dplyr

Jingwen SU

12/9/2020

introduction

Dans cet article, nous allons vous présenter comment utiliser le package **dplyr**. Afin d'approfondir notre compréhension, nous importerons l'ensemble de données "birthwt" dans le package **MASS** pour une démonstration.

Installez dplyr

```
## install.packages("tidyverse")
## install.packages("dplyr")
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyver
se 1.3.0 ──
```

```
## ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
## ✓ tibble  3.0.4     ✓ dplyr   1.0.2
## ✓ tidyr   1.1.2     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_c
onflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
```

Faites attention aux informations de conflit (Conflits) fournies lorsque vous importez le package tidyverse. Cela vous indique que dplyr couvre les fonctions du package de base R. Si vous souhaitez utiliser ces fonctions après avoir chargé le paquet tidyverse, vous devez utiliser les noms complets des fonctions stats :: filter () et stats :: lag () pour les appeler.

arrange () : Réorganiser les lignes

La fonction arrange () fonctionne de la même manière que filter (), mais au lieu de sélectionner des lignes, elle change l'ordre des lignes. Il utilise un bloc de données et une série de variables de colonne ordonnées (ou d'expressions plus complexes) en entrée. Si vous indiquez plusieurs noms de colonnes, les autres colonnes sont triées en conséquence.

```
library(MASS)#birthwt
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
head(birthwt)
```

```
##    low age lwt race smoke ptl ht ui ftv  bwt
## 85   0  19 182    2     0   0  0  1   0 2523
## 86   0  33 155    3     0   0  0  0   3 2551
## 87   0  20 105    1     1   0  0  0   1 2557
## 88   0  21 108    1     1   0  0  1   2 2594
## 89   0  18 107    1     1   0  0  1   0 2600
## 91   0  21 124    3     0   0  0  0   0 2622
```

```
# arrange(birthwt,age,lwt,race)
head(arrange(birthwt,age,lwt,race))
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt
## 81    1  14 100    3     0   0  0  0   2 2495
## 78    1  14 101    3     1   1  0  0   0 2466
## 213   0  14 135    1     0   0  0  0   0 3941
## 102   0  15  98    2     0   0  0  0   0 2778
## 57    1  15 110    1     0   0  0  0   0 2353
## 62    1  15 115    3     0   0  0  1   0 2381
```

Utilisez desc () pour trier dans l'ordre inverse (ordre décroissant):

```
# arrange(birthwt,desc(age))
head(arrange(birthwt,desc(age)))
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt
## 226   0  45 123    1     0   0  0  0   1 4990
## 108   0  36 202    1     0   0  0  0   1 2836
## 183   0  36 175    1     0   0  0  0   0 3600
## 119   0  35 121    2     1   1  0  0   1 2948
## 223   0  35 170    1     0   1  0  0   1 4174
## 11    1  34 187    2     1   0  1  0   0 1135
```

select(): Sélectionner la colonne

En général, le jeu de données original que nous analysons contient beaucoup de variables (colonnes). Le premier problème que nous devons résoudre est de restreindre la portée pour trouver les données (variables) dont nous avons besoin. select () nous permet de sous-ensemble rapidement l'ensemble de données par nom de variable.

```
head(birthwt)
```

```
##    low age lwt race smoke ptl ht ui ftv  bwt
## 85   0  19 182    2     0   0  0  1   0 2523
## 86   0  33 155    3     0   0  0  0   3 2551
## 87   0  20 105    1     1   0  0  0   1 2557
## 88   0  21 108    1     1   0  0  1   2 2594
## 89   0  18 107    1     1   0  0  1   0 2600
## 91   0  21 124    3     0   0  0  0   0 2622
```

```
# dplyr::select(birthwt, smoke, ui, bwt)
head(dplyr::select(birthwt, smoke, ui, bwt))
```

```
##    smoke ui  bwt
## 85     0  1 2523
## 86     0  0 2551
## 87     1  0 2557
## 88     1  1 2594
## 89     1  1 2600
## 91     0  0 2622
```

```
select=dplyr::select
# select(birthwt,age:smoke)
head(select(birthwt,age:smoke))
```

```
##    age lwt race smoke
## 85  19 182    2     0
## 86  33 155    3     0
## 87  20 105    1     1
## 88  21 108    1     1
## 89  18 107    1     1
## 91  21 124    3     0
```

```
# select(birthwt,-(age:smoke))
head(select(birthwt,-(age:smoke)))
```

```
##    low ptl ht ui ftv  bwt
## 85   0   0  0  1   0 2523
## 86   0   0  0  0   3 2551
## 87   0   0  0  0   1 2557
## 88   0   0  0  1   2 2594
## 89   0   0  0  1   0 2600
## 91   0   0  0  0   0 2622
```

Il existe de nombreuses fonctions d'assistance qui peuvent être utilisées dans la fonction select (): * starts_with ("abc"): correspond aux noms commençant par "abc" * ends_with ("xui") correspond aux noms se terminant par "xui" * contient ("jkw") correspond aux noms contenant "jkw". * num_range ("x", 1: 3) correspond à x1, x2, x3.

select () peut également être utilisé en conjonction avec la fonction d'assistance everything(). C'est très utile lorsque vous avez beaucoup de variables et que vous souhaitez vous déplacer au début (le plus à gauche) du bloc de données.

```
# select(birthwt,age,everything())
head(select(birthwt,age,everything()))
```

```
##    age low lwt race smoke ptl ht ui ftv  bwt
## 85  19   0 182    2     0   0  0  1   0 2523
## 86  33   0 155    3     0   0  0  0   3 2551
## 87  20   0 105    1     1   0  0  0   1 2557
## 88  21   0 108    1     1   0  0  1   2 2594
## 89  18   0 107    1     1   0  0  1   0 2600
## 91  21   0 124    3     0   0  0  0   0 2622
```

summarize(): Calcule la valeur récapitulative

summarize(), qui réduit un bloc de données en une seule ligne:

```
summarize(birthwt,delay=mean(age,na.rm = TRUE))
```

```
##     delay
## 1 23.2381
```

À moins que nous n'associons summary () avec group_by (), summary () semble inutile. Cette opération transférera l'unité d'analyse de l'ensemble de données vers un seul groupe. Ensuite, lorsque vous utilisez le verbe dplyr pour opérer sur le bloc de données groupées, il effectuera automatiquement des calculs de regroupement. Par exemple, nous voulons regrouper par race pour obtenir l'âge moyen de chaque race:

```
by_race=group_by(birthwt,race)
summarize(by_race,delay=mean(age,na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 2
##    race delay
##   <int> <dbl>
## 1     1  24.3
## 2     2  21.5
## 3     3  22.4
```

L'utilisation combinée de group_by () et de summary () peut effectuer des regroupements et des résumés, ce qui nous est très utile pour traiter des données multivariées.