# Résumé sur Illustration dans R

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 d'utiliser ces packages d'installation de R de manière plus complète, 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:** MNIST et Fashion MNIST (Illustration dans R)
- **Auteurs:** Kanlanfeyi Kabirou, Hounsinou Jordy
- **Lien sur Github:** kabirou7

## Synthèse du travail en question

Dans cet article, l'auteur a d'abord brièvement expliqué le contexte. Ensuite, deux bases de données et algorithmes à utiliser sont introduits. Ensuite, l'auteur a combiné l'algorithme et la base de données pour afficher le code. Enfin, l'auteur résume les deux algorithmes en fonction des résultats d'exploitation. Et les matériaux de référence énumérés à la fin.

Dans la partie affichage du code, l'auteur utilise principalement le package d'installation: dplyr. Grâce à l'utilisation de la base de données, la segmentation des données, la normalisation des données, la division jeu de la formation, la construction de modèles et la visualisation des données sont présentées.

## Contenu principal et explication

### 1.Partie d'importation de package de données

Pour cette partie, nous n'avons pas besoin d'importer tous les packages d'installation que nous connaissons, il suffit d'importer les packages d'installation dont nous avons besoin.

L'auteur a également écrit sur caret et ggplot2 dans l'article. Bien sûr, il n'est pas exclu que l'auteur initialement prévu d'utiliser ces packages d'installation dans une analyse ultérieure, mais il ne peut pas être indiqué dans l'article en cours en raison de problèmes de temps.

PS:Comme il n'a pas fourni les fichiers de base de données pertinents, afin d'éviter les erreurs, j'ajouterai "#" devant le code de la base de données ci-dessous.

```
library(readr)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

```
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```
library(class)

#Pour fractionner les données
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# mnist <- read_csv("train.csv")
# fashion <- read_csv("fashion.csv")
```

### 2. la fonction factor

Cette étape est très importante. Pour les grandes bases de données, cela peut améliorer notre utilisation ultérieure. Apportez-nous la commodité.

```
# mnist$label = factor(mnist$label)
# fashion$label = factor(fashion$label)
```

### 3.Normalisation des données

Pour les bases de données que nous ne connaissons pas, cette étape peut nous aider à traiter plus rapidement quelques paramètres insignifiants dans les données.

```
# mnist[,2:785] = mnist[,2:785]/255
# fashion[,2:785] = fashion[,2:785]/255
```

### 4.Créer un ensemble de test

Afin d'assurer le caractère aléatoire de nos données, nous pouvons diviser les données en deux parties avant de construire le modèle, une pour les tests et une pour la prédiction des résultats. Pour éviter la duplication des données et affecter les résultats.

```
# train_mnist <-  sample_frac(mnist, 0.8)
# test_mnist <- anti_join(mnist, train_mnist)

# train_fashion <-  sample_frac(fashion, 0.8)
# test_fashion <- anti_join(fashion, train_fashion)
```

### 5.Random Forest et Naive Bayes

Le modèle est construit en utilisant la fonction **predict**

**Random Forest** est **rf_MNIST**

```
# rf_MNIST <- randomForest(label ~ ., data = train_mnist, ntree = 10)
# pred_MNIST1 <- predict(rf_MNIST, test_mnist)

# rf_FASH <- randomForest(label ~ ., data = train_fashion, ntree = 10)
# pred_FASH1 <- predict(rf_FASH, test_fashion)
```

**Naive Bayes** est **bayes_MNIST**

```
# bayes_MNIST <- randomForest(label ~ ., data = train_mnist)
# pred_MNIST2 <- predict(bayes_MNIST, test_mnist)

# bayes_FASH <- randomForest(label ~ ., data = train_fashion)
# pred_FASH2 <- predict(bayes_FASH, test_fashion)
```

## Evaluation et résumer

Selon mes critères d'évaluation, je pense en fait que c'est presque un article parfait. Il répondait à presque toutes mes exigences: format concis et clair, explications détaillées, exemples de fonctions, titres et annexes. Son format est le meilleur parmi les documents de mes collègues que j'ai vus, ce qui est très confortable.

Le seul inconvénient est que vous pouvez utiliser plus de fonctions et donner des exemples, les fonctions de l'article sont un peu moins nombreuses. Deuxièmement, le titre de son article n'est pas clair. Mais cela n'affecte pas son excellence.