pacman

Jingwen SU

12/10/2020

Introduction

Dans cet article, nous allons essayer d'utiliser le package **pracma** pour effectuer la différence polynomiale et l'ajustement et l'ajustement linéaire.

Installer le package pracma

```
#install.packages("parcma")
library(pracma)
```

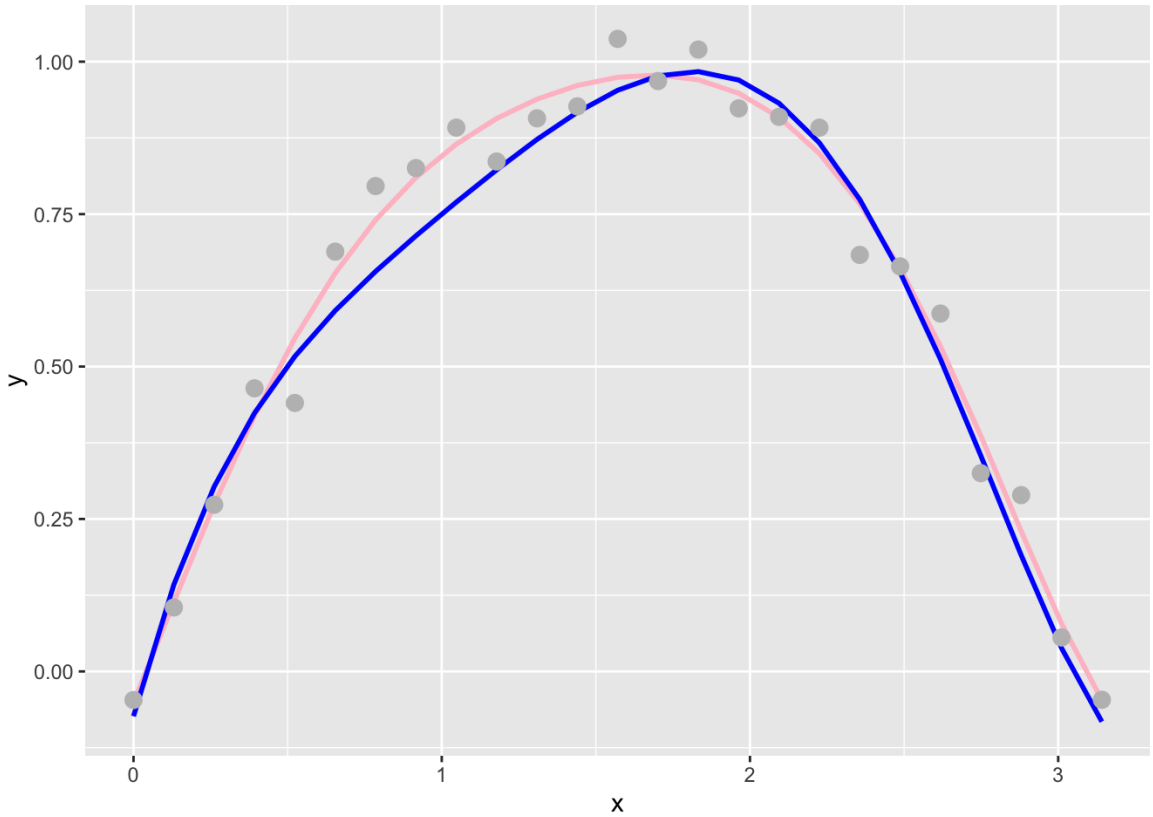
Différence et ajustement polynomial

- polyfit (x, y, n) génère des coefficients polynomiaux, et sa puissance est triée de haut en bas, et n <(longueur (x) -1) ajuste automatiquement les données
- polyfix (x, y, n, xfix, yfix) est également un paramètre de coefficient polynomial. Les paramètres xfix et yfix représentent les coordonnées du point de référence, ce qui signifie que la courbe d'ajustement doit passer ce point
- polyvalent (p, x) Selon le vecteur de coefficient polynomial P, générer un polynôme, puis calculer la valeur de la coordonnée x en fonction du polynôme

```
library(ggplot2)

set.seed(1)
x <- seq(0, pi, length.out = 25)
y <- sin(x) + 0.05 * runif(length(x), -2, 2)
p1 <- polyfit(x, y, 6)  # Ajuster un polynôme d'ordre 6 et renvoyer un vecteur de longueur 7
p2 <- polyfix(x, y, 6, xfix = c(1, 3), yfix = c(0.75, 0.05))
p3 <- polyval(p1, x)

data1 <- data.frame(x = x, y = p3, y_fix = polyval(p2, x), y_point = y, stringsAsFactors = F)
g1 <- ggplot(data1) + geom_line(aes(x, y), color = "pink", size = 1) + geom_line(aes(x,
    y_fix), color = "blue", size = 1) + geom_point(aes(x, y_point), color = "grey",
    size = 3)
g1
```

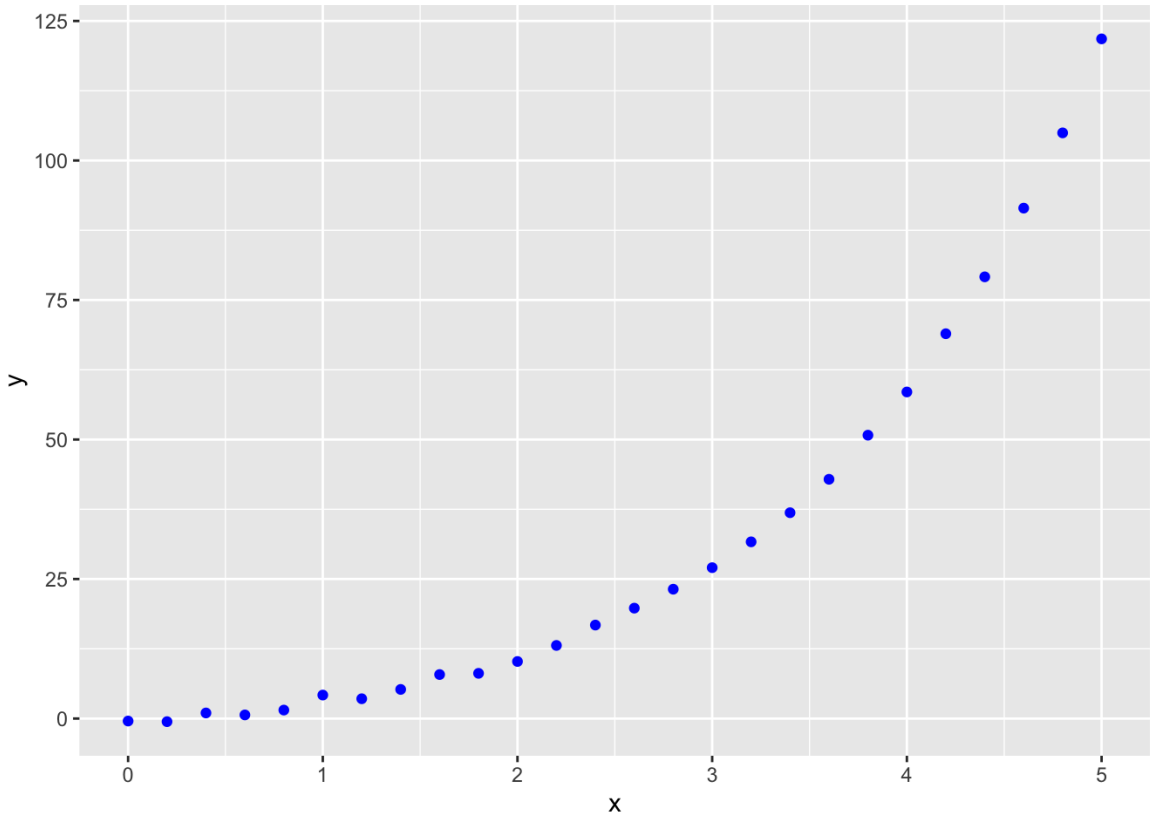


Ajustement linéaire

En combinant le contenu du précédent caret.rmd, les étapes pour créer un ajustement linéaire sont les suivantes: * Le premier est le prétraitement des données (y compris l'élimination des valeurs aberrantes, la normalisation des données: suppression d'unités, soustraction de la valeur minimale, division par la valeur maximale, de sorte que les données soient dans l'intervalle (0, 1)) * Puis dessinez un nuage de points et établissez une relation fonctionnelle basée sur la distribution des points (pas nécessairement une fonction linéaire) * Ensuite, selon la relation de fonction, la relation de fonction linéaire est dérivée * Selon la relation de fonction linéaire, effectuez un ajustement linéaire, résolvez le coefficient de corrélation, apportez la solution dans la fonction d'origine et dessinez l'image de la fonction

```
set.seed(11)
black <- function(x) {
    2 * x * exp(0.5 * x) + runif(length(x), min = -1, max = 1)
}
x_test <- seq(0, 5, 0.2)
y_test <- black(x_test)

rigion <- data.frame(x = x_test, y = y_test)  # Données prétraitées
ggplot(rigion, aes(x, y)) + geom_point(color = "blue")  # Dessinez un nuage de points à partir duqu
el vous pouvez construire un modèle de fonction exponentielle
```




```
# Modèle de fonction: y = c1*t*exp(c2*t), Linéarisation des fonctions: lny = lnc1 + lnt + c2*t
# Substituer des variables et transformer des inconnues en coefficients de fonctions linéaires: lny
-lnt = c2*t + k,
# La variable indépendante est t, la variable dépendante est lny-lnt et le coefficient est calculé
 par ajustement linéaire: k, c2
fun_y <- log(y_test, base = exp(1)) - log(x_test, base = exp(1))
```

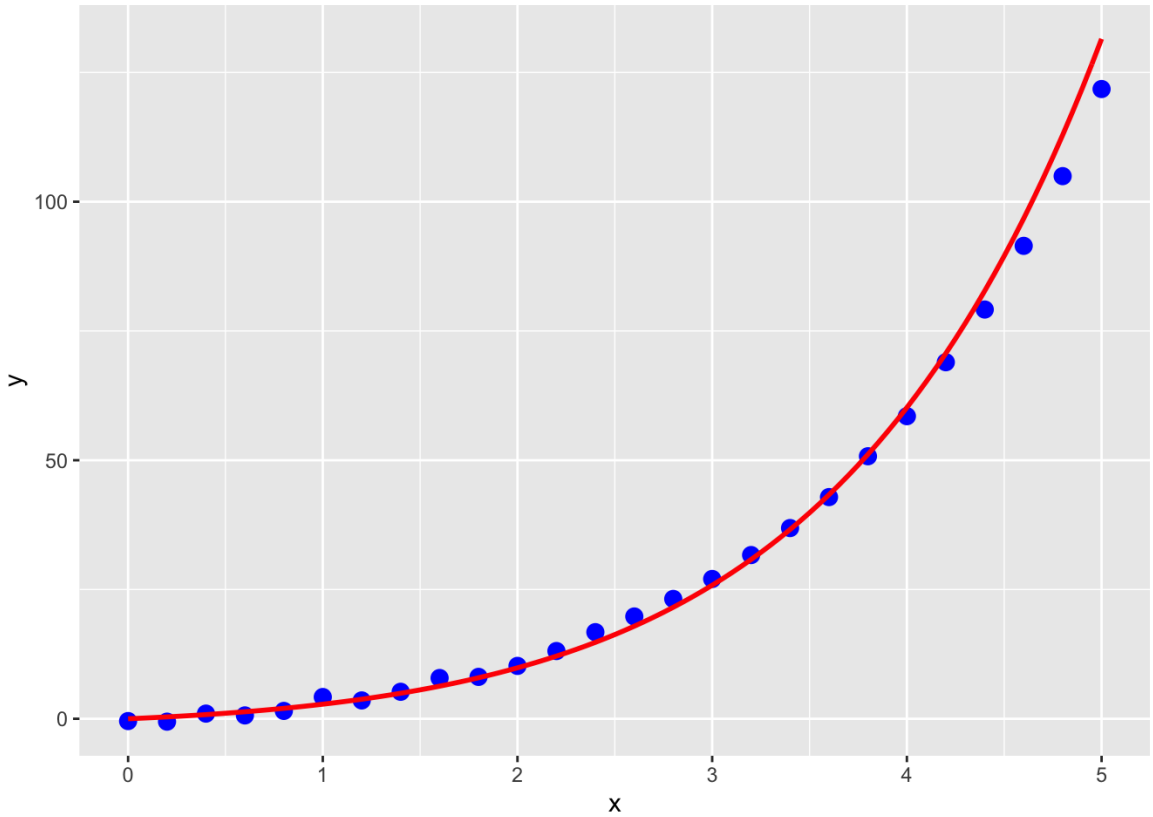
```
## Warning: NaNs produced
```

```
fun_x <- x_test
relation <- lm(fun_y ~ fun_x)  # Établir une relation linéaire
print(relation)  # Afficher le coefficient de relation, les résultat: c2 = 0.5583, k = 0.4780
```

```
##
## Call:
## lm(formula = fun_y ~ fun_x)
##
## Coefficients:
## (Intercept)        fun_x
##      0.4780       0.5583
```

```
k <- relation[[1]][1]
c2 <- relation[[1]][2]

# Apportez les valeurs de c2 et k pour trouverc1 = e^k
c1 <- exp(k)
# Le modèle de fonction est:
fun_last <- function(x) {
    c1 * x * exp(c2 * x)
}
# Dessinez le graphique ajusté
ggplot(rigion, aes(x, y)) + geom_point(color = "blue", size = 3) + stat_function(fun = fun_last,
    color = "red", size = 1)
```


100
50
0
y
0
1
2
3
4
5
x