Solution Exercice #2, Série 2

Francis Duval

12/02/2020

Pour les énoncés des exercices, cliquer sur ce lien: https://nbviewer.jupyter.org/github/nmeraihi/ACT6100/blob/master/exercices_2.ipynb

Activer les librairies utiles.

```r
library(here)
library(tidyverse)
library(magrittr)
library(FNN)
options(scipen = 999)
```

Lire la base de données `credit.csv`.

```r
d <- read_csv(here("0_data", "dataEX01.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

Pour voir rapidement à quoi ressemble la base de données, on peut utiliser la fonction `glimpse`.

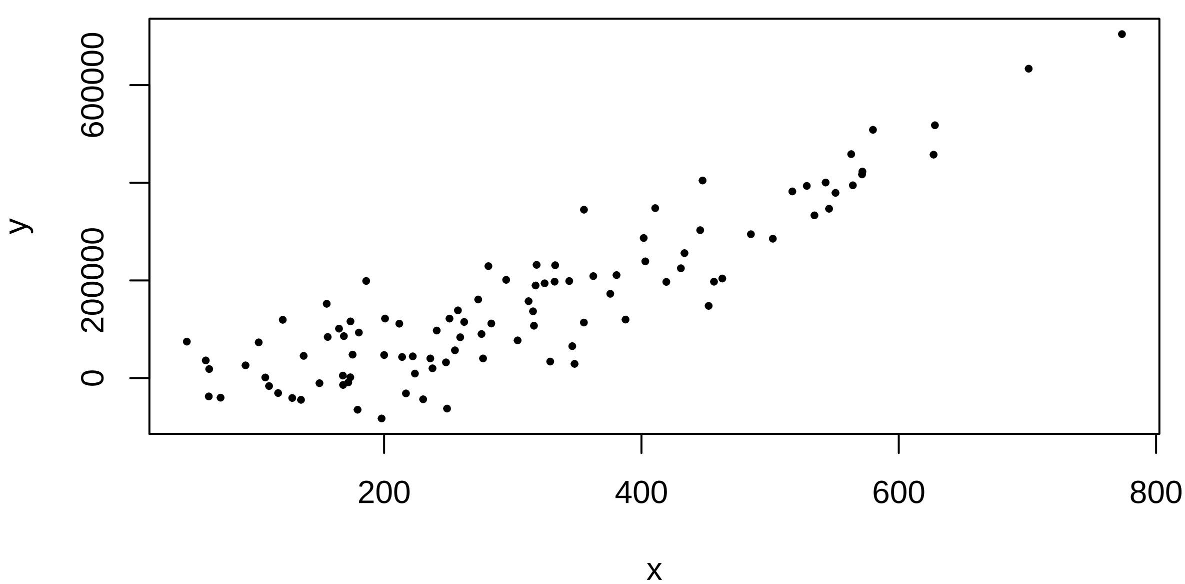
```r
glimpse(d)
```

```
## Observations: 100
## Variables: 3
## $ X1 <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...
## $ Y  <dbl> 74687.968, 36175.636, -37510.441, 18547.110, -40104.518, 26...
## $ X  <dbl> 46.66490, 61.38507, 63.74009, 64.06131, 72.90545, 92.31801,...
```

Partie 1

Nuage de points

```r
plot(d$X, d$Y, xlab = "x", ylab = "y", cex = 0.6, pch = 20)
```



Ajustement de la droite des moindres carrés

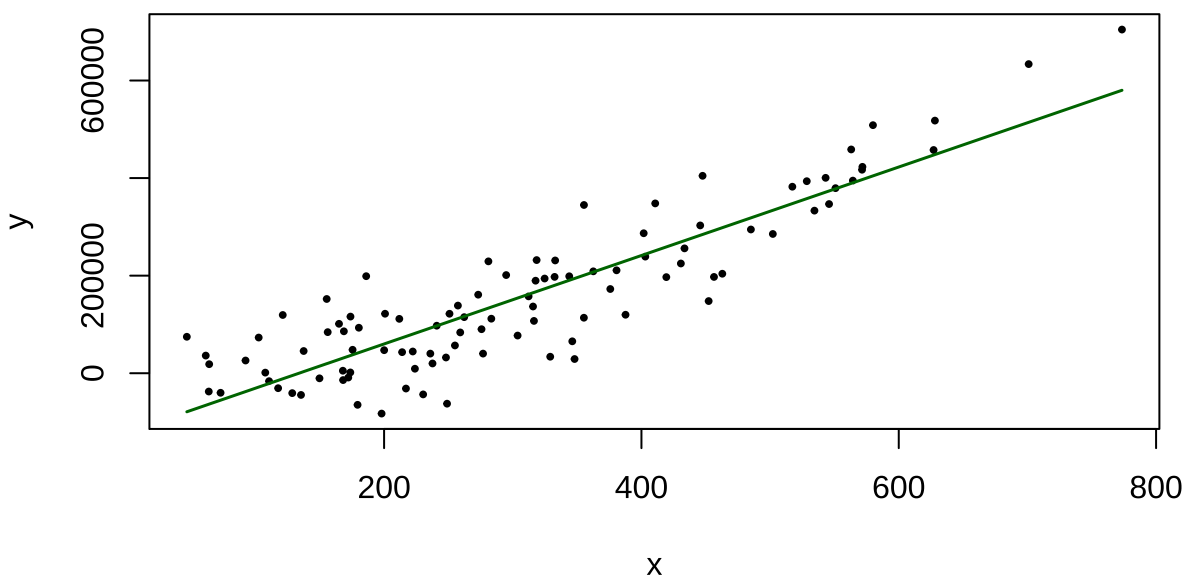
```
lin_fit <- lm(Y ~ X, data = d)
summary(lin_fit)
```

```
##
## Call:
## lm(formula = Y ~ X, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -166778  -45345    1335   50729  153733
##
## Coefficients:
##               Estimate Std. Error t value            Pr(>|t|)
## (Intercept) -121356.90   16241.48  -7.472      0.0000000000333 ***
## X               906.72      46.79  19.379 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 74500 on 98 degrees of freedom
## Multiple R-squared:  0.793,  Adjusted R-squared:  0.7909
## F-statistic: 375.5 on 1 and 98 DF,  p-value: < 0.00000000000000022
```

On voit que les 2 paramètres sont significatifs parce que leur valeur-p est très petite.

Visualiser l'ajustement sur le nuage de points

```
plot(d$X, d$Y, xlab = "x", ylab = "y", cex = 0.6, pch = 20)
lines(d$X, lin_fit$fitted.values, col = "darkgreen", lw = 1.5)
```



Calculer l'erreur quadratique moyenne (EQM), ou mean squared error en anglais (MSE)

```
mean((d$Y - lin_fit$fitted.values) ^ 2)
```

```
## [1] 5439107719
```

Partie 2

Ajout de x au carré comme prédicteur

```
d %<>% mutate(X_2 = X ^ 2)
```

Ajustement du modèle quadratique

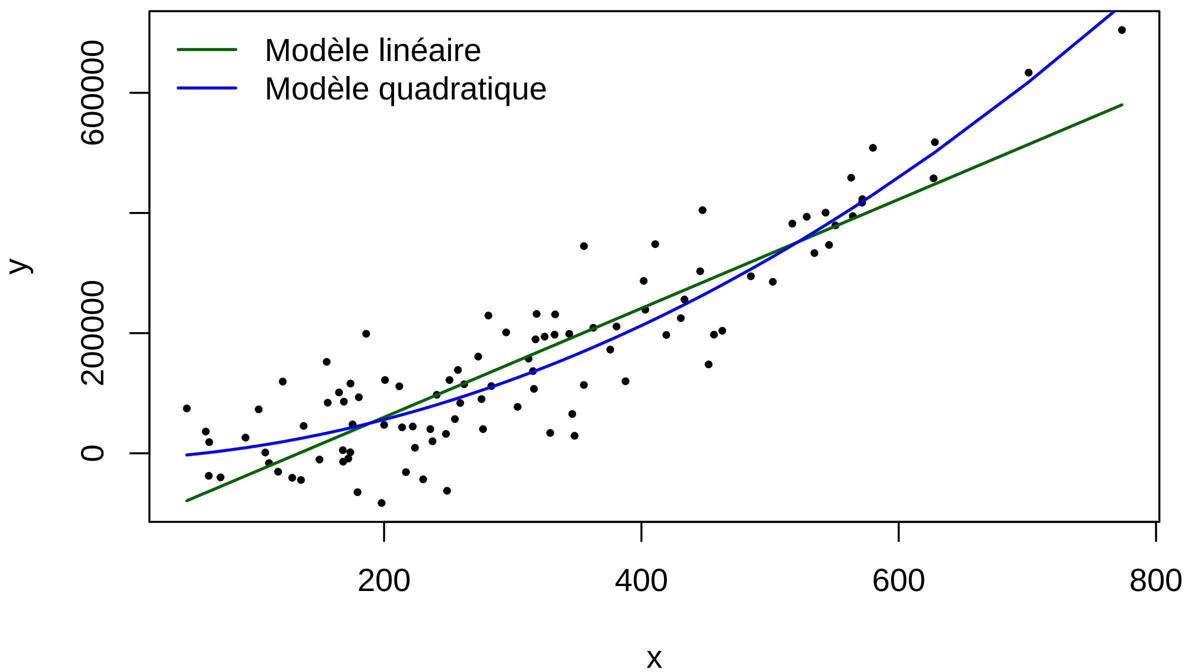
```
quad_fit <- lm(Y ~ X + X_2, data = d)
summary(quad_fit)
```

```
##
## Call:
## lm(formula = Y ~ X + X_2, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -148471  -41797    -132   47102  175067
##
## Coefficients:
##               Estimate  Std. Error t value   Pr(>|t|)
## (Intercept) -10175.7173  26552.1416  -0.383      0.702
## X              105.9690    165.3555   0.641      0.523
## X_2              1.1271      0.2251   5.006 0.00000248 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 66750 on 97 degrees of freedom
## Multiple R-squared:  0.8355, Adjusted R-squared:  0.8322
## F-statistic: 246.4 on 2 and 97 DF,  p-value: < 0.0000000000000022
```

Les valeurs p associées aux paramètres permettent de conclure que X n'est pas une variable significative dans le modèle.

Visualiser l'ajustement sur le nuage de points

```
plot(d$X, d$Y, xlab = "x", ylab = "y", cex = 0.6, pch = 20)
lines(d$X, lin_fit$fitted.values, col = "darkgreen", lw = 1.5)
lines(d$X, quad_fit$fitted.values, col = "blue", lw = 1.5)
legend(
  "topleft",
  legend = c("Modèle linéaire", "Modèle quadratique"),
  col = c("darkgreen", "blue"),
  lwd = 1.5,
  bty = "n"
)
```



Calculer l'erreur quadratique moyenne

```
mean((d$Y - quad_fit$fitted.values) ^ 2)
```

```
## [1] 4322345712
```

Partie 3

On entraine les modèles pour K = 1, . . . , 20 (la fonction map c'est un peu comme lapply, mais meilleur).

```
knn_fits <- map(
  1:20,
  ~ knn.reg(train = enframe(d$X, name = NULL), test = enframe(d$X, name = NULL), y = d$Y, k = .x)
)
```

Enuite, on obtient les prédictions des 20 modèles.

```
knn_pred <- map(knn_fits, ~ .x$pred)
```

On calcule le MSE pour chacun des 20 modèles.

```
(knn_mse <- map_dbl(knn_pred, ~ mean((.x - d$Y) ^ 2)))
```

```
##  [1]          0 1896957322 3048235721 3721561291 4067497199 4442591152
##  [7] 4665245022 4791490018 4677422318 4728709910 4730485065 4853004628
## [13] 4911970299 4993862620 5176783275 5411826880 5414742365 5664690133
## [19] 5802564218 6106575888
```

La valeur de K qui minimise le MSE d'entrainement est donc

```
(opt_k <- which.min(knn_mse))
```

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

et son erreur quadratique moyenne est

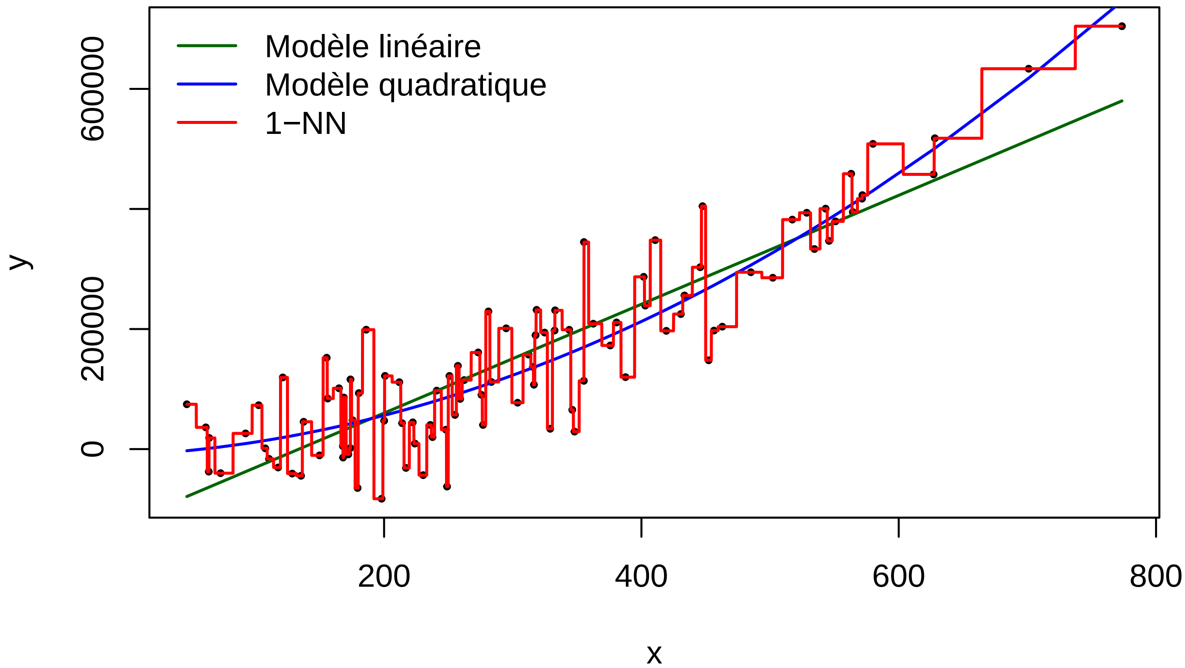
```
knn_mse[which.min(knn_mse)]
```

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

On peut ensuite ajouter la fonction de prédiction au graphique.

```
x_grid <- tibble(X = (seq(min(d$X), max(d$X), length.out = 10000)))
knn_fit <- knn.reg(train = d["X"], test = x_grid, y = d$Y, k = opt_k)

plot(d$X, d$Y, xlab = "x", ylab = "y", cex = 0.6, pch = 20)
lines(d$X, lin_fit$fitted.values, col = "darkgreen", lw = 1.5)
lines(d$X, quad_fit$fitted.values, col = "blue", lw = 1.5)
lines(x_grid$X, knn_fit$pred, col = "red", lw = 1.5)
legend(
  "topleft",
  legend = c("Modèle linéaire", "Modèle quadratique", "1-NN"),
  col = c("darkgreen", "blue", "red"),
  lwd = 1.5,
  bty = "n"
)
```



Partie 4

On va tout d'abord définir une fonction qui calcule l'EQM avec leave one out cross-validation.

```
loocv_mse_knn <- function(x_train_vec, y_train_vec, k) {
  n <- length(x_train_vec)

  preds <- map_dbl(
    1:n,
    ~ knn.reg(train = x_train_vec[-.x], test = x_train_vec[.x], y = y_train_vec[-.x], k = k)$pred
  )

  mse <- mean((preds - y_train_vec) ^ 2)
  return(mse)
}
```

On peut ensuite calculer l'EQM LOOCV pour $K = 1, \ldots, 20$.

```
(knn_loocv_mse <- map_dbl(1:20, ~ loocv_mse_knn(d$X, d$Y, k = .x)))
```

```
##  [1] 7587829289 6858530373 6616108961 6355464374 6397331258 6349916836
##  [7] 6258272676 5919862621 5837913469 5723886929 5775476582 5764742921
```

```
## [13] 5791698659 5942735902 6157456361 6112736498 6350725270 6465202724
## [19] 6766289073 6742108789
```

Finalement, on détermine la valeur optimale de K.

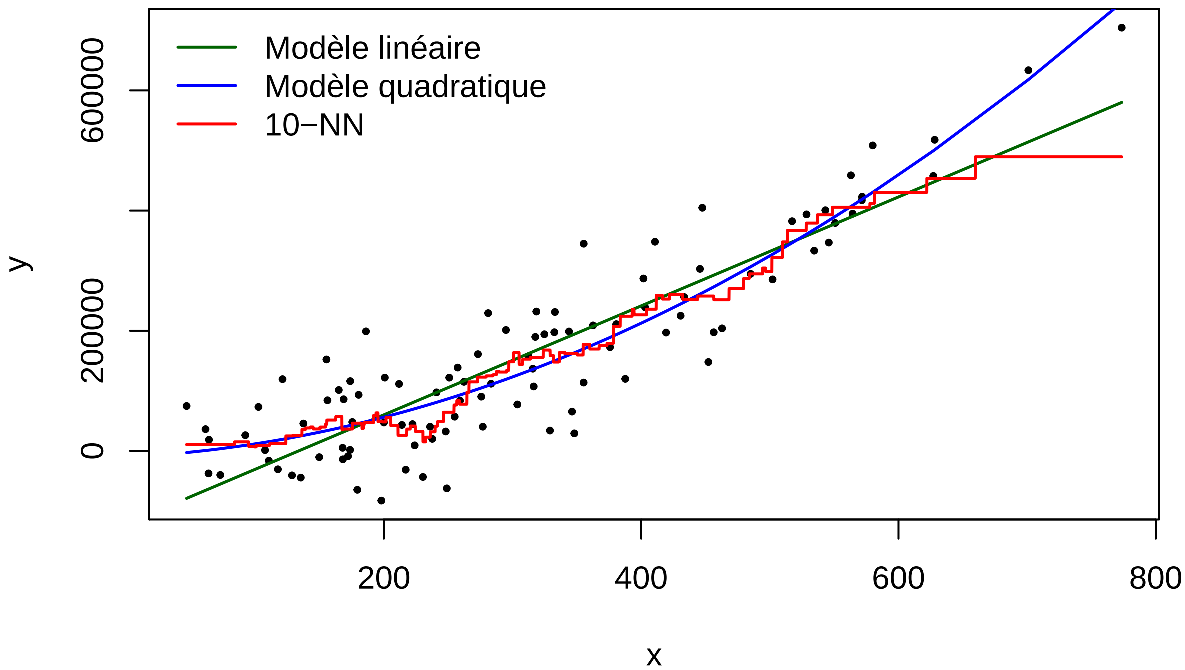
```
(opt_k_loocv <- which.min(knn_loocv_mse))
```

```
## [1] 10
```

On ajoute le modèle au graphique.

```
knn_fit <- knn.reg(train = d["X"], test = x_grid, y = d$Y, k = opt_k_loocv)

plot(d$X, d$Y, xlab = "x", ylab = "y", cex = 0.6, pch = 20)
lines(d$X, lin_fit$fitted.values, col = "darkgreen", lw = 1.5)
lines(d$X, quad_fit$fitted.values, col = "blue", lw = 1.5)
lines(x_grid$X, knn_fit$pred, col = "red", lw = 1.5)
legend(
  "topleft",
  legend = c("Modèle linéaire", "Modèle quadratique", "10-NN"),
  col = c("darkgreen", "blue", "red"),
  lwd = 1.5,
  bty = "n"
)
```



L'erreur quadratique moyenne LOOCV associée à ce modèle est

```
knn_loocv_mse[which.min(knn_loocv_mse)]
```

```
## [1] 5723886929
```

Partie 5

```
x0 <- 333.2522
y0 <- 99508.44
newdata <- tibble(X = x0, X_2 = x0 ^ 2)
```

Modèle linéaire

```
(y_hat_lin_fit <- predict(lin_fit, type = "response", newdata = newdata) %>% as.numeric()) # Prédiction
```

```
## [1] 180809.5
```

```
(y_hat_lin_fit - y0) ^ 2 # Erreur quadratique de prédiction
```

```
## [1] 6609860934
```

Modèle quadratique

```
(y_hat_quad_fit <- predict(quad_fit, type = "response", newdata = newdata) %>% as.numeric()) # Prédiction
```

```
## [1] 150313
```

```
(y_hat_quad_fit - y0) ^ 2 # Erreur quadratique de prédiction
```

```
## [1] 2581105753
```

1-NN

```
y_hat_1_nn <- knn.reg(train = d$X, test = x0, y = d$Y, k = 1)$pred # Prédiction
(y_hat_1_nn - y0) ^ 2 # Erreur quadratique de prédiction
```

```
## [1] 17333310634
```

10-NN

```
y_hat_10_nn <- knn.reg(train = d$X, test = x0, y = d$Y, k = 10)$pred # Prédiction
(y_hat_10_nn - y0) ^ 2 # Erreur quadratique de prédiction
```

```
## [1] 2338794058
```

On remarque que le modèle des plus proches voisins avec $K = 1$ produit une erreur quadratique de prédiction très élevée et que le modèle optimal ($K = 10$) est celui dont l'erreur quadratique de prédiction est la plus faible.