# Mon analyse R en mathématiques

**Jingwen SU**

**12/20/2020**

# Introduction

À la fin du cours, tous les étudiants ont téléchargé les documents pertinents sur Github et partagé les résultats avec nous. Afin de profiter pleinement de ces packages R sur les mathématiques, je vais lire, analyser et évaluer les articles de mes collègues pour améliorer mes lacunes.

Ceci est une introduction à l'article lu dans cet article. Si vous souhaitez en savoir plus, vous pouvez rechercher des références.

- **Title:** pacma
- **Auteurs:** Jingwen SU
- **Lien sur Github:** Jingwen SU (https://github.com/Jingwen-su/PSBX/blob/main/Document%20final/gr01_Jingwen_SU_pacman.pdf)

# Synthèse du travail en question

Ceci est un article d'introduction sur pacma. L'auteur a donné une brève introduction sur le contenu de l'article au début. Dites au lecteur que dans l'article, elle essaiera d'utiliser le package Pracma pour la différence polynomiale, l'ajustement et l'ajustement linéaire.

Dans la deuxième partie, l'auteur a présenté le code d'installation du package Pracma. Après cela, l'auteur a commencé à effectuer des calculs de fonction.

Pour Différence et ajustement polynomial, il a introduit trois fonctions: polyfit, polyfix et polyvalent. D'un certain point de vue, le premier et le second sont similaires.

Pour Ajustement linéaire, l'auteur a d'abord passé au peigne fin le processus d'ajustement linéaire en détail, puis a démontré en fonction de chaque étape.

# Contenu principal et explication

## 1. Installer le package pracma

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

## 2. Les fonctions

- **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.

## 3. Ajustement linéaire

Ici, nous devons maîtriser sa pensée logique. Dans un projet ou un calcul, il n'y a souvent pas de package d'installation unique impliqué. Nous devons apprendre à combiner plusieurs packages d'installation et à maîtriser les fonctions fréquemment utilisées.

L'auteur raconte un processus complet de traitement des données. Pour l'ajustement linéaire, lorsque nous obtenons les données:

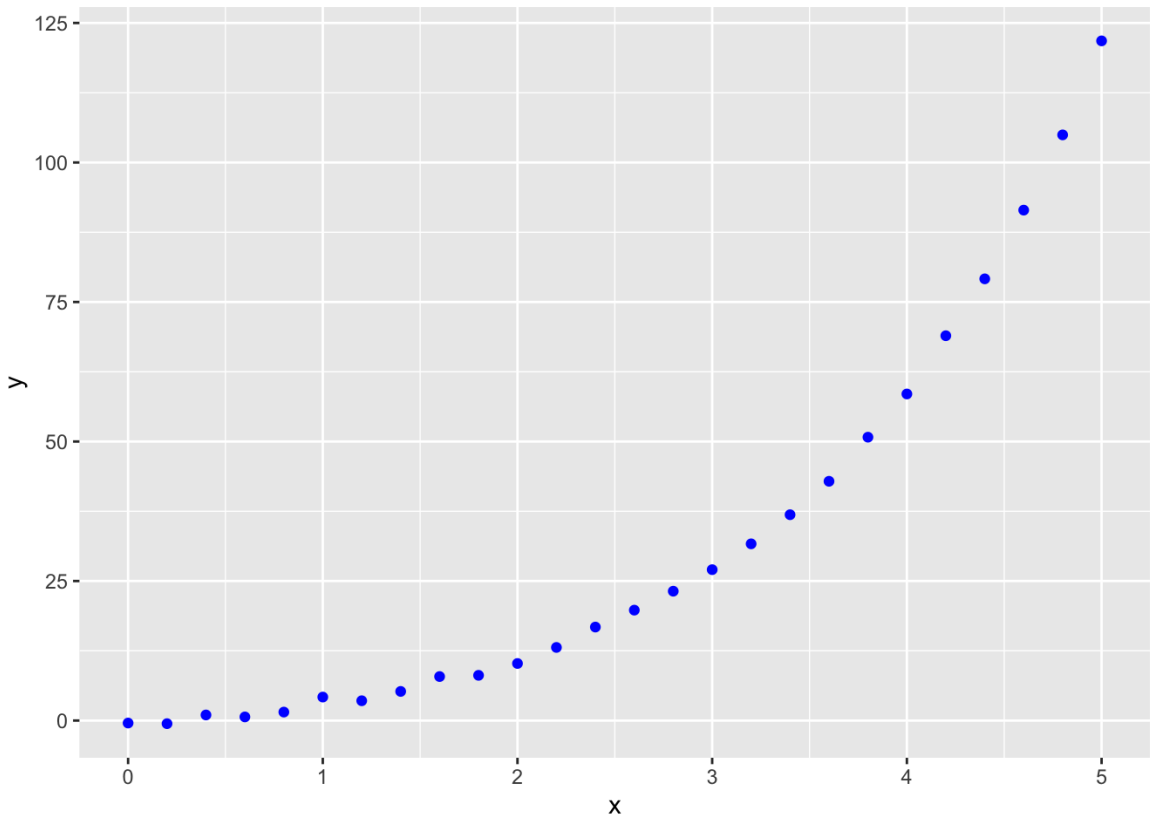
- Le premier est le prétraitement des données (y compris l'élimination des valeurs aberrantes, la normalisation des données: suppression d'unités, soustraction de la valeur minimale, division par la valeur maximale, de sorte que les données soient dans l'intervalle (0, 1)).
- Puis dessinez un nuage de 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.

La démonstration est la suivante:

```
library(ggplot2)

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))
```

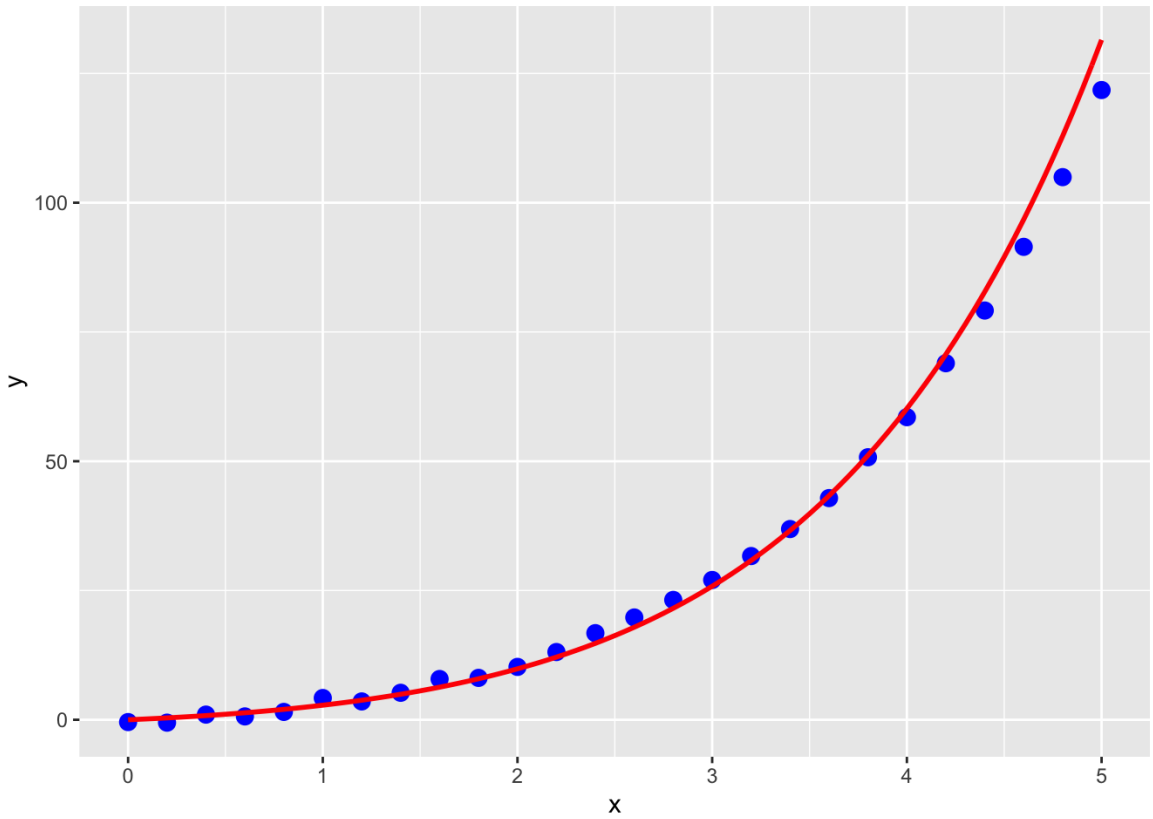
```
## Warning: NaNs produced
```

```
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, c
olor = "red", size = 1)
```




# Evaluation et résumer

Selon mes normes, je pense que c'est un article moyen.

Tout d'abord, son format est très concis et il y a des explications détaillées sur le contenu important. De plus, le contenu proposé a également été démontré. Le cadre logique est très bon.

Cependant, il n'a pas donné une introduction générale au package d'installation, et n'a pas donné de bibliographie, de sorte que les gens ne pouvaient pas le comprendre davantage.