Dplyr.R

aramaciabarrado

Fri Jun 24 13:07:35 2016

```
#Realizado por Araceli Macía Barrado
#Ejercicio de dplyr
library(dplyr)
```

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

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

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

```
library(ggplot2)

#Vemos el dataset de diamonds
tbl_df(diamonds)
```

```
## Source: local data frame [53,940 x 10]
##
##    carat       cut  color clarity depth table price     x     y     z
##    (dbl)    (fctr) (fctr)  (fctr) (dbl) (dbl) (int) (dbl) (dbl) (dbl)
## 1   0.23     Ideal      E     SI2  61.5    55   326  3.95  3.98  2.43
## 2   0.21   Premium      E     SI1  59.8    61   326  3.89  3.84  2.31
## 3   0.23      Good      E     VS1  56.9    65   327  4.05  4.07  2.31
## 4   0.29   Premium      I     VS2  62.4    58   334  4.20  4.23  2.63
## 5   0.31      Good      J     SI2  63.3    58   335  4.34  4.35  2.75
## 6   0.24 Very Good      J    VVS2  62.8    57   336  3.94  3.96  2.48
## 7   0.24 Very Good      I    VVS1  62.3    57   336  3.95  3.98  2.47
## 8   0.26 Very Good      H     SI1  61.9    55   337  4.07  4.11  2.53
## 9   0.22      Fair      E     VS2  65.1    61   337  3.87  3.78  2.49
## 10  0.23 Very Good      H     VS1  59.4    61   338  4.00  4.05  2.39
## ..   ...       ...    ...     ...   ...   ...   ...   ...   ...   ...
```

```
#1) Filtrar los diamantes con corte “Ideal”.

corteIdeal<-filter(diamonds, cut == "Ideal")
#Vemos a ver como ha quedado el DataSet Ideal
head(corteIdeal)
```

```
## Source: local data frame [6 x 10]
##
##   carat    cut  color clarity depth table price     x     y     z
##   (dbl) (fctr) (fctr)  (fctr) (dbl) (dbl) (int) (dbl) (dbl) (dbl)
## 1  0.23  Ideal      E     SI2  61.5    55   326  3.95  3.98  2.43
## 2  0.23  Ideal      J     VS1  62.8    56   340  3.93  3.90  2.46
## 3  0.31  Ideal      J     SI2  62.2    54   344  4.35  4.37  2.71
## 4  0.30  Ideal      I     SI2  62.0    54   348  4.31  4.34  2.68
## 5  0.33  Ideal      I     SI2  61.8    55   403  4.49  4.51  2.78
## 6  0.33  Ideal      I     SI2  61.2    56   403  4.49  4.50  2.75
```

```
#2)   Seleccionar las columnas carat, cut, color, price y clarity

SelColumnas <- select(diamonds,carat,cut,color,price,clarity)
head(SelColumnas)
```

```
## Source: local data frame [6 x 5]
##
##   carat       cut  color price clarity
##   (dbl)    (fctr) (fctr) (int)  (fctr)
## 1  0.23     Ideal      E   326     SI2
## 2  0.21   Premium      E   326     SI1
## 3  0.23      Good      E   327     VS1
## 4  0.29   Premium      I   334     VS2
## 5  0.31      Good      J   335     SI2
## 6  0.24 Very Good      J   336    VVS2
```

```
#3)   Crear una nueva columna precio/quilate.

NewCol <- mutate(SelColumnas, Pre_qui = price/carat)
head(NewCol)
```

```
## Source: local data frame [6 x 6]
##
##   carat       cut  color price clarity  Pre_qui
##   (dbl)    (fctr) (fctr) (int)  (fctr)    (dbl)
## 1  0.23     Ideal      E   326     SI2 1417.391
## 2  0.21   Premium      E   326     SI1 1552.381
## 3  0.23      Good      E   327     VS1 1421.739
## 4  0.29   Premium      I   334     VS2 1151.724
## 5  0.31      Good      J   335     SI2 1080.645
## 6  0.24 Very Good      J   336    VVS2 1400.000
```

```
#4)    Agrupar los diamantes por color.

group_by(diamonds, color)
```

```
## Source: local data frame [53,940 x 10]
## Groups: color [7]
##
##    carat       cut  color clarity depth table price     x     y     z
##    (dbl)    (fctr) (fctr)  (fctr) (dbl) (dbl) (int) (dbl) (dbl) (dbl)
## 1   0.23     Ideal      E     SI2  61.5    55   326  3.95  3.98  2.43
## 2   0.21   Premium      E     SI1  59.8    61   326  3.89  3.84  2.31
## 3   0.23      Good      E     VS1  56.9    65   327  4.05  4.07  2.31
## 4   0.29   Premium      I     VS2  62.4    58   334  4.20  4.23  2.63
## 5   0.31      Good      J     SI2  63.3    58   335  4.34  4.35  2.75
## 6   0.24 Very Good      J    VVS2  62.8    57   336  3.94  3.96  2.48
## 7   0.24 Very Good      I    VVS1  62.3    57   336  3.95  3.98  2.47
## 8   0.26 Very Good      H     SI1  61.9    55   337  4.07  4.11  2.53
## 9   0.22      Fair      E     VS2  65.1    61   337  3.87  3.78  2.49
## 10  0.23 Very Good      H     VS1  59.4    61   338  4.00  4.05  2.39
## ..   ...       ...    ...     ...   ...   ...   ...   ...   ...   ...
```

```
#para ver el precio maximo de cada color
summarise( group_by(diamonds, color), max(price))
```

```
## Source: local data frame [7 x 2]
##
##    color max(price)
##   (fctr)      (int)
## 1      D      18693
## 2      E      18731
## 3      F      18791
## 4      G      18818
## 5      H      18803
## 6      I      18823
## 7      J      18710
```

```
# 5)  Calcular la media del precio/quilate para cada uno de los grupos
anteriores.

# Utilizo el dataset antes generado con la nueva columna.
summarise( group_by(NewCol, color), mean(Pre_qui))
```

```
## Source: local data frame [7 x 2]
##
##    color mean(Pre_qui)
##   (fctr)         (dbl)
## 1      D      3952.564
## 2      E      3804.611
## 3      F      4134.731
## 4      G      4163.412
## 5      H      4008.027
## 6      I      3996.402
## 7      J      3825.649
```

```
#6)    Ordenar por precio/quilate de forma descendente.

total <- NewCol %>% group_by( color) %>% summarise( med = mean(Pre_qui))
%>% arrange(desc(med))
print(total)
```

```
## Source: local data frame [7 x 2]
##
##    color      med
##   (fctr)    (dbl)
## 1      G 4163.412
## 2      F 4134.731
## 3      H 4008.027
## 4      I 3996.402
## 5      D 3952.564
## 6      J 3825.649
## 7      E 3804.611
```