Introducción a la ciencia de datos.

Marta Verona Almeida

Análisis de datos

En esta sección se realizará un análisis de los conjuntos de datos asignados para conocer en qué consisten los problemas de regresión y clasificación que se afrontarán en los próximos apartados.

Clasificación: Balance dataset.

El conjunto asignado para clasificación se llama **balance** y contiene datos asociados a una balanza. Dispone de cuatro variables numéricas que representan el peso a la derecha y a la izquierda, y distancia a la derecha y a la izquierda. La salida, y etiqueda a buscar, es en un factor que indica si la balanza se encuentra descompensada a derecha o izquierda, o si se encuentra en equilibrio. Además, este conjunto no presenta valores ausentes.

```
# Muestro número y tipo de variables, número de ejemplos,...
str(balance)
```

```
## 'data.frame':    624 obs. of  5 variables:
##  $ Left-weight   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Left-distance : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Right-weight  : num  1 1 1 1 2 2 2 2 2 3 ...
##  $ Right-distance: num  2 3 4 5 1 2 3 4 5 1 ...
##  $ Balance-scale : Factor w/ 3 levels " B"," L"," R": 3 3 3 3 3 3 3 3 3 3 ...
```

Compruebo que no hay datos ausentes en el conjunto.

```
anyNA(balance)
```

```
## [1] FALSE
```

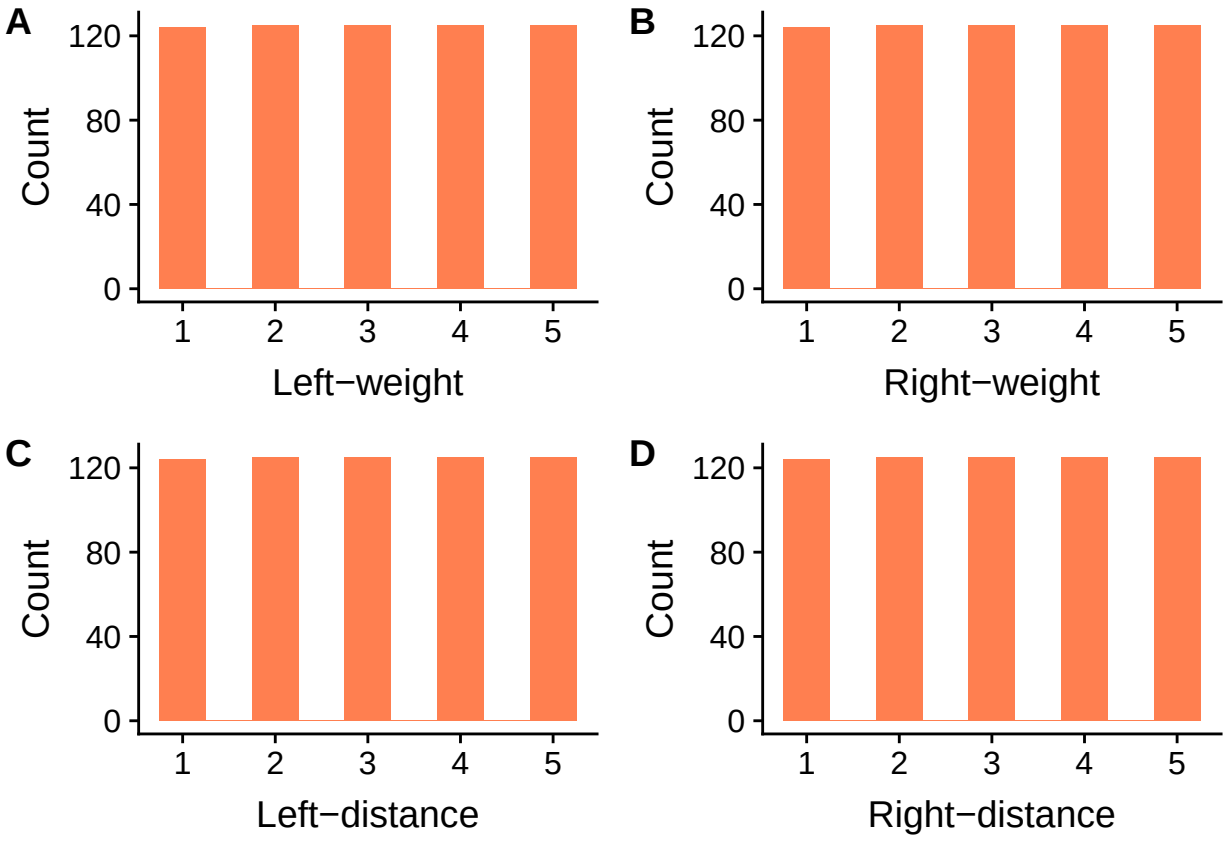
A continuación, se muestran algunos valores estadísticos (mínimo, máximo, media, mediana, primer y tercer cuartil) que permiten tener una idea de cómo están distribuidos.

```
summary(balance)
```

```
##   Left-weight    Left-distance    Right-weight   Right-distance
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.000
##  Median :3.000   Median :3.000   Median :3.000   Median :3.000
##  Mean   :3.003   Mean   :3.003   Mean   :3.003   Mean   :3.003
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :5.000
##  Balance-scale
##   B: 48
##   L:288
##   R:288
##
##
##
```

Se observa que las cuatro variables se encuentran igualmente distribuidas, con valores comprendidos en el intervalo [1,5] y con media y mediana igual a 3. Por lo que suponemos que las cuatro variables se encuentran

escaladas y centradas, cosa que ayudará bastante a la hora de emplear los algoritmos de clasificación. Así mismo, se utiliza ggplot para observar las distribuciones de estas variables.

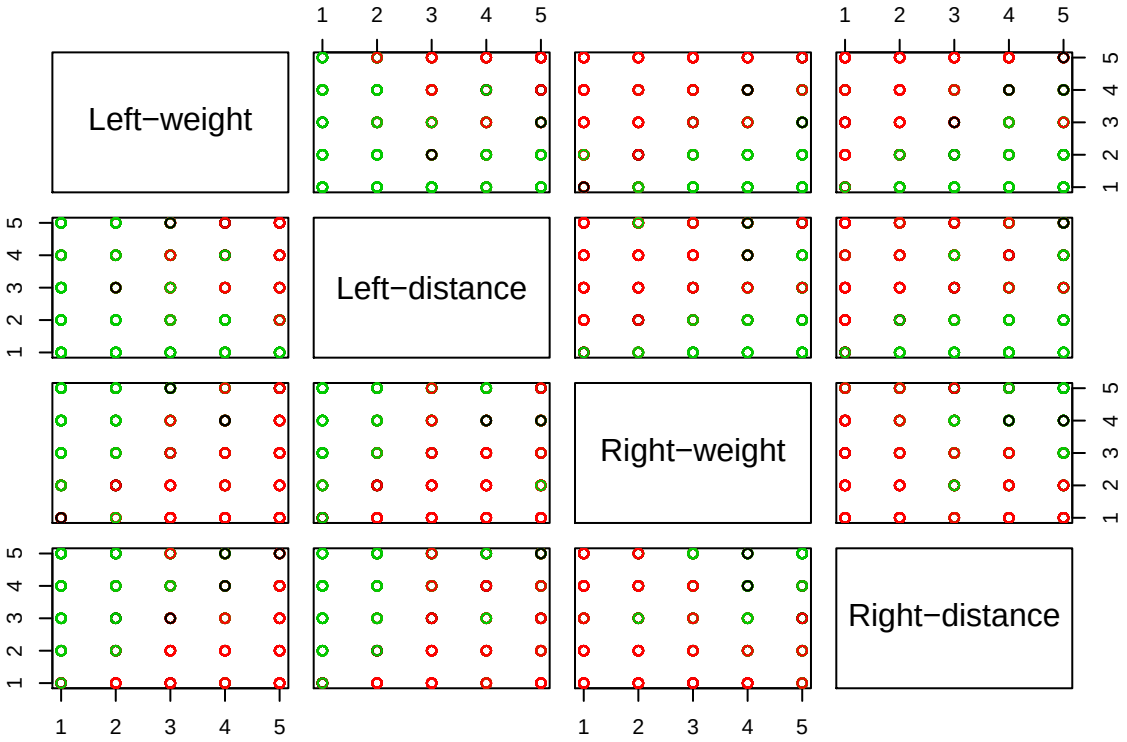


Como puede verse en este histograma, todas ellas contienen la misma cantidad de cada uno de los posibles valores.

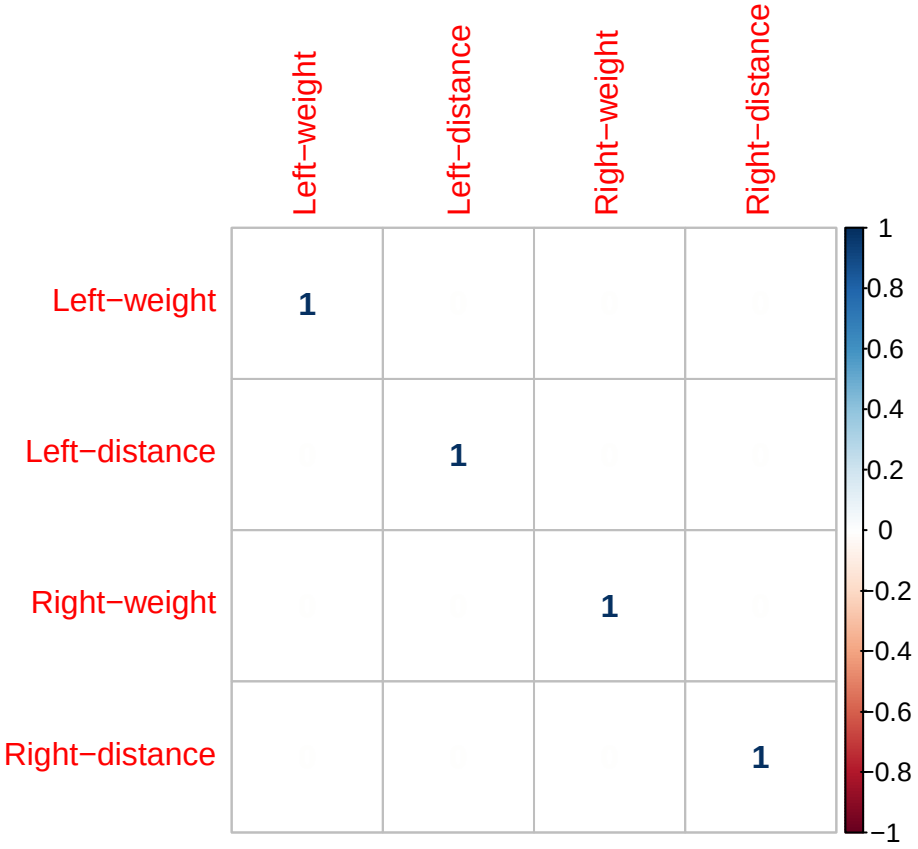
Observando la información mostrada hasta el momento, las variables de entrada parecen seguir una distribución discreta uniforme contenida en el intervalo [1,5]. Debido a que las variables no siguen una distribución normal, es posible que LDA y QDA no ofrezcan buenos resultados, ya que suponen que éstos están normalmente distribuidos.

El siguiente paso en el análisis de las variables será averiguar si existe alguna relación entre ellas, lo que puede observarse la siguiente gráfica. Además, cada etiqueta se muestra de un color para intentar inferir alguna relación, aunque esto no se aprecia demasido, por lo que será analizado porteriormente. En ella se aprecia que éstas no parecen tener relación alguna entre ellas.

```
# Distribución de variables dos a dos.
plot(balance[,-5],col=balance$`Balance-scale`)
```

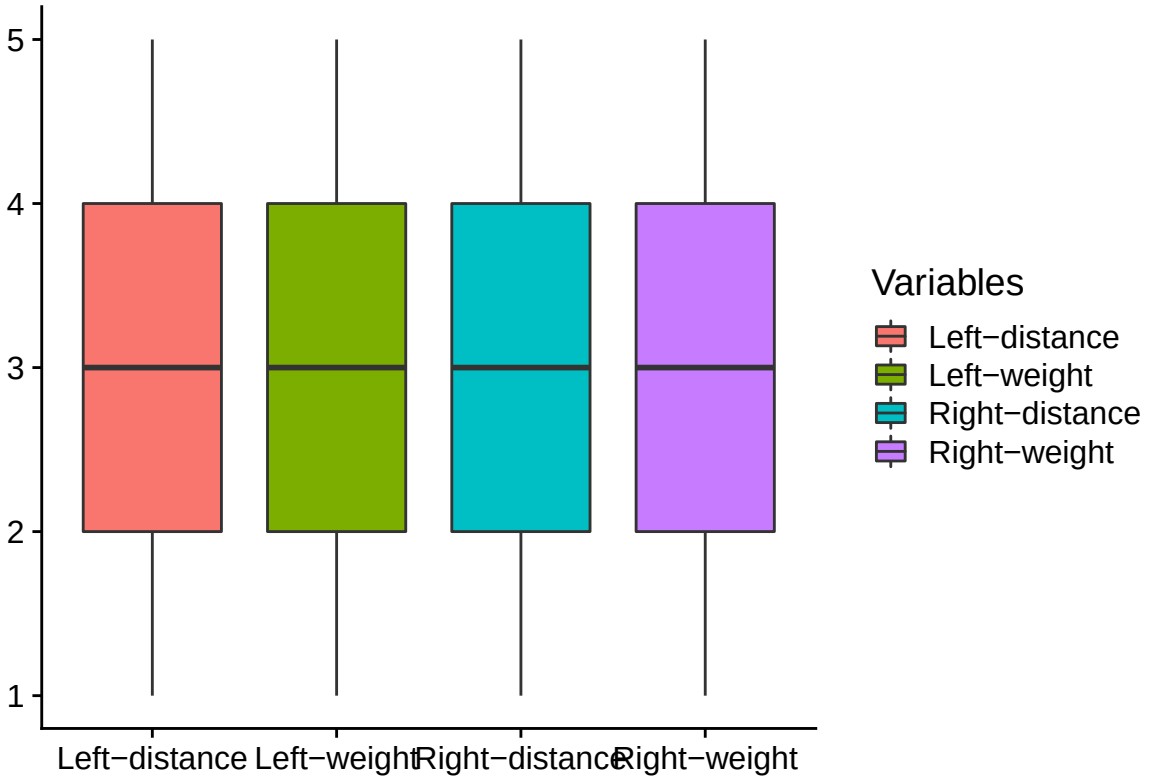


A continuación, para corroborar esta hipótesis muestro la matriz de varianzas covarianzas asociadas a ellas. Efectivamente, no tienen relación alguna entre sí.



Por último, al disponer de pocas variables es sencillo observar mediante un diagrama de caja si existe ruido

en alguna de las variables. A continuación se muestra dicho diagrama para cada variable de entrada con respecto a la variable de salida y puede observarse que no existen outliers en el conjunto. Además, puede observarse de nuevo que los datos se encuentran centrados y escalados.

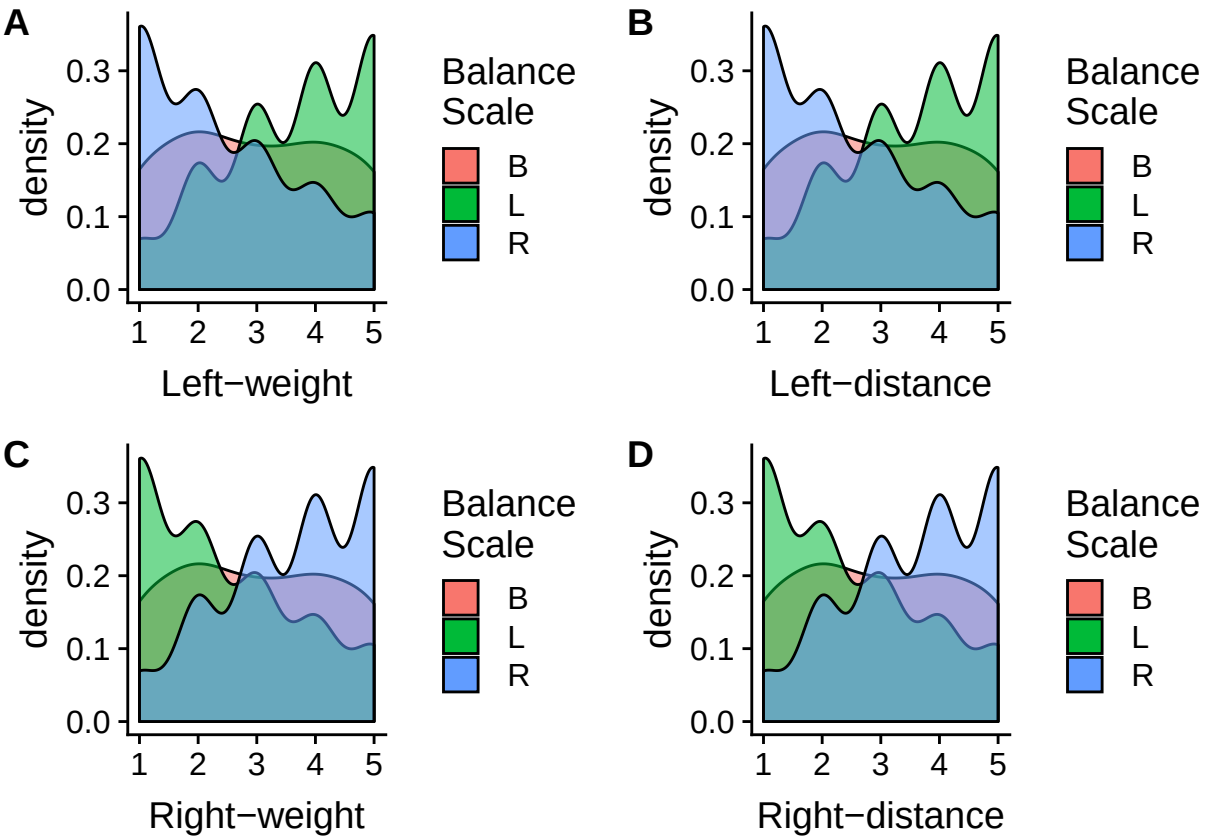


Por otra parte, al mostrar los valores estadísticos, también se observa que se trata un problema multicase con tres clases, donde una de ellas es notablemente inferior en número. Este desbalanceo en las clases puede hacer que algoritmos como el K-NN o el LDA no obtentan resultados demasiado buenos. A continuación se muestra el porcentaje correspondiente a cada una de las posibles etiquetas.

```
# Porcentaje de ejemplos con cada etiqueta
round(prop.table(table(balance$`Balance-scale`)) * 100, digits = 1)
```

```
##
##    B    L    R
##  7.7 46.2 46.2
```

Por último, se analizan las distribuciones de estas variables en función de la clase a la que pertenecen mediante los siguientes diagramas de densidad. De nuevo se ve que todas ellas se distribuyen de la misma manera. Además se observa que la clase balanceada (B) presenta valores más uniformes de todas las variables, mientras que las que representan desbalanceo se encuentran sesgadas a derecha o izquierda.



Regresión: Baseball dataset.

El conjunto asignado para el problema de regresión se llama **baseball** y consiste en un conjunto de variables relacionadas con jugadores de baseball con las que se pretende predecir su salario. Este conjunto consta de 17 variables, incluyendo la de salida. Dos de estas variables son reales, mientras que el resto son enteras. Además, no existen variables ausentes.

```
str(baseball)
```

```
## 'data.frame':    336 obs. of  17 variables:
##  $ Batting_average         : num  0.264 0.251 0.224 0.206 0.238 0.115 0.307 0.187 0.266 0.235 ...
##  $ On_base_percentage      : num  0.318 0.338 0.274 0.262 0.272 0.148 0.405 0.281 0.359 0.353 ...
##  $ Runs                    : int  24 101 28 14 53 0 98 38 115 39 ...
##  $ Hits                    : int  48 141 94 51 106 3 167 50 152 67 ...
##  $ Doubles                 : int  7 35 21 18 18 1 35 9 32 10 ...
##  $ Triples                 : int  0 3 1 1 3 0 1 2 1 0 ...
##  $ HomeRuns                : int  1 32 1 1 15 0 14 15 44 11 ...
##  $ Runs_batted_in          : int  22 105 44 28 59 2 52 37 122 33 ...
##  $ Walks                   : int  15 71 27 17 22 1 84 32 78 48 ...
##  $ Strike_Outs             : int  18 104 54 26 107 6 72 98 152 92 ...
##  $ Stolen_bases            : int  0 34 2 0 14 0 0 0 26 14 ...
##  $ Errors                  : int  7 6 7 3 7 3 15 9 9 3 ...
##  $ Free_agency_eligibility: int  0 0 1 1 0 0 0 0 1 1 ...
##  $ Free_agent              : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ Arbitration_eligibility: int  0 1 0 0 0 0 0 0 0 0 ...
##  $ Arbitration             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Salary                  : int  160 2700 550 300 230 109 560 142 4300 3600 ...
```

```
anyNA(baseball)
```

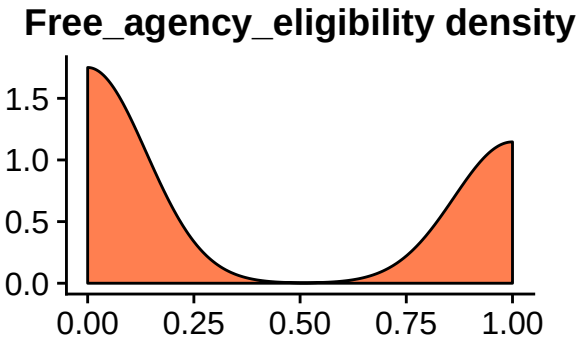
```
## [1] FALSE
```

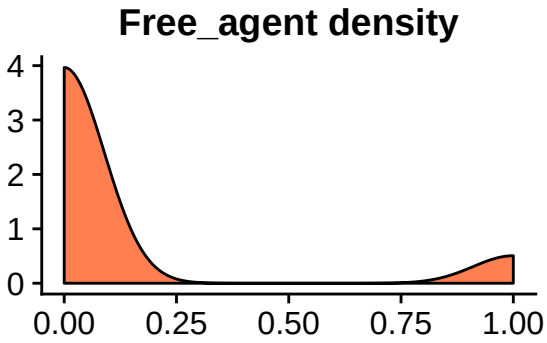
En primer lugar, se muestran distintos valores estadísticos para tener una idea inicial de cómo se encuentran distribuidas las variables. Se observa que las variables no se encuentran escaladas, ya que no pertenecen todas al mismo intervalo numérico. Además algunas de ellas son binarias (*Arbitration_eligibility*,*Arbitration*,*Free_agent*,*Free_agency_eligibility*) y parecen estár sesgadas a la izquierda, es decir, presentan más 0's que 1's.

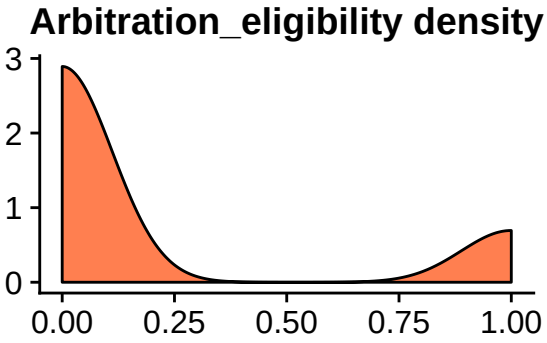
```
summary(baseball)
```

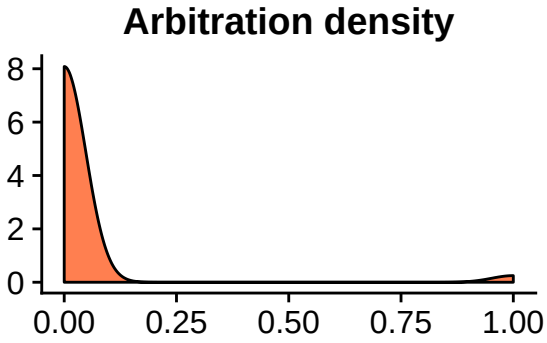
```
##  Batting_average  On_base_percentage      Runs             Hits
##  Min.   :0.0630   Min.   :0.0630     Min.   :  0.00   Min.   :  1.00
##  1st Qu.:0.2380   1st Qu.:0.2970     1st Qu.: 22.00   1st Qu.: 51.00
##  Median :0.2600   Median :0.3225     Median : 41.00   Median : 89.50
##  Mean   :0.2578   Mean   :0.3240     Mean   : 46.62   Mean   : 92.63
##  3rd Qu.:0.2810   3rd Qu.:0.3540     3rd Qu.: 69.00   3rd Qu.:135.25
##  Max.   :0.4570   Max.   :0.4860     Max.   :133.00   Max.   :216.00
##     Doubles         Triples          HomeRuns      Runs_batted_in
##  Min.   : 0.00   Min.   : 0.000   Min.   : 0.000   Min.   :  0.00
##  1st Qu.: 9.00   1st Qu.: 0.000   1st Qu.: 2.000   1st Qu.: 20.75
##  Median :15.00   Median : 2.000   Median : 6.000   Median : 39.00
##  Mean   :16.66   Mean   : 2.327   Mean   : 9.089   Mean   : 43.98
##  3rd Qu.:23.00   3rd Qu.: 3.000   3rd Qu.:15.000   3rd Qu.: 66.00
##  Max.   :49.00   Max.   :15.000   Max.   :44.000   Max.   :133.00
##      Walks         Strike_Outs      Stolen_bases        Errors
##  Min.   :  0.00   Min.   :  1.00   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 15.00   1st Qu.: 31.00   1st Qu.: 1.000   1st Qu.: 2.750
##  Median : 30.00   Median : 49.00   Median : 4.000   Median : 5.000
##  Mean   : 34.98   Mean   : 56.48   Mean   : 8.202   Mean   : 6.741
##  3rd Qu.: 49.00   3rd Qu.: 77.25   3rd Qu.:11.000   3rd Qu.: 9.000
##  Max.   :138.00   Max.   :175.00   Max.   :76.000   Max.   :31.000
##  Free_agency_eligibility   Free_agent     Arbitration_eligibility
##  Min.   :0.0000          Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000          1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000          Median :0.0000   Median :0.0000
##  Mean   :0.3958          Mean   :0.1131   Mean   :0.1935
##  3rd Qu.:1.0000          3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000          Max.   :1.0000   Max.   :1.0000
##   Arbitration          Salary
##  Min.   :0.00000   Min.   : 109
##  1st Qu.:0.00000   1st Qu.: 230
##  Median :0.00000   Median : 745
##  Mean   :0.02976   Mean   :1252
##  3rd Qu.:0.00000   3rd Qu.:2154
##  Max.   :1.00000   Max.   :6100
```

El siguiente paso será visualizar las variables. En primer lugar, se muestran las distribuciones de las variables binarias mostrando su densidad. En el siguiente gráfico puede corroborarse la hipótesis que la mayoría contienen valores iguales a cero y, por lo tanto, no se encuentran centrados.

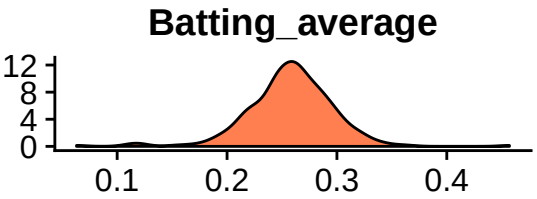


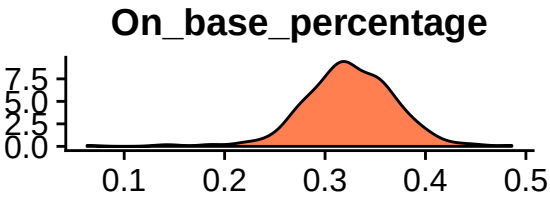


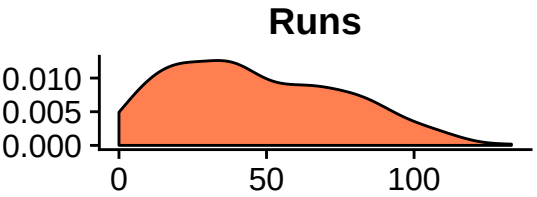


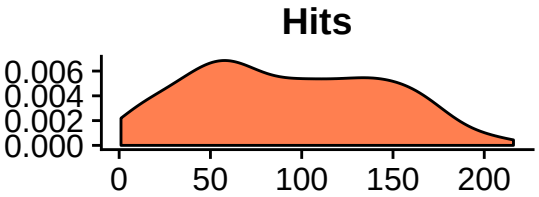


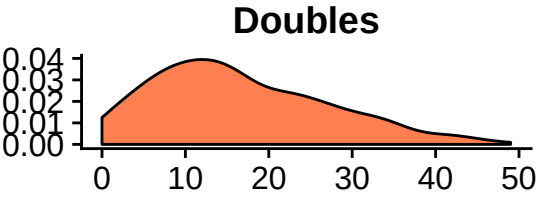
Del mismo modo, las variables no binarias se visualizarán mediante diagramas de densidad con los cuales se tratará de ver sus distribuciones. Puede verse que las dos variables con valores reales, *Batting_average* y *On_base_percentage*, parecen tener una distribución normal. Es resto, a excepción de *Hits* y *Runs*, parecen seguir una distribución normal sesgada a la izquierda.

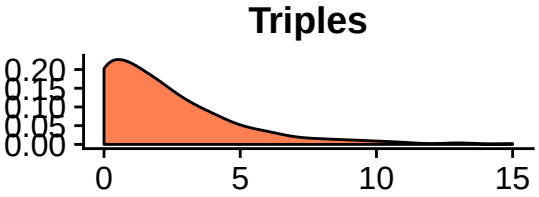


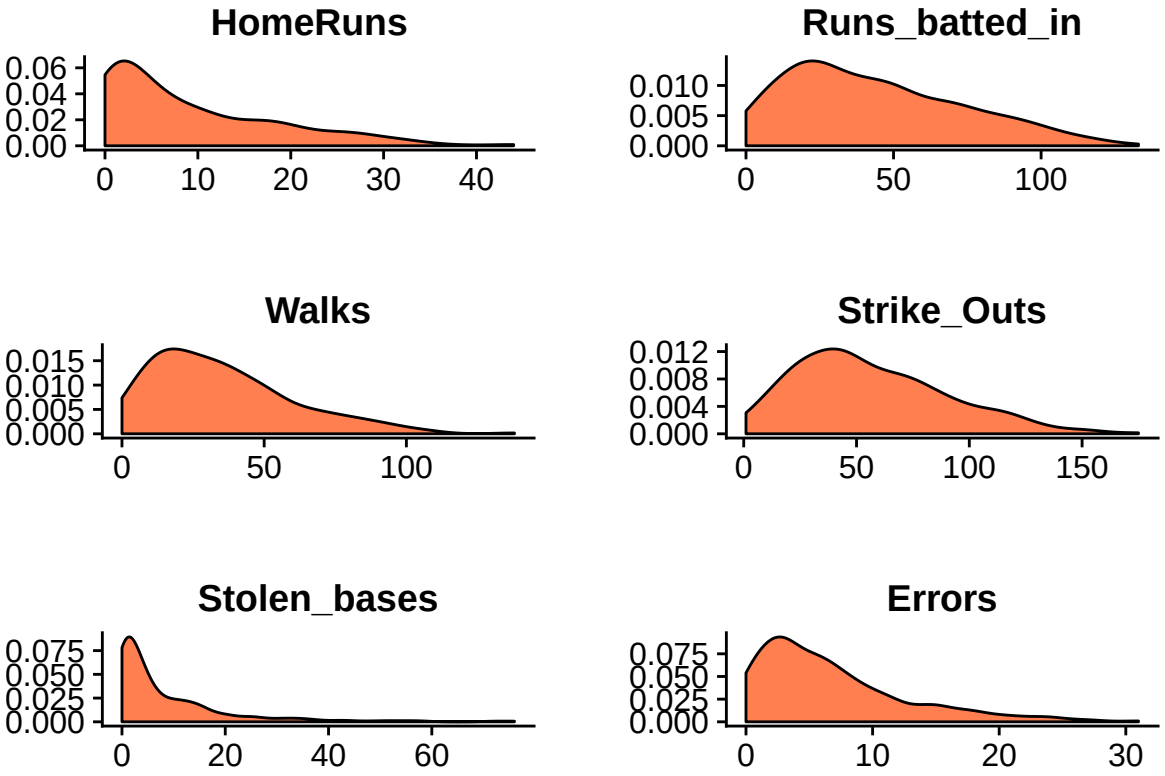






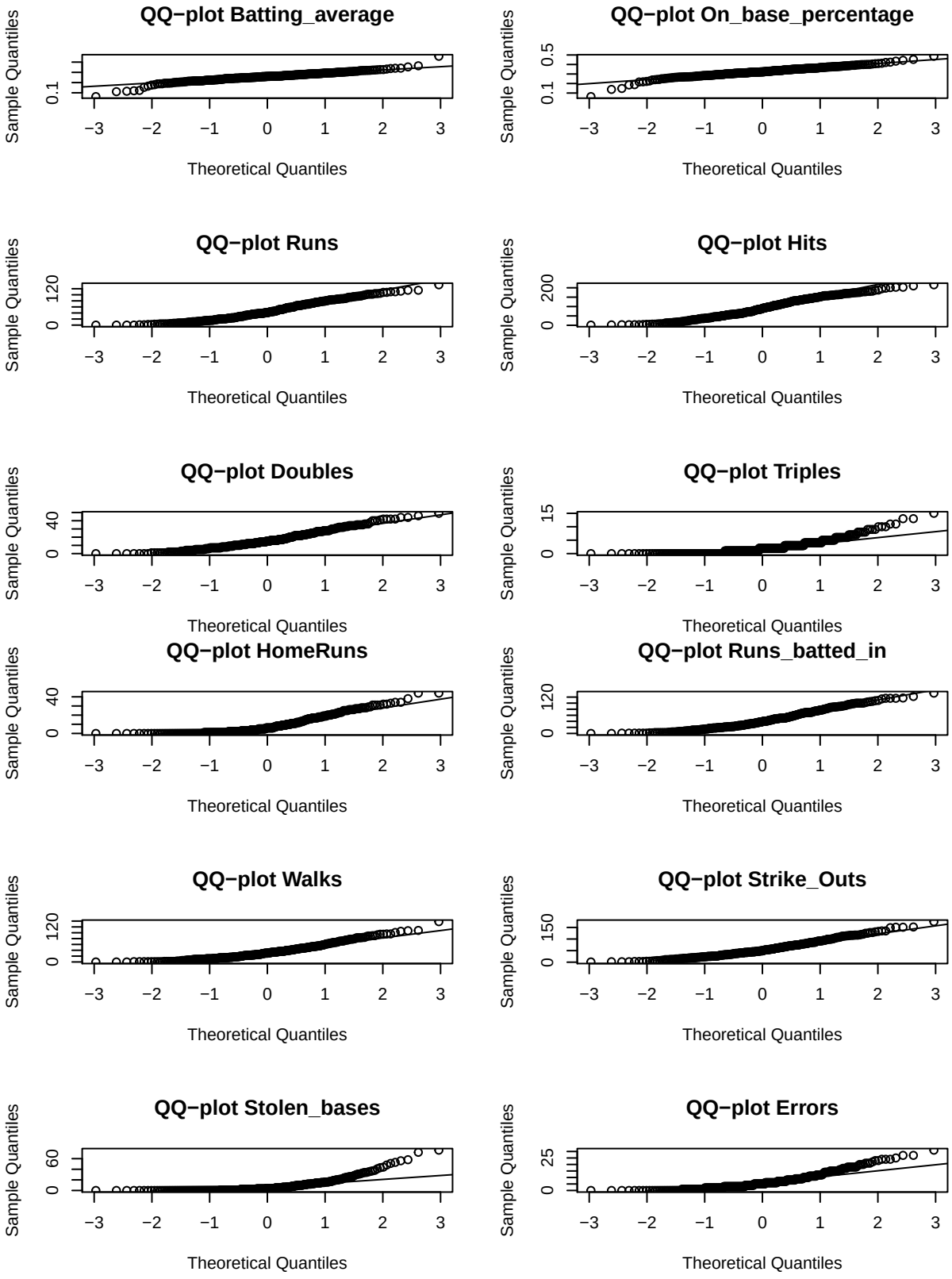






Con el objetivo de conocer si estas variables siguen o no una distribución normal, se utilizan los qqplots y el test de shapiro.

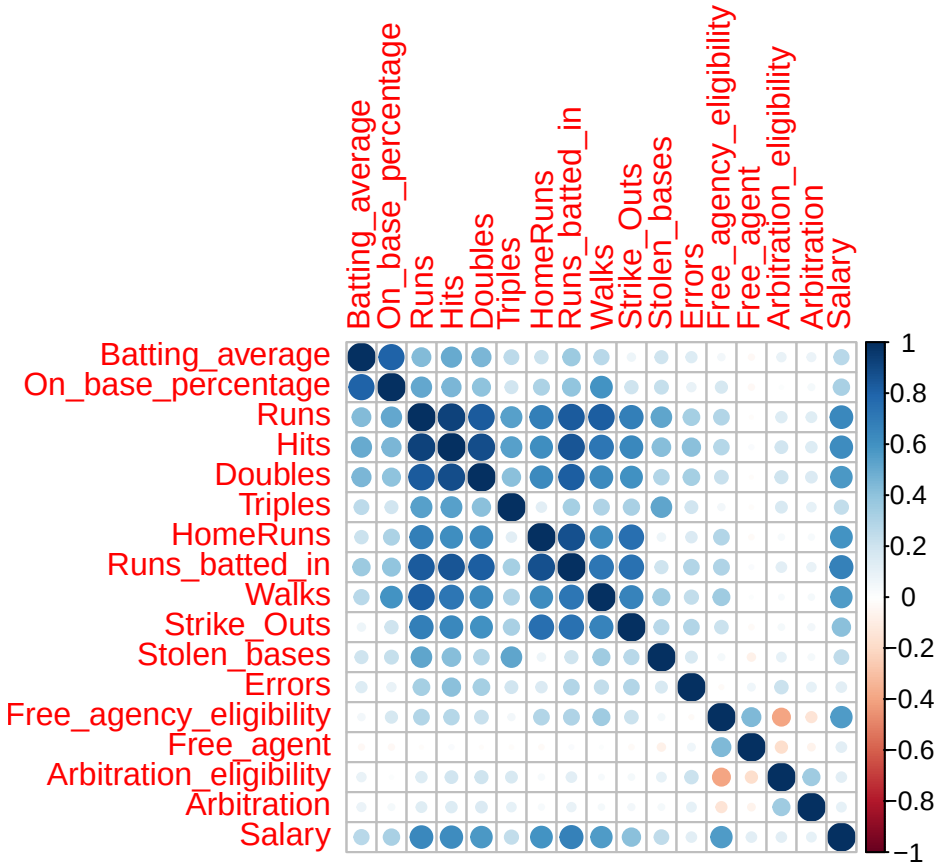
En primer lugar, se muestran los qqplots correspondientes a estas variables, en los que puede verse que éstas se ajustan bastante bien, de hecho la mayoría podrían ser distribuciones normales sesgadas a derecha o izquierda.



Por último se aplica el test de shapiro sobre el conjunto completo de las variables para comprobar si alguna de ellas sigue una distribución normal. Puede observarse que todos los p-valores obtenidos son inferiores a 0.1, por lo que se rechaza la hipótesis nula de que las variables se encuentran igualmente distribuidas.

	P-Valor
Batting_average	0.0e+00
On_base_percentage	1.0e-07
Runs	4.0e-07
Hits	4.3e-06
Doubles	1.0e-07
Triples	0.0e+00
HomeRuns	0.0e+00
Runs_batted_in	0.0e+00
Walks	0.0e+00
Strike_Outs	1.0e-07
Stolen_bases	0.0e+00
Errors	0.0e+00
Free_agency_eligibility	0.0e+00
Free_agent	0.0e+00
Arbitration_eligibility	0.0e+00
Arbitration	0.0e+00

Por último, se analiza la correlación que presetan las variables entre sí. Puede observarse que algunas tienen mucha relación, como *Rus*,*Hits* y *Doubles*. Sin embargo otras no presentan relación alguna con la mayoría de variables, por ejemplo *Free_agent*, *Arbitration_elegibility*, *Errors*, o *Arbitration*, las cuales ni si quiera presentan relación con la variable de salida.



Clasificación

En esta sección, pasamos a abordar el problema de clasificación planteado utilizando tres algoritmos distintos y comparando posteriormente los resultados obtenidos. Para ello se aplicarán tres algoritmos de clasificación diferentes: K-NN, LDA y QDA.

Como ya se ha explicado en la sección previa, las variables de entrada se encuentran ya escaladas y centradas. Es decir, todas se encuentran comprendidas en el intervalo [1,5] y se encuentran centradas en la media, 3. Por tanto, con el objetivo de reducir la complejidad de estos algoritmos y puesto que no es necesario, no se aplica ninguna de estas dos técnicas en el preprocesado de los datos.

K-NN

En primer lugar, utilizamos el algoritmo K-NN para intentar resolver este problema, tratando de en contrar el número de vecimos óptimo para obtener el mejor clasificador posible. Para ello utilizamos el siguiente comando, que permite buscar el mejor número de vecinos cercanos para el conjunto de entrenamiento, intentando obtener el mayor Accuracy y Kappa posible.

```
# Aplico knn sobre el conjunto de entrenamiento
knnModel <- train(x = balance_train, y = balance_train_labels, method = "knn",
                  metric="Accuracy")
knnModel
```

```
## k-Nearest Neighbors
##
## 499 samples
##   4 predictor
##   3 classes: ' B', ' L', ' R'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 499, 499, 499, 499, 499, 499, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.8492345  0.7270488
##   7  0.8695223  0.7609341
##   9  0.8850881  0.7877414
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

Como puede observarse, el algoritmo decide que el mejor número de vecinos a tener en cuenta es 9, obteniendo un accuracy de 0.88 y un Kappa de 0.79 en el conjunto de entrenamiento. Veamos qué ocurre en el conjunto de prueba.

```
# Obtengo precisión y matriz de confusión en el test
knnPred <- predict(knnModel, newdata = balance_test)
postResample(pred = knnPred, obs = balance_test_labels)
```

```
##  Accuracy     Kappa
## 0.8880000 0.7926786
```

```
table(knnPred,balance_test_labels)
```

```
##        balance_test_labels
```

```
## knnPred  B  L  R
##       B  0  0  0
##       L  3 58  1
##       R  7  3 53
```

La presición obtenida en el conjunto de prueba es prácticamente la misma 0.88, sin embargo podemos observar que no se clasifica correctamente ningún elemento de la clase minoritaria.

Decido buscar el número óptimo de vecinos atendiendo al resultado obtenido en el conjunto de prueba en lugar de en el de entrenamiento. Para ello, en lugar de utilizar esta función proporcionada por R, lo hago a mano utilizando validación cruzada.

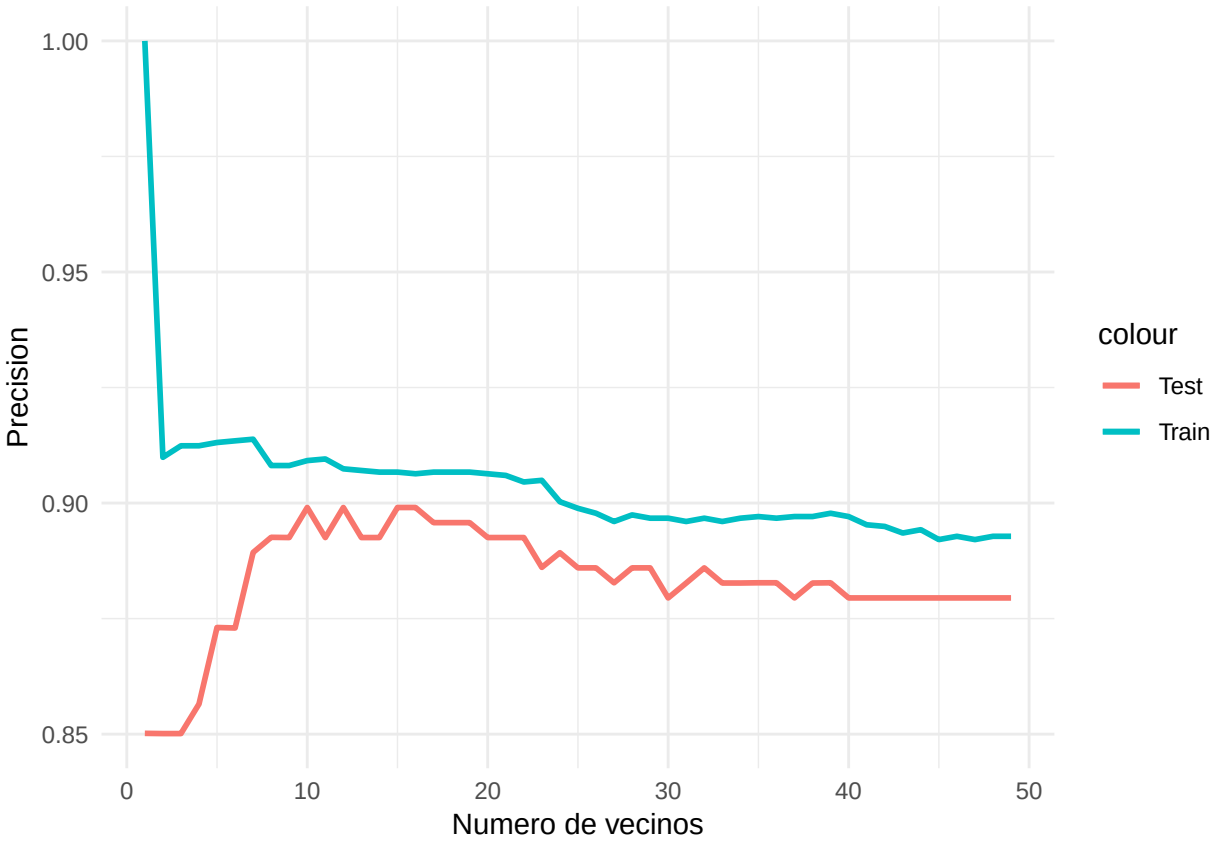
```
k <- 1
result <- c(-1,-1)
accuTrain <- c()
accuTest <- c()
while (k < 50){
  knnAccuracyTrain <-mean(sapply(1:5,cross_val_knn,k,"train"))
  knnAccuracyTest <-mean(sapply(1:5,cross_val_knn,k,"test"))

  accuTrain <- c(accuTrain,knnAccuracyTrain)
  accuTest <- c(accuTest,knnAccuracyTest)

  if (result[2] < knnAccuracyTest){
    result <- c(k,knnAccuracyTest)
  }
  k <- k + 1
}
result
```

```
## [1] 10.0000000  0.8990393
```

En la siguiente gráfica se muestra la evolución sufrida por la precisión a medida que el número de vecinos utilizados para el algoritmo K-NN aumenta. Puede verse que si se utilizan muy pocos vecinos se sufre un gran sobreaprendizaje, aunque a medida que el número de vecinos aumenta este factor es menos significativo. Además se observa que la precisión en el conjunto de prueba no supera el 0.9 en ningún momento y que el número mínimo de vecinos para logar esto es 10. Por tanto, consideramos que este es el número óptimo de vecinos a considerar.



Por último se observa que ningún elemento de la clase minoritaria es clasificado correctamente.

```
##       balance_test_labels
## yprime  B  L  R
##      B  0  0  0
##      L  3 58  1
##      R  7  3 53
```

LDA

En este apartado se utilizará el algoritmo LDA (*Linear Discriminant Analysis*) para afrontar el problema dado mediante la aplicación de una función lineal. Este algoritmo puede no ofrecer muy bueno resultados en este conjunto, ya que asume que las variables de entrada siguien una distribución normal y, como ya se ha visto, esto no ocurre.

```
# Aplico LDA
lda.fit <- lda(`Balance-scale`~.,data=balance_train)
lda.pred <- predict(lda.fit,balance_test[,-5])

table(lda.pred$class,balance_test$`Balance-scale`)
```

```
##
##       B  L  R
##    B  0  0  0
##    L  4 58  2
##    R  6  3 52
```

```r
mean(lda.pred$class==balance_test$`Balance-scale`)
```

```
## [1] 0.88
```

Como puede observarse, este algortimo no es capaz de clasificar correctamente la clase minoritaria y no ofrece mejores resultados que el K-NN expuesto en el apartado previo.

Por último, se muestran los accuracies medios obtenidos en el conjuntro train y test tras la aplicación de validación cruzada sobre las particiones proporcionadas para la práctica.

```r
ldaAccuracyTrain
```

```
## [1] 0.8817698
```

```r
ldaAccuracyTest
```

```
## [1] 0.8469452
```

QDA

Por último, aplicamos el algortimo QDA (*Quadratic Discriminant Analysis*) para clasificación. Éste funciona de forma muy similar a LDA, pero estimando las diferencias de varianza para cada clase.

```r
# Aplico QDA
qda.fit <- qda(`Balance-scale`~.,data=balance_train)
qda.pred <- predict(qda.fit,balance_test[,-5])

table(qda.pred$class,balance_test$`Balance-scale`)
```

```
##
##       B  L  R
##    B  7  3  2
##    L  2 56  1
##    R  1  2 51
```

```r
mean(qda.pred$class==balance_test$`Balance-scale`)
```

```
## [1] 0.912
```

Este algoritmo sí presenta resultados competitivos, ya que es capaz de clasificar correctamente más de la mitad de los elementos de la clase minoritaria, con una presición superior al 0.9.

Por último aplico validación cruzada sobre las particiones dadas para obtener el accuracie medio en los conjuntos de prueba y entrenamiento y tener así una idea más objetiva del correcto funcionamiento del algoritmo.

```r
qdaAccuracyTrain
```

```
## [1] 0.918091
```

```r
qdaAccuracyTest
```

```
## [1] 0.9022651
```

Comparativa

En esta sección se realizará una comparativa de los tres algoritmos de clasificación empleados. Para ello, se utilizarán las tablas de resultados proporcionadas, donde se reescribirán los resultados obtenidos a lo largo de esta sección.

En primer lugar se aplica el test de Wilcoxon sobre cada par de algoritmos para determinar si existen diferencias significativas entre ellos. En la siguiente tabla se muestran los resultados obtenidos. Como puede verse, no existen diferencias significativas entre estos tres algoritmos, aunque una confianza del 81% en que K-NN y QDA sean distintos, sin embargo esto no se considera suficientes para concluir que son distintos.

	P-valor	Confianza (%)
K-NN vs LDA	0.5458755	45.41245
K-NN vs QDA	0.1893482	81.06518
LDA vs QDA	0.7285061	27.14939

Para analizar las posibles diferencias entre estos tres algoritmos se realizarán comparativas múltiples entre ellos. En primer lugar, aplicará el test de Friedman y, en segundo lugar, post-hoc Holm.

Aplicando el test de Friedman se obtiene un p-value de 0.7, por lo que estos algoritmos no parecen presentar diferencias siguificativas entre ellos.

```
# Aplico el test de Friedman sobre el conjunto de prueba
test_friedman <- friedman.test(as.matrix(tablatst))
test_friedman
```

```
##
##  Friedman rank sum test
##
## data:  as.matrix(tablatst)
## Friedman chi-squared = 0.7, df = 2, p-value = 0.7047
```

Finalmente, estas hipótesis son corroboradas por postHolm, ya que no se obtiene ningún p-value demasiado bajo.

```
# Aplico postHolm sobre el conjunto de prueba
tam <- dim(tablatst)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatst), groups, p.adjust = "holm", paired = TRUE)
```

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatst) and groups
##
##   1    2
## 2 1.00 -
## 3 0.57 1.00
##
## P value adjustment method: holm
```

Por otra parte, se realizan estos mismos tests sobre el conjunto de entrenamiento.

En primer lugar, al aplicar Wilcoxon se obtienen los siguientes resultados. Como puede verse existe una confianza del 84% de que LDA y QDA sean distintos, superior al 72% que presentan K-NN y QDA.

	P-valor	Confianza (%)
K-NN vs LDA	0.7011814	29.88186
K-NN vs QDA	0.2773552	72.26448
LDA vs QDA	0.1536465	84.63535

Sin embargo, aplicando el test de Fiedman se obtiene un p-value igual a 0.522, lo que no es muy bajo, por lo que no parecen existir diferencias significativas entre los algoritmos.

```
# Aplico el test de Friedman sobre el conjunto de entrenamiento
test_friedman <- friedman.test(as.matrix(tablatra))
test_friedman
```

```
##
##  Friedman rank sum test
##
## data:  as.matrix(tablatra)
## Friedman chi-squared = 1.3, df = 2, p-value = 0.522
```

Por último se aplica post-Holm y ninguno de los p-values obtenidos son muy bajos. Por tanto se concluye que en el conjunto de entrenamiento tampoco existen diferencias significativas entre los algoritmos.

```
# Aplico postHolm sobre el conjunto de entrenamiento
tam <- dim(tablatra)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatra), groups, p.adjust = "holm", paired = TRUE)
```

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatra) and groups
##
##   1    2
## 2 0.65 -
## 3 0.59 0.53
##
## P value adjustment method: holm
```

Regresión

En esta sección se aborda el problema de regresión asociado al conjunto **basebal**. Se trata de predecir el salario de un jugador de este deporte en función de las variables analizadas en el primer apartado.

Regresión lineal simple

Para seleccionar las cinco variables más influyentes utilizamos el coeficiente de correlación de Pearson.

```
# Calculo matriz de correlaciones
corMatrix <- cor(baseball,method="pearson")
corMatrix <- round(corMatrix,2)

# Obtengo los valores asociados al salario
# Busco ver qué valores tienen más relación con la variable de salida para
# seleccionar los cinco mejores candidatos.
numCols <- length(names(baseball))
min <- length((corMatrix))- numCols + 2
salaryCor <- corMatrix[min :length(corMatrix)-1]
highCorrIndex <- which( salaryCor > 0.57 )
highCorrIndex
```

```
## [1] 3 4 5 7 8
```

```
names(baseball)[highCorrIndex]
```

```
## [1] "Runs"           "Hits"           "Doubles"        "HomeRuns"
## [5] "Runs_batted_in"
```

	Hits	Doubles	Runs	Runs_batted_in	HomeRuns
R2 ajustados	0.3916246	0.334153	0.4170694	0.4483109	0.3486897

Puede verse que los R^2 son bastante malos, por lo que decido pasar a aplicar regresión lineal múltiple.

Regresión lineal múltiple

Buscando una mejora signifativa se aplicará regresión lineal múltiple sobre el conjunto completo de variables.

```
## [1] 0.6892459
```

Efectivamente, se obtiene el mejor resultado hasta el momento.

A continuación, pruebo interacciones y combinaciones no lineales entre las variables. En primer lugar, las interacciones entre las cinco variables más relevantes, en segundo lugar, aplico el cuadrado sobre la suma de ellas. Por último, aplico logaritmo sobre el regresor *Hits*, ya que parece crecer con una forma similar a la que sigue esta función.

```
# Busco interacciones y combinariones no lineales
fit7 = lm(Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns),data=baseball)
summary(fit7)$adj.r.squared
```

```
## [1] 0.7258317
```

```
fit8 = lm(Salary~. + I(Hits + Doubles + Runs + Runs_batted_in + HomeRuns)^2,data=baseball)
summary(fit8)$adj.r.squared
```

```
## [1] 0.6892459
```

```
fit9 = lm(Salary~. + I(log(Hits)) ,data=baseball)
summary(fit9)$adj.r.squared
```

```
## [1] 0.6905094
```

Como puede observarse, el mejor resultado se ha obtenido al aplicar interacciones entre los cinco regresores más representativos.

Llegados a este punto, decido utilizar la matriz de correlación calculada en el apartado anterior para eliminar aquellos regresores que tengan poca relación con la variable de salida y poder así reducir la complejidad.

```
# Busco índices de los regresores menos correlados
# con la variable de salida
lowCorrIndex <- which( salaryCor < 0.15 )
names(baseball)[lowCorrIndex]
```

```
## [1] "Errors"                  "Free_agent"
## [3] "Arbitration_eligibility" "Arbitration"
```

En primer lugar pruebo a eliminar estas variables del conjunto completo, pero observamos que se empeoran notablemente los resultados.

```r
# Pruebo a eliminar estos regresores de los conjuntos probados
fit10 = lm(Salary~.-Arbitration-Free_agent-Errors-Arbitration_eligibility,data=baseball)
summary(fit10)$adj.r.squared
```

```
## [1] 0.6362775
```

Además, observamos que elimnando sólo tres de ellas se obtiene un mejor resultado.

```r
fit11 = lm(Salary~.-Arbitration-Free_agent-Errors,data=baseball)
summary(fit10)$adj.r.squared
```

```
## [1] 0.6362775
```

En segundo lugar, pruebo a eliminarlas del mejor modelo obtenido anteriormente y observo que si elimino las tres seleccionadas en el apartado anterior factor apenas se ve afectado.

```r
fit12 = lm(Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
           -Arbitration-Free_agent-Errors-Arbitration_eligibility,data=baseball)
summary(fit12)$adj.r.squared
```

```
## [1] 0.6750753
```

```r
fit13 = lm(Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
           -Arbitration-Free_agent-Errors,data=baseball)
summary(fit13)$adj.r.squared
```

```
## [1] 0.7196426
```

```r
fit14 = lm(Salary~.  + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
          + I(Batting_average + On_base_percentage + Runs + Hits + Doubles
          + Triples + HomeRuns + Runs_batted_in + Walks + Strike_Outs + Stolen_bases +
          Free_agency_eligibility )^2, data=baseball)
summary(fit14)$adj.r.squared
```

```
## [1] 0.7258317
```

No supone una mejora representativa teniendo en cuenta la complejidad añadida. Por tanto, el mejor resultado obtenido, atendiendo tanto a la complejidad del modelo como a los resultados obtenidos ha resultado al utilizar las interacciones entre los cinco regresores más relacionados con el salario, junto con el resto de variables a excepción de *Arbitration*,*Free_agent* y *Errors*.

Por último aplico validación cruzada sobre el conjunto seleccionado como el mejor en los pasos previos.

```r
# Aplico validación cruzada
lmMSEtrain<-mean(sapply(1:5,cross_val_lm,
                        Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
                        -Arbitration-Free_agent-Errors,"train"))
lmMSEtest<-mean(sapply(1:5,cross_val_lm,
                       Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
                       -Arbitration-Free_agent-Errors,"test"))
lmMSEtrain
```

```
## [1] 364564.2
```

```r
lmMSEtest
```

```
## [1] 685171.4
```

Observamos que existe sobreaprendizaje, ya que el MSE en el conjunto de prueba es prácticamente el doble que en el de entrenamiento. Probamos con otro modelo menos acertado para comprobar hasta dónde llega el sobreaprendizaje y obtener así el mejor regresor lineal posible.

A continuación aplico valicación cruzada sobre el conjunto completo y observo que el sobreaprendizaje se ve notablemente reducido y los resultados mejoran notablemente

```
lmMSEtrain<-mean(sapply(1:5,cross_val_lm,Salary~.,"train"))
lmMSEtest<-mean(sapply(1:5,cross_val_lm,Salary~.,"test"))
lmMSEtrain
```

```
## [1] 448159.2
```

```
lmMSEtest
```

```
## [1] 536675.5
```

Añado las interacciones con las cinco variables más correladas con el salario y observo que el sobreaprendizaje reaparece, por lo que decido prescindir de ellas.

```
lmMSEtrain<-mean(sapply(1:5,cross_val_lm,
                        Salary~.+ (Hits * Doubles * Runs * Runs_batted_in * HomeRuns),"train"))
lmMSEtest<-mean(sapply(1:5,cross_val_lm,
                       Salary~.+ (Hits * Doubles * Runs * Runs_batted_in * HomeRuns),"test"))
lmMSEtrain
```

```
## [1] 351836.8
```

```
lmMSEtest
```

```
## [1] 662117.2
```

Por último elimino las tres variables que se consideraron menos significativas basándonos en la correlación con la variable de salida y se obtienen resultados aún mejores. Ésta será la combinación escogida como la mejor, basándonos en el MSE tanto en el conjunto de entrenamiendo como en el de prueba.

```
lmMSEtrain<-mean(sapply(1:5,cross_val_lm,Salary~.-Arbitration-Free_agent-Errors,"train"))
lmMSEtest<-mean(sapply(1:5,cross_val_lm,Salary~.-Arbitration-Free_agent-Errors,"test"))
lmMSEtrain
```

```
## [1] 461989.9
```

```
lmMSEtest
```

```
## [1] 530286.8
```

Podemos concluir que el mejor resultado obtenido mediante regresión lineal resulta al aplicar este método sobre todos los regresores menos tres de los menos corelados con la variable de salida. Los resultados obtenidos tanto en el conjunto de prueba como en el de entrenamiento pueden verse en la siguiente tabla.

	Train	Test
MSE	461989.9	530286.8

K-NN

En este apartado se utiliza el algoritmo K-NN para regresión. Para ello, se busca el número óptimo de vecinos a tener en cuenta tratando de minimizar el MSE en el conjunto de prueba. Además, el MSE se calculará como la media obtenida al aplicar validación cruzada sobre los conjuntos proporcionados para le realización de la práctica.

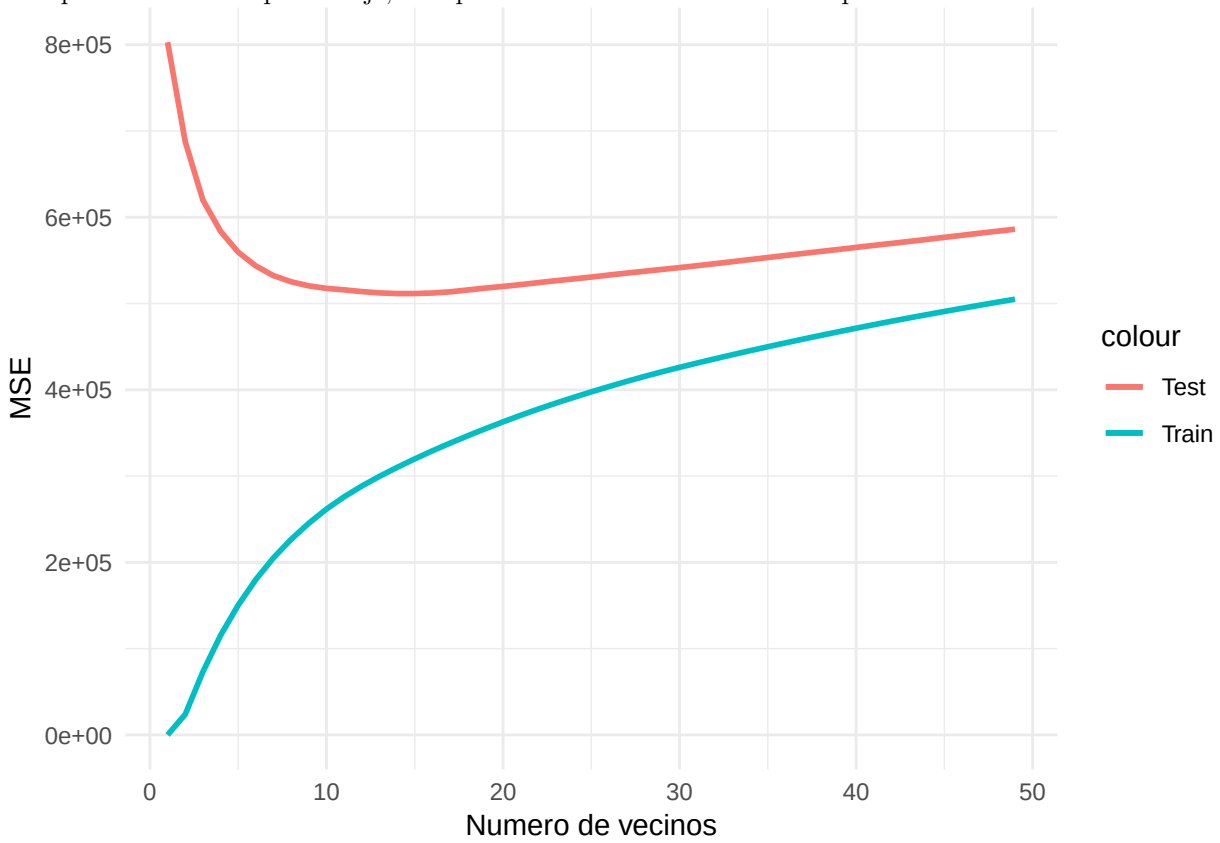
```
# Busco mejor número de vecinos para obtener el menor MSE posible
k <- 1
result <- c(-1, 9999999)
mseTrain <- c()
```

```
mseTest <- c()
accuTrain <- c()
accuTest <- c()
while (k < 50){
  knnMSETrain <-mean(sapply(1:5,cross_val_knn,
                            Salary ~ .-Arbitration-Free_agent-Errors,k,"train"))
  knnMSETest <-mean(sapply(1:5,cross_val_knn,
                           Salary ~ .-Arbitration-Free_agent-Errors,k,"test"))
  accuTrain <- c(accuTrain,knnMSETrain)
  accuTest <- c(accuTest,knnMSETest)

  if ( knnMSETest < result[2]){
    result <- c(k,knnMSETest)
  }
  k <- k + 1
}
result
```

```
## [1]     14 511567
```

A continuación se muestran gráficamente los MSE obtenidos a medida que el número de vecinos utilizados aumenta. Puede verse que el error es creciente en el conjunto de entrenamiento y que siempre existe sobreaprendizaje, aunque éste se estabiliza a medida que el número de vecinos aumenta.



Finalmente, se muestran los resultados obtenidos tras la aplicación de validación cruzada en los conjuntos de entrenamiento y prueba.

	Train	Test
MSE	310093	511567

Comparativa.

En este apartado se realizará una comparativa general de los dos algoritmos de regresión múltiple utilizados en los apartados previos: regresión lineal y knn. Para ello, se obtienen los resultados utilizando el conjunto completo y aplicando validación cruzada sobre las particiones proporcionadas para el conjunto. Además, mediante comparativas múltiples, se compararán dichos algoritmos con un tercero, M5', cuyos resultados se encuentran en las tablas proporcionadas para la práctica.

Para comenzar, se compararán los resultados obtenidos en regresión lineal y knn para el conjunto completo. Para ello, en primer lugar, se obtienen los resultados de estos dos algoritmos utilizando el conjunto completo. Éstos pueden verse en la siguiente tabla.

	Train	Test
LM-MSE	448159.2	536675.5
KNN-MSE	505007.3	566112.6

En segundo lugar, se aplica el test de Wilcoxon para comparar dichos algoritmos.

	Rmenos	Rmas	pvalue
LM(R+) vs KNN(R-)	93	78	0.7660294

Como puede observarse, sólo hay un $(1 - 0.766) \times 100 = 23.4\%$ de confianza en que sean distintos, por lo que podemos concluir que no existen diferencias significativas entre estos dos métodos.

A continuación se realizarán comparativas múltiples entre estos dos algoritmos y M5'. Para ello, se aplicará en primer lugar el test de Friedman y, en segundo lugar, post-hoc Holm.

```
##
##  Friedman rank sum test
##
## data:  as.matrix(tablatst)
## Friedman chi-squared = 8.4444, df = 2, p-value = 0.01467
```

Friedman indica, con una confianza del $(1 - 0.01467) * 100 = 98.533\%$, que existen diferencias significativas entre, al menos, un par de algoritmos. Gracias a los resultados obtenidos al aplicar el test de Wilcoxon se sabe que estas diferencias serán del M5' con regresión lineal y/o knn.

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatst) and groups
##
##   1     2
## 2 0.580 -
## 3 0.081 0.108
##
## P value adjustment method: holm
```

Aplicando post-hoc Holm concluimos que estas diferencias significativas son a favor de M5', mientras que los otros dos pueden considerarse equivalentes.

Esta comparativa a sido realizada sobre el conjunto de prueba, sin embargo es recomendable realizarla también en el conjunto de entrenamiento, ya que esto puede ofrecer nueva información. A continuación, se realizará el mismo procedimiento sobre el dicho conjutno.

	Rmenos	Rmas	pvalue
LM(R+) vs KNN(R-)	161	10	0.0003281

En este caso, observamos que existe prácitamente un 100% de confianza en que LM y KNN sean distintos.

```
##
##  Friedman rank sum test
##
## data:  as.matrix(tablatra)
## Friedman chi-squared = 20.333, df = 2, p-value = 3.843e-05
```

Efectivamente, Friedman establece con casi un 100% de confianza que existen diferencias significativas en, al menos un par de los algoritmos comparados. Esto confirma el resultado obtenido con Wilcoxon.

Por último, al aplicar post-hoc Holm se observa que no sólo existen diferencias significativas entre K-NN y LM, como se había visto con Wilcoxon, sino que, además, existen diferencias significativas entre estos y M5'. Por tanto, los tres algoritmos analizados son distintos en el conjutno de entrenamiento.

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatra) and groups
##
##   1      2
## 2 0.0031 -
## 3 0.0032 0.0032
##
## P value adjustment method: holm
```

Apéndice: código.

```
library(class)
library(MASS)
library(ggplot2)
library(cowplot)
require(kknn)
require(caret)
library(tidyr)
library(corrplot)
knitr::opts_chunk$set(echo = TRUE)
set.seed(12345)

# ------------- ------------- ------------- #
# -------------       EDA       ------------- #
# ------------- ------------- ------------- #
```

```r
# Cargo el conjunto de datos y doy nombre a las variables
balance <- read.csv("./balance/balance.dat", header=TRUE, comment.char = "@")
names(balance) <- c("Left-weight", "Left-distance", "Right-weight", "Right-distance","Balance-scale")
# Muestro número y tipo de variables, número de ejemplos,...
str(balance)
anyNA(balance)
summary(balance)
# Visualización de variables mediante histogramas.
x1 <- ggplot(balance, aes(x=balance$`Left-weight`)) + geom_histogram(binwidth = 0.5,fill='coral') + labs
x2 <- ggplot(balance, aes(x=balance$`Right-weight`)) + geom_histogram(binwidth = 0.5,fill='coral') + lab
x3 <- ggplot(balance, aes(x=balance$`Left-distance`)) + geom_histogram(binwidth = 0.5,fill='coral') + la
x4 <- ggplot(balance, aes(x=balance$`Right-distance`)) + geom_histogram(binwidth = 0.5,fill='coral') + l

plot_grid(x1, x2, x3, x4, labels = "AUTO")
# Distribución de variables dos a dos.
plot(balance[,-5],col=balance$`Balance-scale`)
# Correlación entre las variables
y <- cor(balance[1:4])
corrplot(y, method="number")
# Diagaramas de caja
aux_gather <- gather(balance, var, value, -`Balance-scale`)
ggplot(aux_gather, aes(y=value, x=factor(var)))+geom_boxplot(aes(fill=factor(var))) +guides(fill=guide_l
# Porcentaje de ejemplos con cada etiqueta
round(prop.table(table(balance$`Balance-scale`)) * 100, digits = 1)
# Diagrama de densidad de las variables en función de su clase
x1 <- ggplot(balance, aes(x=balance$`Left-weight`,fill=balance$`Balance-scale`))+geom_density(aes(alpha=
x2 <- ggplot(balance, aes(x=balance$`Left-distance`))+geom_density(aes(fill=balance$`Balance-scale`, alp
x3 <- ggplot(balance, aes(x=balance$`Right-weight`))+geom_density(aes(fill=balance$`Balance-scale`, alph
x4 <- ggplot(balance, aes(x=balance$`Right-distance`))+geom_density(aes(fill=balance$`Balance-scale`, al

plot_grid(x1, x2, x3, x4, labels = "AUTO")
# Cargo conjunto de datos
baseball <- read.csv("./baseball/baseball.dat", comment.char = "@")
names(baseball) <- c('Batting_average', 'On_base_percentage', 'Runs', 'Hits', 'Doubles', 'Triples', 'Hor
str(baseball)
anyNA(baseball)
summary(baseball)
x13 <- ggplot(baseball, aes(x=baseball$Free_agency_eligibility)) +geom_density(fill='coral') + labs(x='
x14 <- ggplot(baseball, aes(x=baseball$Free_agent)) + geom_density(fill='coral') + labs(x='',y='',title=
x15 <- ggplot(baseball, aes(x=baseball$Arbitration_eligibility)) + geom_density(fill='coral') + labs(x=
x16 <- ggplot(baseball, aes(x=baseball$Arbitration)) + geom_density(fill='coral') + labs(x='',y='',titl

plot_grid(x13,x14,x15,x16,ncol = 2)
x1 <- ggplot(baseball, aes(x=baseball$Batting_average)) +geom_density(fill='coral') + labs(x='',y='',ti
x2 <- ggplot(baseball, aes(x=baseball$On_base_percentage)) +geom_density(fill='coral') + labs(x='',y=''
x3 <- ggplot(baseball, aes(x=baseball$Runs)) +geom_density(fill='coral') + labs(x='',y='',title='Runs')
x4 <- ggplot(baseball, aes(x=baseball$Hits)) +geom_density(fill='coral') + labs(x='',y='',title='Hits')
x5 <- ggplot(baseball, aes(x=baseball$Doubles)) +geom_density(fill='coral') + labs(x='',y='',title='Doub
x6 <- ggplot(baseball, aes(x=baseball$Triples)) +geom_density(fill='coral') + labs(x='',y='',title='Trip
x7 <- ggplot(baseball, aes(x=baseball$HomeRuns)) +geom_density(fill='coral') + labs(x='',y='',title='Hor
x8 <- ggplot(baseball, aes(x=baseball$Runs_batted_in)) +geom_density(fill='coral') + labs(x='',y='',titl
x9 <- ggplot(baseball, aes(x=baseball$Walks)) +geom_density(fill='coral') + labs(x='',y='',title='Walks
x10 <- ggplot(baseball, aes(x=baseball$Strike_Outs)) +geom_density(fill='coral') + labs(x='',y='',title=
```

```
x11 <- ggplot(baseball, aes(x=baseball$Stolen_bases)) +geom_density(fill='coral') + labs(x='',y='',titl
x12 <- ggplot(baseball, aes(x=baseball$Errors)) +geom_density(fill='coral') + labs(x='',y='',title='Erro
plot_grid(x1, x2, x3, x4,x5,x6,ncol = 2)
plot_grid(x7,x8,x9,x10,x11,x12,ncol = 2)
par(mfrow=c(3,2))

qqnorm(y = baseball$Batting_average, main="QQ-plot Batting_average")
qqline(y=baseball$Batting_average)

qqnorm(y = baseball$On_base_percentage, main="QQ-plot On_base_percentage")
qqline(y=baseball$On_base_percentage)

qqnorm(y = baseball$Runs, main="QQ-plot Runs")
qqline(y=baseball$Runs)

qqnorm(y = baseball$Hits, main="QQ-plot Hits")
qqline(y=baseball$Hits)

qqnorm(y = baseball$Doubles, main="QQ-plot Doubles")
qqline(y=baseball$Doubles)

qqnorm(y = baseball$Triples, main="QQ-plot Triples")
qqline(y=baseball$Triples)

par(mfrow=c(3,2))

qqnorm(y = baseball$HomeRuns, main="QQ-plot HomeRuns")
qqline(y=baseball$HomeRuns)

qqnorm(y = baseball$Runs_batted_in, main="QQ-plot Runs_batted_in")
qqline(y=baseball$Runs_batted_in)

qqnorm(y = baseball$Walks, main="QQ-plot Walks")
qqline(y=baseball$Walks)

qqnorm(y = baseball$Strike_Outs, main="QQ-plot Strike_Outs")
qqline(y=baseball$Strike_Outs)

qqnorm(y = baseball$Stolen_bases, main="QQ-plot Stolen_bases")
qqline(y=baseball$Stolen_bases)

qqnorm(y = baseball$Errors, main="QQ-plot Errors")
qqline(y=baseball$Errors)
aux <- apply(baseball[1:16],2, shapiro.test)
v1 <- aux$Batting_average$p.value
v2 <- aux$On_base_percentage$p.value
v3 <- aux$Runs$p.value
v4 <- aux$Hits$p.value
v5 <- aux$Doubles$p.value
v6 <- aux$Triples$p.value
v7 <- aux$HomeRuns$p.value
v8 <- aux$Runs_batted_in$p.value
v9 <- aux$Walks$p.value
```

```r
v10 <- aux$Strike_Outs$p.value
v11 <- aux$Stolen_bases$p.value
v12 <- aux$Errors$p.value
v13 <- aux$Free_agency_eligibility$p.value
v14 <- aux$Free_agent$p.value
v15 <- aux$Arbitration_eligibility$p.value
v16 <- aux$Arbitration$p.value

results <- data.frame( c(v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12,v13,v14,v15,v16))
rownames(results) <- names(baseball)[1:16]
names(results) <- 'P-Valor'
knitr::kable(results)
y <- cor(baseball)
corrplot(y, method="circle")

# ------------ ------------ ------------ #
# ------------ CLASIFICACIÓN ------------ #
# ------------ ------------ ------------ #

# Creo conjuntos de train y test
shuffle <- sample(dim(balance)[1])
eightypct <- (dim(balance)[1] * 80) %/% 100
balance_train <- balance[shuffle[1:eightypct], ][,-5]
balance_test <- balance[shuffle[(eightypct+1):dim(balance)[1]], ][,-5]

# Obtengo las etiquetas de cada conjunto
balance_train_labels <- balance[shuffle[1:eightypct], 5]
balance_test_labels <- balance[shuffle[(eightypct+1):dim(balance)[1]], 5]
# Aplico knn sobre el conjunto de entrenamiento
knnModel <- train(x = balance_train, y = balance_train_labels, method = "knn",
                  metric="Accuracy")
knnModel
# Obtengo precisión y matriz de confusión en el test
knnPred <- predict(knnModel, newdata = balance_test)
postResample(pred = knnPred, obs = balance_test_labels)
table(knnPred,balance_test_labels)
# Valicación cruzada aplicando knn
cross_val_knn <- function(i, k, tt = "test") {
  x<- "balance"
  file <- paste(paste('./balance/',x,sep=""), "-10-", i, "tra.dat", sep="")
  x_tra <- read.csv(file,  header=TRUE, comment.char="@")
  file <- paste(paste('./balance/',x,sep=""), "-10-", i, "tst.dat", sep="")
  x_tst <- read.csv(file,  header=TRUE, comment.char="@")
  In <- length(names(x_tra)) - 1
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tra)[In+1] <- "Y"
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tst)[In+1] <- "Y"
  if (tt == "train") {
    test <- x_tra
  }
  else {
    test <- x_tst
```

```
  }
  yprime <- knn(train = x_tra[1:In], test = test[1:In], cl = x_tra$Y, k=k)
  mean(yprime==test$Y)
}
k <- 1
result <- c(-1,-1)
accuTrain <- c()
accuTest <- c()
while (k < 50){
  knnAccuracyTrain <-mean(sapply(1:5,cross_val_knn,k,"train"))
  knnAccuracyTest <-mean(sapply(1:5,cross_val_knn,k,"test"))

  accuTrain <- c(accuTrain,knnAccuracyTrain)
  accuTest <- c(accuTest,knnAccuracyTest)

  if (result[2] < knnAccuracyTest){
    result <- c(k,knnAccuracyTest)
  }
  k <- k + 1
}
result
results <- data.frame(c(1:49),accuTest,accuTrain)
names(results) <- c('k','accuTest','accuTrain')
ggplot(data=results, aes(x=k)) + geom_line(aes( y=accuTest, col='Test' ),size=1) + geom_line(aes( y=accu
knnAccuracyTrain <-mean(sapply(1:5,cross_val_knn,result[1],"train"))
knnAccuracyTest <-mean(sapply(1:5,cross_val_knn,result[1],"test"))

yprime <- knn(train = balance_train, test = balance_test, cl = balance_train_labels, k=result[1])
table(yprime,balance_test_labels)
# Obtengo conjuntos de train y test con etiquetas
balance_train = balance[shuffle[1:eightypct], ]
balance_test = balance[shuffle[(eightypct+1):dim(balance)[1]], ]
# Aplico LDA
lda.fit <- lda(`Balance-scale`~.,data=balance_train)
lda.pred <- predict(lda.fit,balance_test[,-5])

table(lda.pred$class,balance_test$`Balance-scale`)
mean(lda.pred$class==balance_test$`Balance-scale`)
# Validación cruzada sobre LDA
cross_val_lda <- function(i, tt = "test") {
  x<- "balance"
  file <- paste(paste('./balance/',x,sep=""), "-10-", i, "tra.dat", sep="")
  x_tra <- read.csv(file,  header=TRUE, comment.char="@")
  file <- paste(paste('./balance/',x,sep=""), "-10-", i, "tst.dat", sep="")
  x_tst <- read.csv(file,  header=TRUE, comment.char="@")
  In <- length(names(x_tra)) - 1
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tra)[In+1] <- "Y"
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tst)[In+1] <- "Y"
  if (tt == "train") {
    test <- x_tra
  }
```

```
  else {
    test <- x_tst
  }
  lda.fit <- lda(Y~.,data=x_tra)
  lda.pred <- predict(lda.fit,test[,-5])
  mean(lda.pred$class==test$Y)
}
ldaAccuracyTrain <-mean(sapply(1:5,cross_val_lda,"train"))
ldaAccuracyTest <-mean(sapply(1:5,cross_val_lda,"test"))
ldaAccuracyTrain
ldaAccuracyTest
# Aplico QDA
qda.fit <- qda(`Balance-scale`~.,data=balance_train)
qda.pred <- predict(qda.fit,balance_test[,-5])

table(qda.pred$class,balance_test$`Balance-scale`)
mean(qda.pred$class==balance_test$`Balance-scale`)
# Validación cruzada con QDA
cross_val_qda <- function(i, tt = "test") {
  x<- "balance"
  file <- paste(paste('./balance/',x,sep=""), "-10-", i, "tra.dat", sep="")
  x_tra <- read.csv(file,  header=TRUE, comment.char="@")
  file <- paste(paste('./balance/',x,sep=""), "-10-", i, "tst.dat", sep="")
  x_tst <- read.csv(file,  header=TRUE, comment.char="@")
  In <- length(names(x_tra)) - 1
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tra)[In+1] <- "Y"
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tst)[In+1] <- "Y"
  if (tt == "train") {
    test <- x_tra
  }
  else {
    test <- x_tst
  }
  qda.fit <- qda(Y~.,data=x_tra)
  qda.pred <- predict(qda.fit,test[,-5])
  mean(qda.pred$class==test$Y)
}
qdaAccuracyTrain <-mean(sapply(1:5,cross_val_qda,"train"))
qdaAccuracyTest <-mean(sapply(1:5,cross_val_qda,"test"))
qdaAccuracyTrain
qdaAccuracyTest
# Leo la tabla con los errores medios de test
resultados <- read.csv("clasif_test_alumos.csv")
tablatst <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatst) <- resultados[,1]

# Leo la tabla con los errores medios de entrenamiento
resultados <- read.csv("clasif_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
```

```r
rownames(tablatra) <- resultados[,1]

# Actualizo las tablas con los resultados obtenidos para regresión lineal y knn
tablatra[,1][3] <- knnAccuracyTrain
tablatra[,2][3] <- ldaAccuracyTrain
tablatra[,3][3] <- qdaAccuracyTrain

tablatst[,1][3] <- knnAccuracyTest
tablatst[,2][3] <- ldaAccuracyTest
tablatst[,3][3] <- qdaAccuracyTest
# Creo un método para aplicar Wilcoxon sobre las diferencias obtenidas para cada par de algoritmos
wilcoxon <- function(difs){
  wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0, abs(difs)+0.1, 0+0.1))
  colnames(wilc_1_2) <- c(colnames(tablatst)[1], colnames(tablatst)[2])
  head(wilc_1_2)

  alg1VSalg2tst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative = "two.sided", paired=TRUE)
  Rmas <- alg1VSalg2tst$statistic
  pvalue <- alg1VSalg2tst$p.value
  alg1VSalg2tst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative = "two.sided", paired=TRUE)
  Rmenos <- alg1VSalg2tst$statistic
  c(Rmas,Rmenos,pvalue)
}
# Aplico el test de Wilcoxon y almaceno los pvalues obtenidos
pvalues <- c()

difs <-  (tablatst[,1] - tablatst[,2]) / tablatst[,1]
pvalues <- c(pvalues,wilcoxon(difs)[3])

difs <-  (tablatst[,1] - tablatst[,3]) / tablatst[,1]
pvalues <- c(pvalues,wilcoxon(difs)[3])

difs <-  (tablatst[,2] - tablatst[,3]) / tablatst[,2]
pvalues <- c(pvalues,wilcoxon(difs)[3])

# Creo una tabla con estos valores
results <- data.frame(pvalues,(1-pvalues)*100)
names(results) <- c('P-valor','Confianza (%)')
rownames(results) <- c('K-NN vs LDA','K-NN vs QDA','LDA vs QDA')
knitr::kable(results)
# Aplico el test de Friedman sobre el conjunto de prueba
test_friedman <- friedman.test(as.matrix(tablatst))
test_friedman
# Aplico postHolm sobre el conjunto de prueba
tam <- dim(tablatst)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatst), groups, p.adjust = "holm", paired = TRUE)
# Aplico Wilcoxon sobre cada par de algoritmos
#en el conjutno de entrenamiento

pvalues <- c()

difs <-  (tablatra[,1] - tablatra[,2]) / tablatra[,1]
```

```r
pvalues <- c(pvalues,wilcoxon(difs)[3])

difs <-  (tablatra[,1] - tablatra[,3]) / tablatra[,1]
pvalues <- c(pvalues,wilcoxon(difs)[3])

difs <-  (tablatra[,2] - tablatra[,3]) / tablatra[,2]
pvalues <- c(pvalues,wilcoxon(difs)[3])

results <- data.frame(pvalues,(1-pvalues)*100)
names(results) <- c('P-valor','Confianza (%)')
rownames(results) <- c('K-NN vs LDA','K-NN vs QDA','LDA vs QDA')
knitr::kable(results)
# Aplico el test de Friedman sobre el conjunto de entrenamiento
test_friedman <- friedman.test(as.matrix(tablatra))
test_friedman
# Aplico postHolm sobre el conjunto de entrenamiento
tam <- dim(tablatra)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatra), groups, p.adjust = "holm", paired = TRUE)

# ------------- ------------- ------------- #
# -------------   REGRESIÓN   ------------- #
# ------------- ------------- ------------- #

# Calculo matriz de correlaciones
corMatrix <- cor(baseball,method="pearson")
corMatrix <- round(corMatrix,2)

# Obtengo los valores asociados al salario
# Busco ver qué valores tienen más relación con la variable de salida para
# seleccionar los cinco mejores candidatos.
numCols <- length(names(baseball))
min <- length((corMatrix))- numCols + 2
salaryCor <- corMatrix[min :length(corMatrix)-1]
highCorrIndex <- which( salaryCor > 0.57 )
highCorrIndex
names(baseball)[highCorrIndex]
# Aplico regresión lineal sobre los regresores más
# correladas con la variable de salida.
fit1 = lm(Salary~Hits,data=baseball)
Hits <- summary(fit1)$adj.r.squared

fit2 = lm(Salary~Doubles,data=baseball)
Doubles <- summary(fit2)$adj.r.squared

fit3 = lm(Salary~Runs,data=baseball)
Runs <- summary(fit3)$adj.r.squared

fit4 = lm(Salary~Runs_batted_in,data=baseball)
Runs_batted_in <- summary(fit4)$adj.r.squared

fit5 = lm(Salary~HomeRuns,data=baseball)
HomeRuns <- summary(fit5)$adj.r.squared
```

```r
results <- data.frame(Hits,Doubles,Runs,Runs_batted_in,HomeRuns)
rownames(results) <- "R2 ajustados"
knitr::kable(results)
# Aplico regresión lineal múltiple sobre el conjunto completo
fit6 = lm(Salary~.,data=baseball)
summary(fit6)$adj.r.squared
# Busco interacciones y combinariones no lineales
fit7 = lm(Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns),data=baseball)
summary(fit7)$adj.r.squared

fit8 = lm(Salary~. + I(Hits + Doubles + Runs + Runs_batted_in + HomeRuns)^2,data=baseball)
summary(fit8)$adj.r.squared

fit9 = lm(Salary~. + I(log(Hits)) ,data=baseball)
summary(fit9)$adj.r.squared
# Busco índices de los regresores menos correlados
# con la variable de salida
lowCorrIndex <- which( salaryCor < 0.15 )
names(baseball)[lowCorrIndex]
# Pruebo a eliminar estos regresores de los conjuntos probados
fit10 = lm(Salary~.-Arbitration-Free_agent-Errors-Arbitration_eligibility,data=baseball)
summary(fit10)$adj.r.squared
fit11 = lm(Salary~.-Arbitration-Free_agent-Errors,data=baseball)
summary(fit10)$adj.r.squared
fit12 = lm(Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
           -Arbitration-Free_agent-Errors-Arbitration_eligibility,data=baseball)
summary(fit12)$adj.r.squared
fit13 = lm(Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
           -Arbitration-Free_agent-Errors,data=baseball)
summary(fit13)$adj.r.squared
fit14 = lm(Salary~.  + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
          + I(Batting_average + On_base_percentage + Runs + Hits + Doubles
          + Triples + HomeRuns + Runs_batted_in + Walks + Strike_Outs + Stolen_bases +
          Free_agency_eligibility )^2, data=baseball)
summary(fit14)$adj.r.squared
# Validación cruzada aplicando regresión lineal simple
# Permite especificar el conjunto sobre el que aplicar
# este método
cross_val_lm <- function(i, set, tt = "test") {
  x<- "baseball"
  file <- paste(paste('./baseball/',x,sep=""), "-5-", i, "tra.dat", sep="")
  x_tra <- read.csv(file,  header=TRUE, comment.char="@")
  file <- paste(paste('./baseball/',x,sep=""), "-5-", i, "tst.dat", sep="")
  x_tst <- read.csv(file,  header=TRUE, comment.char="@")
  In <- length(names(x_tra)) - 1
  names(x_tra) <- c('Batting_average', 'On_base_percentage', 'Runs', 'Hits', 'Doubles', 'Triples', 'Hom
  names(x_tst) <- names(x_tra)
  if (tt == "train") {
    test <- x_tra
  }
  else {
    test <- x_tst
  }
```

```
  fitMulti = lm(set,x_tra)
  yprime = predict(fitMulti, test)
  sum(abs(test$Salary-yprime)^2)/length(yprime) ##MSE
}
# Aplico validación cruzada
lmMSEtrain<-mean(sapply(1:5,cross_val_lm,
                        Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
                        -Arbitration-Free_agent-Errors,"train"))
lmMSEtest<-mean(sapply(1:5,cross_val_lm,
                       Salary~. + (Hits * Doubles * Runs * Runs_batted_in * HomeRuns)
                       -Arbitration-Free_agent-Errors,"test"))
lmMSEtrain
lmMSEtest
lmMSEtrain<-mean(sapply(1:5,cross_val_lm,Salary~.,"train"))
lmMSEtest<-mean(sapply(1:5,cross_val_lm,Salary~.,"test"))
lmMSEtrain
lmMSEtest
lmMSEtrain<-mean(sapply(1:5,cross_val_lm,
                        Salary~.+ (Hits * Doubles * Runs * Runs_batted_in * HomeRuns),"train"))
lmMSEtest<-mean(sapply(1:5,cross_val_lm,
                       Salary~.+ (Hits * Doubles * Runs * Runs_batted_in * HomeRuns),"test"))
lmMSEtrain
lmMSEtest
lmMSEtrain<-mean(sapply(1:5,cross_val_lm,Salary~.-Arbitration-Free_agent-Errors,"train"))
lmMSEtest<-mean(sapply(1:5,cross_val_lm,Salary~.-Arbitration-Free_agent-Errors,"test"))
lmMSEtrain
lmMSEtest
# Muestro mejores resultados
tabla <- data.frame(lmMSEtrain,lmMSEtest)
names(tabla) <- c('Train','Test')
rownames(tabla) <- 'MSE'
knitr::kable(tabla)
# Creo conjuntos de entrenamiento y prueba
shuffle <- sample(dim(baseball)[1])
eightypct <- (dim(baseball)[1] * 80) %/% 100
baseball_train <- baseball[shuffle[1:eightypct], ]
baseball_test <- baseball[shuffle[(eightypct+1):dim(baseball)[1]], ]
# Validación cruzada para knn, permite especificar conjunto de regresores
# y número de vecinos a considerar
cross_val_knn <- function(i, set, k, tt = "test") {
  x<- "baseball"
  file <- paste(paste('./baseball/',x,sep=""), "-5-", i, "tra.dat", sep="")
  x_tra <- read.csv(file,  header=TRUE, comment.char="@")
  file <- paste(paste('./baseball/',x,sep=""), "-5-", i, "tst.dat", sep="")
  x_tst <- read.csv(file,  header=TRUE, comment.char="@")
  In <- length(names(x_tra)) - 1
  names(x_tra) <- c('Batting_average', 'On_base_percentage', 'Runs', 'Hits', 'Doubles', 'Triples', 'Hom
  names(x_tst) <- names(x_tra)
  if (tt == "train") {
    test <- x_tra
  }
  else {
    test <- x_tst
```

```
  }
  if (k>0){
    fitknn = kknn(set,x_tra, test, k=k)
  }
  else{
    fitknn = kknn(set,x_tra, test)
  }
  yprime = fitknn$fitted.values
  sum(abs(test$Salary-yprime)^2)/length(yprime) ##MSE
}
# Busco mejor número de vecinos para obtener el menor MSE posible
k <- 1
result <- c(-1, 9999999)
mseTrain <- c()
mseTest <- c()
accuTrain <- c()
accuTest <- c()
while (k < 50){
  knnMSETrain <-mean(sapply(1:5,cross_val_knn,
                            Salary ~ .-Arbitration-Free_agent-Errors,k,"train"))
  knnMSETest <-mean(sapply(1:5,cross_val_knn,
                           Salary ~ .-Arbitration-Free_agent-Errors,k,"test"))
  accuTrain <- c(accuTrain,knnMSETrain)
  accuTest <- c(accuTest,knnMSETest)

  if ( knnMSETest < result[2]){
    result <- c(k,knnMSETest)
  }
  k <- k + 1
}
result
# Muestro resultados
results <- data.frame(c(1:49),accuTest,accuTrain)
names(results) <- c('k','accuTest','accuTrain')
ggplot(data=results, aes(x=k)) + geom_line(aes( y=accuTest, col='Test' ),size=1) + geom_line(aes( y=accu
knnMSEtrain<-mean(sapply(1:5,cross_val_knn,
                         Salary ~ .-Arbitration-Free_agent-Errors,result[1],"train"))
knnMSEtest<-mean(sapply(1:5,cross_val_knn,
                        Salary ~ .-Arbitration-Free_agent-Errors,result[1],"test"))

tabla <- data.frame(knnMSEtrain,knnMSEtest)
names(tabla) <- c('Train','Test')
rownames(tabla) <- 'MSE'
knitr::kable(tabla)
# Aplico knn y lm sobre los conjuntos completos para realizar los test
knnMSEtrain<-mean(sapply(1:5,cross_val_knn,Salary ~ . ,-1,"train"))
knnMSEtest<-mean(sapply(1:5,cross_val_knn,Salary ~ .,-1,"test"))

lmMSEtrain<-mean(sapply(1:5,cross_val_lm,Salary~.,"train"))
lmMSEtest<-mean(sapply(1:5,cross_val_lm,Salary~.,"test"))

tabla <- data.frame(c(lmMSEtrain,knnMSETrain),c(lmMSEtest,knnMSEtest))
names(tabla) <- c('Train','Test')
```

```r
rownames(tabla) <- c('LM-MSE','KNN-MSE')
knitr::kable(tabla)
# Leo la tabla con los errores medios de test
resultados <- read.csv("regr_test_alumnos.csv")
tablatst <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatst) <- resultados[,1]

# Leo la tabla con los errores medios de entrenamiento
resultados <- read.csv("regr_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatra) <- resultados[,1]

# Actualizo las tablas con los resultados obtenidos para regresión lineal y knn
tablatra[,1][5] <- lmMSEtrain
tablatra[,2][5] <- knnMSEtrain
tablatst[,1][5] <- lmMSEtest
tablatst[,2][5] <- knnMSEtest
# Aplico Wilcoxon: knn vs lm
difs <-  (tablatst[,1] - tablatst[,2]) / tablatst[,1]
wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0, abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2) <- c(colnames(tablatst)[1], colnames(tablatst)[2])
LMvsKNNtst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative = "two.sided", paired=TRUE)
Rmas <- LMvsKNNtst$statistic
pvalue <- LMvsKNNtst$p.value
LMvsKNNtst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative = "two.sided", paired=TRUE)
Rmenos <- LMvsKNNtst$statistic
tabla <- data.frame(Rmenos,Rmas,pvalue)
rownames(tabla) <- "LM(R+) vs KNN(R-)"
knitr::kable(tabla)
# Aplico el test de Friedman sobre el conjunto de prueba
test_friedman <- friedman.test(as.matrix(tablatst))
test_friedman
# Aplico postHolm sobre el conjunto de prueba
tam <- dim(tablatst)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatst), groups, p.adjust = "holm", paired = TRUE)
difs <-  (tablatra[,1] - tablatra[,2]) / tablatra[,1]
wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0, abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2) <- c(colnames(tablatra)[1], colnames(tablatra)[2])
LMvsKNNtst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative = "two.sided", paired=TRUE)
Rmas <- LMvsKNNtst$statistic
pvalue <- LMvsKNNtst$p.value
LMvsKNNtst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative = "two.sided", paired=TRUE)
Rmenos <- LMvsKNNtst$statistic
tabla <- data.frame(Rmenos,Rmas,pvalue)
rownames(tabla) <- "LM(R+) vs KNN(R-)"
knitr::kable(tabla)
# Aplico el test de Friedman sobre el conjunto de entrenamiento
test_friedman <- friedman.test(as.matrix(tablatra))
test_friedman
# Aplico postHolm sobre el conjunto de entrenamiento
```

```r
tam <- dim(tablatra)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatra), groups, p.adjust = "holm", paired = TRUE)
```