Some practices of dplyr 20.09.2021

You Zhou

20/09/2021

Contents

Introduction

In this coding tasks, we will do some practices about using the `dplyr` package. Before moving to the questions, please make sure that you load the library and use the correct test data set ;)

```
## loading the required package
library(tidyverse) ## this library is a libraries collection which includes the 'dplyr'
## loading the correct data set for the incoming questions
data("diamonds")
head(diamonds)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <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  4.2   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
```

Practice 1

Extract the carat, cut, color and price column from the data.

Tips: `select` function may help.

```
df <- diamonds %>% select(carat:color, price)
df
```

```
## # A tibble: 53,940 x 4
##    carat cut       color price
##    <dbl> <ord>     <ord> <int>
##  1  0.23 Ideal     E       326
##  2  0.21 Premium   E       326
##  3  0.23 Good      E       327
##  4  0.29 Premium   I       334
##  5  0.31 Good      J       335
##  6  0.24 Very Good J       336
##  7  0.24 Very Good I       336
##  8  0.26 Very Good H       337
##  9  0.22 Fair      E       337
## 10  0.23 Very Good H       338
## # ... with 53,930 more rows
```

Practice 2

Making a new column to show the number of row for all the diamonds with function `mutate`.

```
df <- df %>% mutate(row_number = row_number())
df
```

```
## # A tibble: 53,940 x 5
##    carat cut       color price row_number
##    <dbl> <ord>     <ord> <int>      <int>
##  1  0.23 Ideal     E       326          1
##  2  0.21 Premium   E       326          2
##  3  0.23 Good      E       327          3
##  4  0.29 Premium   I       334          4
##  5  0.31 Good      J       335          5
##  6  0.24 Very Good J       336          6
##  7  0.24 Very Good I       336          7
##  8  0.26 Very Good H       337          8
##  9  0.22 Fair      E       337          9
## 10  0.23 Very Good H       338         10
## # ... with 53,930 more rows
```

Practice 3

Let's first make a unique ID for each diamond.

```
set.seed(1)
diamonds$ID <- sample(1:nrow(diamonds), nrow(diamonds), replace = FALSE)
```

And we have a new data which have the column ID, virtual price and some number.

```
df_virtual <- data.frame(
  ID = 1:nrow(diamonds),
  virtual_price = nrow(diamonds):1,
  some_number = nrow(diamonds):1
```

```
)
head(df_virtual)
```

```
##   ID virtual_price some_number
## 1  1         53940       53940
## 2  2         53939       53939
## 3  3         53938       53938
## 4  4         53937       53937
## 5  5         53936       53936
## 6  6         53935       53935
```

Now the question is how can we joint the `df_virtual` to the original `diamonds` data frame.

```
df <- left_join(diamonds, df_virtual, by = "ID") %>%
  select(!(x:z))
head(df)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price    ID virtual_price some_number
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <int>         <int>       <int>
## 1  0.23 Ideal     E     SI2      61.5    55   326 24388         29553       29553
## 2  0.21 Premium   E     SI1      59.8    61   326 43307         10634       10634
## 3  0.23 Good      E     VS1      56.9    65   327  4050         49891       49891
## 4  0.29 Premium   I     VS2      62.4    58   334 11571         42370       42370
## 5  0.31 Good      J     SI2      63.3    58   335 25173         28768       28768
## 6  0.24 Very Go~  J     VVS2     62.8    57   336 32618         21323       21323
```

Practice 4

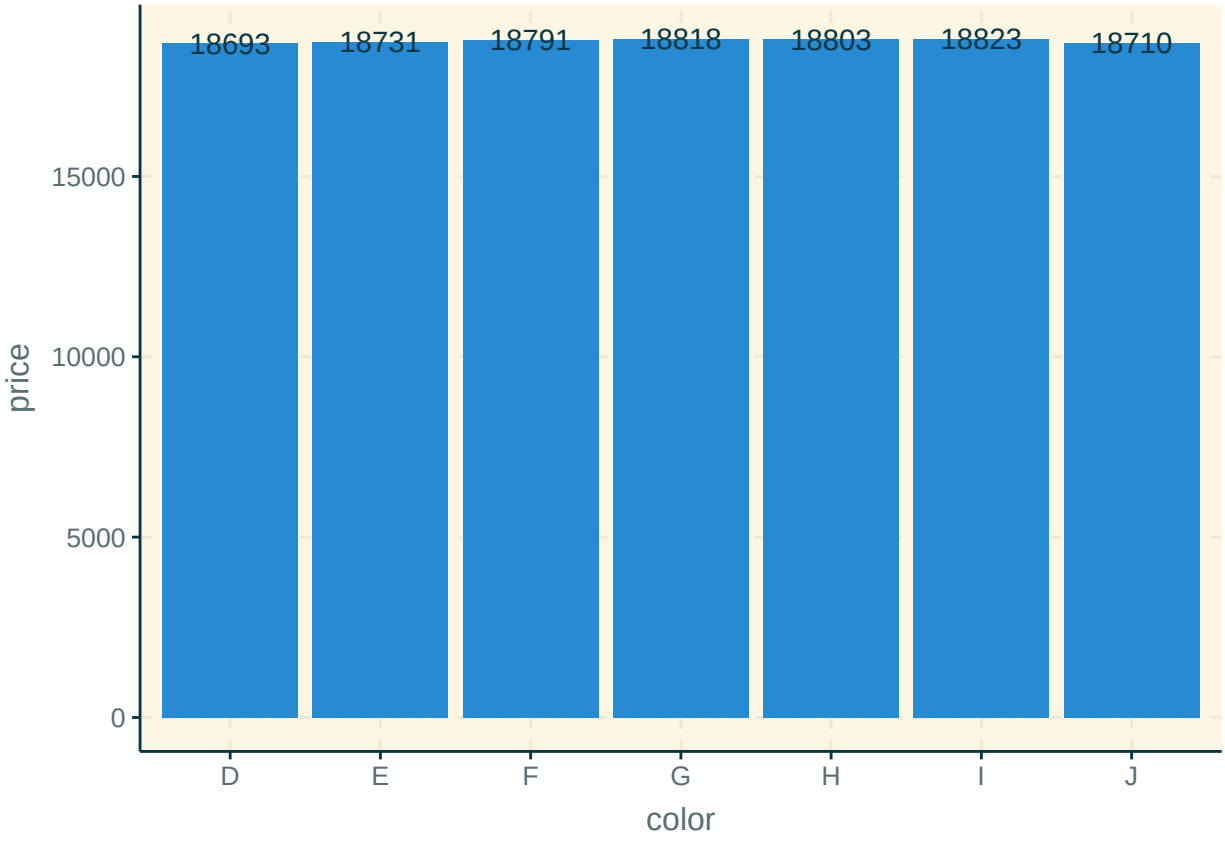
Find out the maximum `price` for different `color` and convert the result into a bar plot.

Tips: `group_by` function may help.

```
## this package is unnecessary but can make the plot nicer ; )
library(ggthemr)
ggthemr('solarized')

df <- diamonds %>% group_by(color) %>%
  summarise(price = max(price))

ggplot(df, aes(x = color, y = price, label = price)) +
  geom_bar(stat = "identity") +
  geom_text()
```

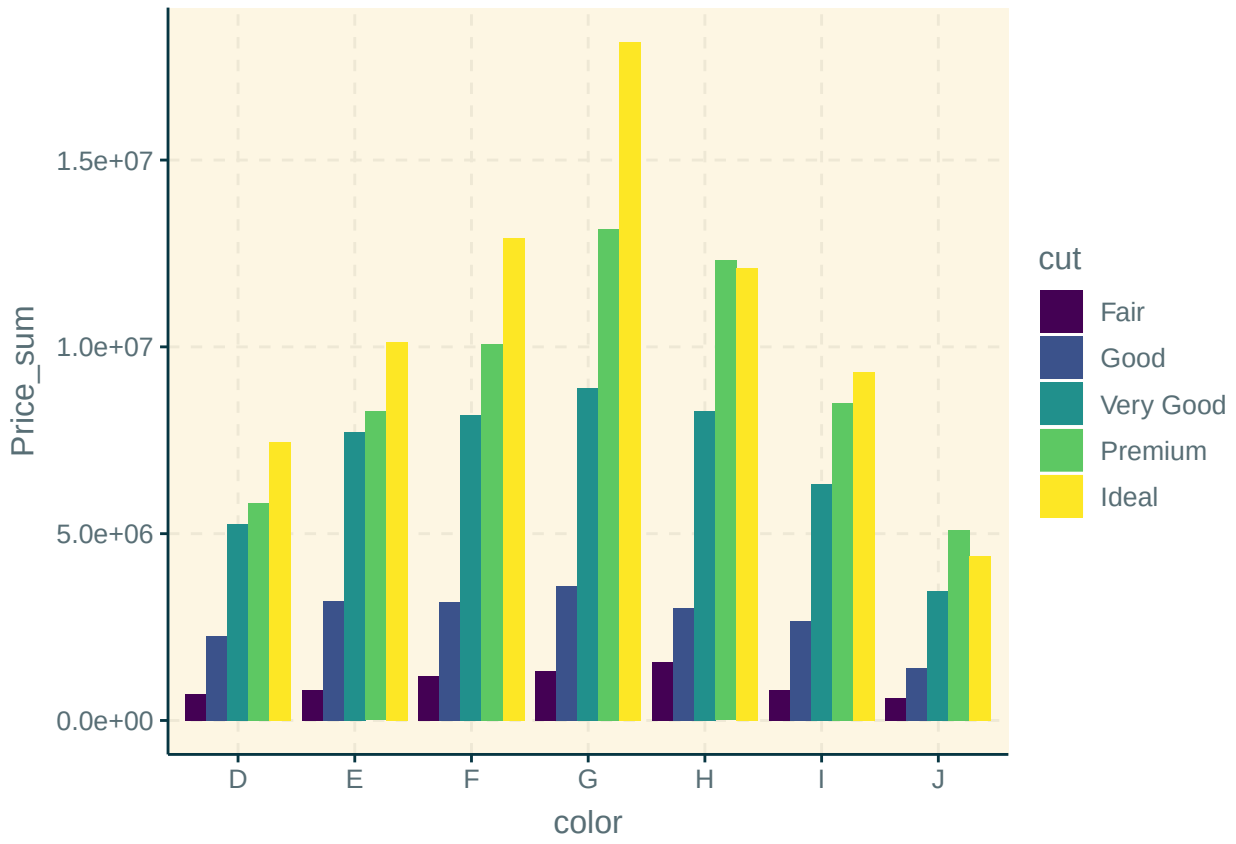


Practice 5

Sum up all the price for the diamonds with the same `cut` and `color`. And using a `dodge bar` chart for the visualization.

```
df <- diamonds %>% group_by(cut, color) %>%
  summarise(Price_sum = sum(price))

ggplot(df, aes(x = color, y = Price_sum, fill = cut)) +
  geom_bar(stat = "identity", position = "dodge")
```

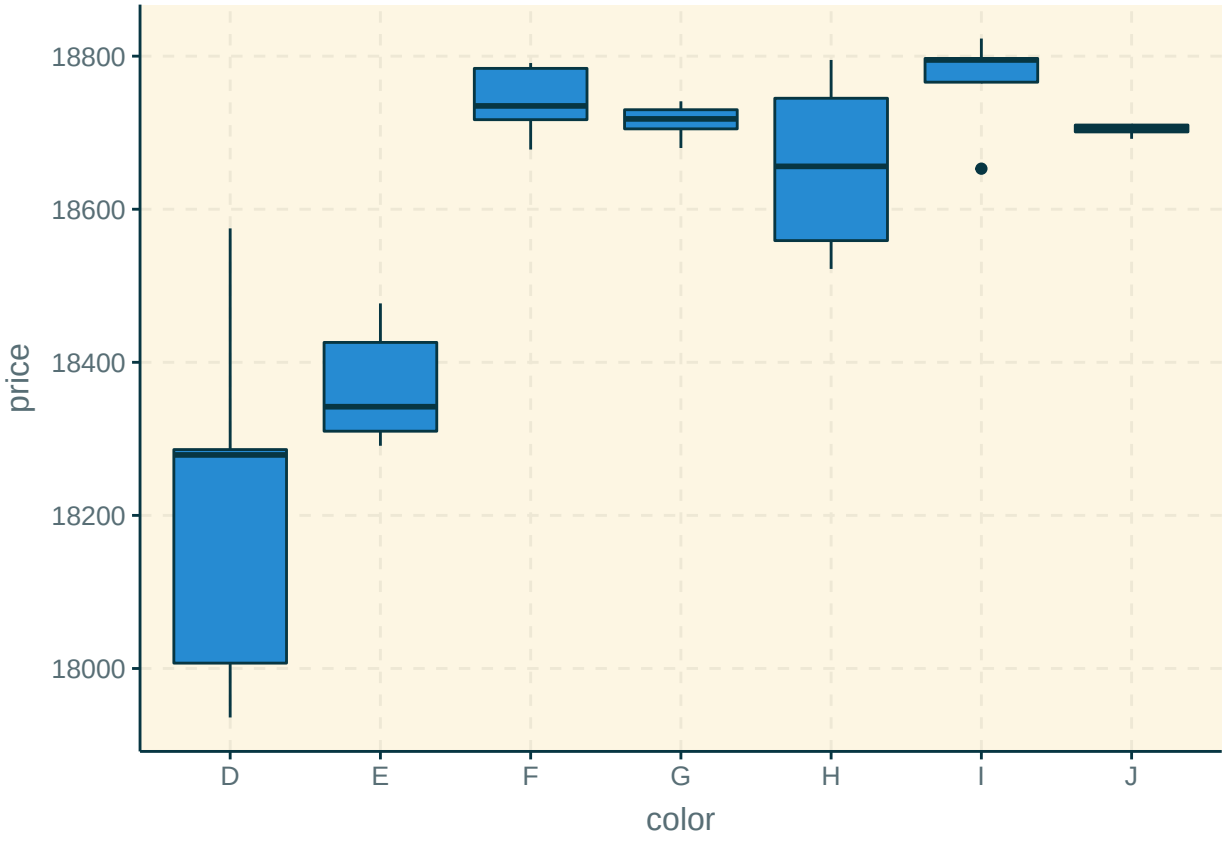


Practice 6

For the diamonds with the **Premium** `cut`, extract the **top 5** diamonds with the highest `price` for each `color`. Then make a *boxplot* to show this result.

```
df <- diamonds %>% arrange(desc(price)) %>%
  filter(cut == "Premium") %>%
  group_by(color) %>%
  slice_max(order_by = price, n = 5)

ggplot(df, aes(x = color, y = price)) +
  geom_boxplot()
```



Practice 7

Count the number for the diamonds which have the same `cut` and `color`.

```
df <- diamonds %>% group_by(cut, color) %>%
  summarise(Number = n())
head(df)
```

```
## # A tibble: 6 x 3
## # Groups:   cut [1]
##   cut   color Number
##   <ord> <ord>  <int>
## 1 Fair  D        163
## 2 Fair  E        224
## 3 Fair  F        312
## 4 Fair  G        314
## 5 Fair  H        303
## 6 Fair  I        175
```