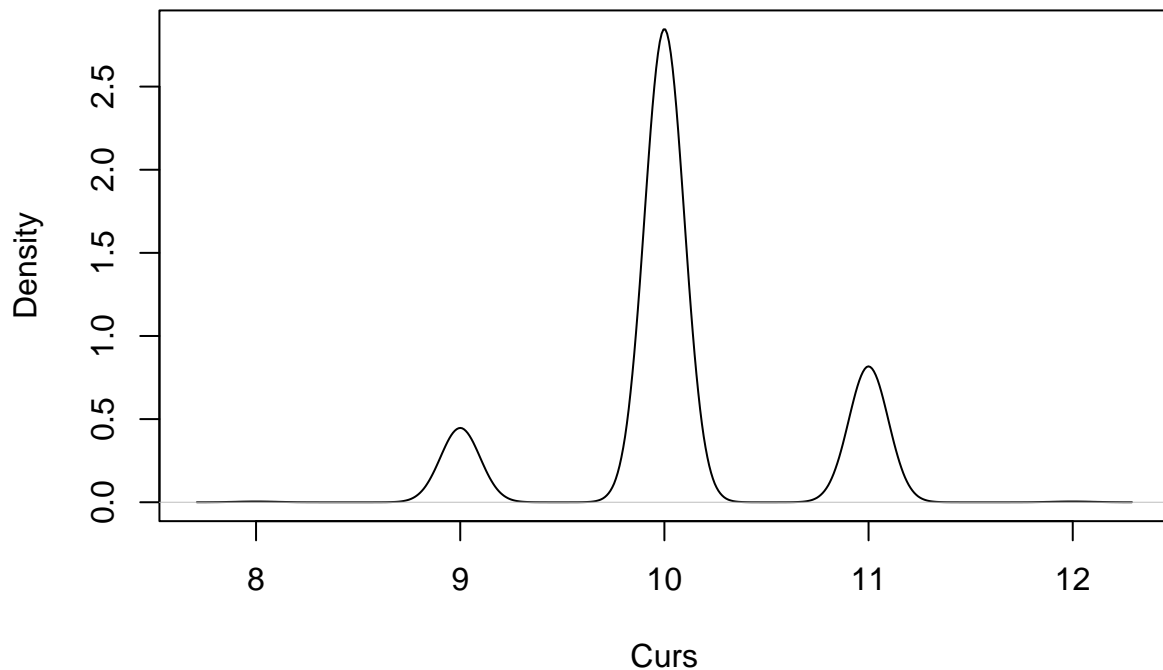
Per a mostrar la distribució de la variable es considera que la millor opció és mitjançant un gràfic de densitat, ja que aquests mostren la forma real de la distribució.

```

plot(density(data$grade), main = "Distribució de cursos", xlab = "Curs")

```


Distribució de cursos



Com es pot observar en el gràfic, la gran majoria de la densitat se l'emporta la categoria del desè curs (tal com s'esperava). En aquest cas, es considera que no hi ha cap valor extrem/erroni atès que en el gràfic de densitat els cursos que tenen alguna rellevància (a més del desè curs) són un curs per sota o un curs per sobre i, per tant, s'ajusta al que diu l'enunciat respecte a què hi poden haver casos d'estudiants que estiguin en cursos més avançats o en cursos inferiors.

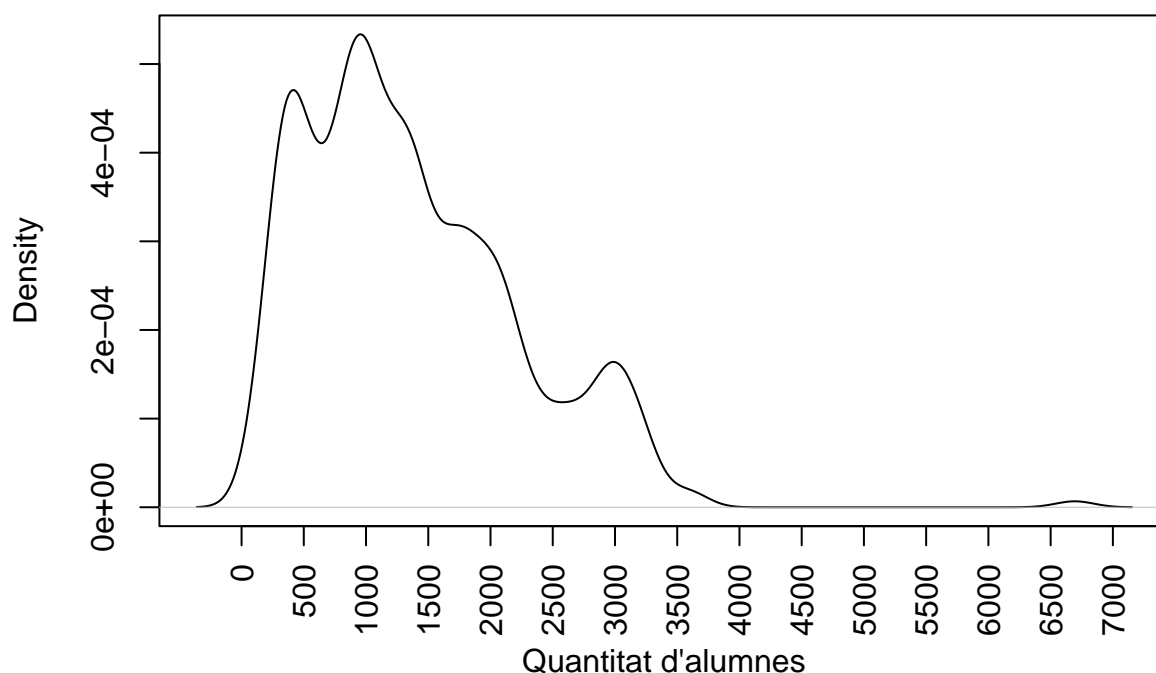
4.3 Variable schoolSize

Mostreu visualment la distribució de la variable `schoolSize`. Si hi ha valors erronis, substituiu per NA. La imputació es farà més endavant.

Pel mateix motiu que en l'apartat anterior, es mostrarà la distribució de la variable mitjançant un gràfic de densitat.

```
par(xaxt = "n") # No incloure les etiquetes de l'eix x
plot(density(data$schoolSize, na.rm = TRUE), main = "Nombre d'estudiants a l'escola",
     xlab = "Quantitat d'alumnes")
par(xaxt = "s") # Incloure les etiquetes de l'eix x
axis(1, at = seq(0, 8000, by = 500), las = 2)
```

Nombre d'estudiants a l'escola



```
filtrat <- data$schoolSize[data$schoolSize <= 0 & !is.na(data$schoolSize)]  
print(filtrat)
```

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

Com es pot observar en el gràfic de dispersió, la gran majoria d'escoles es troben en un nivell d'alumnat d'entre 1000 i 1500 alumnes. En aquest cas, no es considera que hi hagi valors erronis en el conjunt de dades atès que l'únic valor que es pot considerar erroni és que l'escola tingui un nombre negatiu d'alumnes o que no tingui alumnes i com es pot comprovar en la cel·la de codi anterior no hi ha cap valor igual o inferior a 0 pel que fa a nombre d'alumnes.

5: Imputació

En aquest apartat, farem la imputació sobre els valors perduts de la variable `schoolSize`. Apliqueu imputació per veïns més propers, utilitzant la distància de Gower, considerant en el còmput dels veïns més propers les variables numèriques. Per realitzar aquesta imputació, es pot fer servir la funció “kNN” de la llibreria VIM amb un nombre de veïns igual a 5. Demostreu que la imputació s'ha realitzat correctament, visualitzant algunes de les dades afectades per la imputació. Finalment, analitzeu des d'un punt de vista crític el procés d'imputació realitzat.

Després d'instal·lar (i carregar) la llibreria VIM es pot procedir a resoldre l'exercici. Per fer-ho, el primer pas serà seleccionar aquelles columnes que volem utilitzar per calcular la distància de Gower. En el nostre cas, i tal com indica l'enunciat, s'usaran totes les variables numèriques. Un cop seleccionades les variables, s'emprarà la funció `kNN` de la llibreria que s'acaba d'instal·lar tot indicant un nombre de veïns igual a 5 per a calcular la distància de Gower.

```
variables_calc_dist <- c("grade", "male", "preschool", "expectBachelors",
                        "motherHS", "motherBachelors", "motherWork",
                        "fatherHS", "fatherBachelors", "fatherWork",
                        "selfBornUS", "motherBornUS", "fatherBornUS",
                        "englishAtHome", "computerForSchoolwork",
                        "schoolHasLibrary", "publicSchool", "urban",
                        "minutesPerWeekEnglish", "studentsInEnglish",
                        "readingScore")
```

Un cop executada la cel·la anterior, queden consolidades les variables a utilitzar. Seguidament es calcula la distància.

```
library(VIM)
```

```
## Loading required package: colorspace
## Loading required package: grid
## VIM is ready to use.
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
##
## Attaching package: 'VIM'
## The following object is masked from 'package:datasets':
##
##      sleep
imputed <- kNN(data2, variable = "schoolSize", dist_var = variables_calc_dist , k = 5)
```

Per comprovar que la imputació s'ha realitzat correctament, es poden comparar les dades originals amb les imputades. Concretament, es compararan les mitjanes abans i després d'imputar.

```
cat("Mitjana abans d'imputar = ",
    mean(data2[is.na(data2$schoolSize),]$schoolSize, na.rm = TRUE), "\n")

## Mitjana abans d'imputar =  NaN

cat("Mitjana després d'imputar = ",
    mean((imputed[rownames(data2[is.na(data2$schoolSize),]),])$schoolSize))

## Mitjana després d'imputar =  1471.599
```

Com s'observa en la comparativa de mitjanes, abans d'imputar la mitjana de *schoolSize* és *NaN* atès que hi ha valors perduts, però després de la imputació sí que s'obté una mitjana numèrica i real. En altres paraules, abans d'imputar, no era possible obtenir una mitjana d'alumnes, ja que hi havia valors *NA*, després d'imputar sí que és possible obtenir aquesta mitjana atès que ja no hi ha valors *NA*.