Tutorial 4: Data Manipulation

Mr. Zebin Yang

Oct 9, 2018

Before we start

Install the library "dplyr"

- install.packages("dplyr")
- library(dplyr)
- help(package="dplyr")

```
library(dplyr)
library(ggplot2)
```

Before we start

Diamond Dataset Introduction

- price: price in US dollars ($326–$18,823)
- carat: weight of the diamond (0.2–5.01)
- cut: quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- color: diamond colour, from J (worst) to D (best)
- claritya: measurement of how clear the diamond is
- x: length in mm (0–10.74)
- y: width in mm (0–58.9)
- z: depth in mm (0–31.8)
- depth: total depth percentage = 2 * z / (x + y) (43–79)
- table: width of top of diamond relative to widest point (43–95)

```
head(diamonds, 3)
```

```
## # A tibble: 3 x 10
##   carat cut     color clarity depth table price     x     y     z
##   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.230 Ideal   E     SI2      61.5   55.   326  3.95  3.98  2.43
## 2 0.210 Premium E     SI1      59.8   61.   326  3.89  3.84  2.31
## 3 0.230 Good    E     VS1      56.9   65.   327  4.05  4.07  2.31
```

Outline

- R:dplyr package
  - filter() to select observations
  - arrange() to order observations
  - mutate() to add new variables
  - group_by() to group variables for summarise
- Pipes %>%
- Exercises

R:dplyr package

filter()

- allows you to select a subset of rows.
  - Mathematical relationship: ==, <, <=, >, >=.
  - Logical Operators: &, |, !.
  - %in%, is.na(), between().

```
filter(diamonds, cut == "Premium")[1:6,]
```

```
## # 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 0.210 Premium E     SI1      59.8   61.   326  3.89  3.84  2.31
## 2 0.290 Premium I     VS2      62.4   58.   334  4.20  4.23  2.63
## 3 0.220 Premium F     SI1      60.4   61.   342  3.88  3.84  2.33
## 4 0.200 Premium E     SI2      60.2   62.   345  3.79  3.75  2.27
## 5 0.320 Premium E     I1       60.9   58.   345  4.38  4.42  2.68
## 6 0.240 Premium I     VS1      62.5   57.   355  3.97  3.94  2.47
```

R:dplyr package

filter()

- allows you to select a subset of rows.
  - Mathematical relationship: ==, <, <=, >, >=.
  - Logical Operators: &, |, !.
  - %in%, is.na(), between().

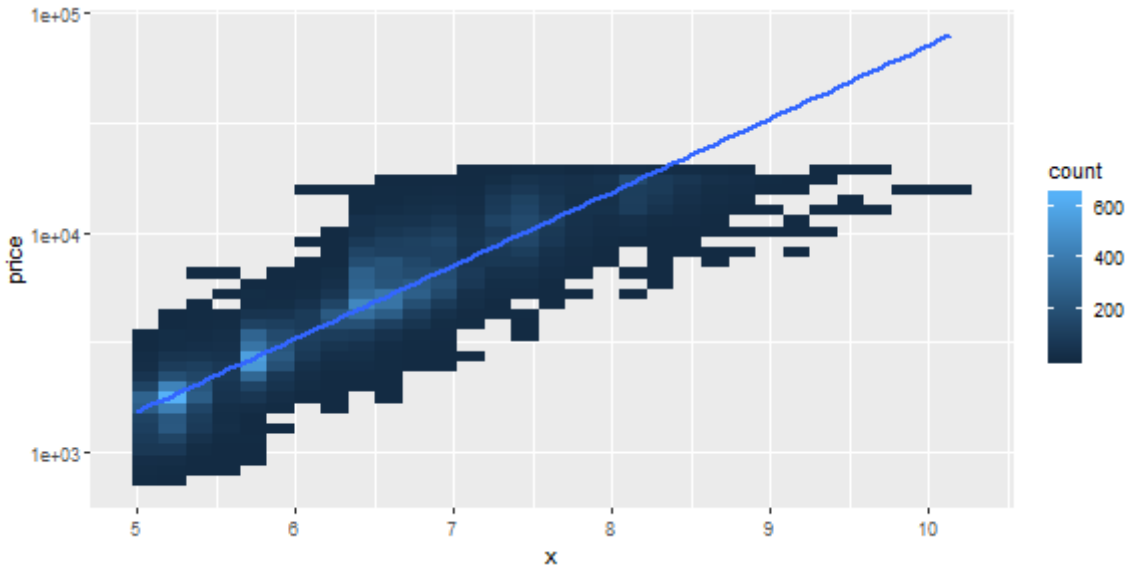
```
filter_diamonds = filter(diamonds, x>=5 &
                           cut %in% c("Premium","Ideal"))
head(filter_diamonds)
```

```
## # 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 0.700 Ideal   E     SI1      62.5   57.  2757  5.70  5.72  3.57
## 2 0.700 Ideal   G     VS2      61.6   56.  2757  5.70  5.67  3.50
## 3 0.800 Premium H     SI1      61.5   58.  2760  5.97  5.93  3.66
## 4 0.750 Premium E     SI1      59.9   54.  2760  6.00  5.96  3.58
## 5 0.740 Ideal   G     SI1      61.6   55.  2760  5.80  5.85  3.59
## 6 0.750 Premium G     VS2      61.7   58.  2760  5.85  5.79  3.59
```

R:dplyr package

filter()

```
ggplot(filter_diamonds, aes(x,price)) +
  geom_bin2d() +
  scale_y_continuous(trans = "log10") +
  geom_smooth(method = "lm")
```



R:dplyr package

arrange()

- works similarly to except that instead of filtering or selecting rows, it reorders them.

```
head(arrange(diamonds, cut, depth)) # by default ascending
```

```
## # 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 1.00  Fair  G     SI1      43.0   59.  3634  6.32  6.27  3.97
## 2 1.00  Fair  G     VS2      44.0   53.  4032  6.31  6.24  4.12
## 3 1.43  Fair  I     VS1      50.8   60.  6727  7.73  7.25  3.93
## 4 0.300 Fair  E     VVS2     51.0   67.   945  4.67  4.62  2.37
## 5 0.700 Fair  D     SI1      52.2   65.  1895  6.04  5.99  3.14
## 6 0.370 Fair  F     IF       52.3   61.  1166  4.96  4.91  2.58
```

R:dplyr package

arrange()

```
head(arrange(diamonds, desc(price)))
```

```
## # 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  2.29 Premium   I     VS2      60.8   60. 18823  8.50  8.47  5.16
## 2  2.00 Very Good G     SI1      63.5   56. 18818  7.90  7.97  5.04
## 3  1.51 Ideal     G     IF       61.7   55. 18806  7.37  7.41  4.56
## 4  2.07 Ideal     G     SI2      62.5   55. 18804  8.20  8.13  5.11
## 5  2.00 Very Good H     SI1      62.8   57. 18803  7.95  8.00  5.01
## 6  2.29 Premium   I     SI1      61.8   59. 18797  8.52  8.45  5.24
```

R:dplyr package

mutate()

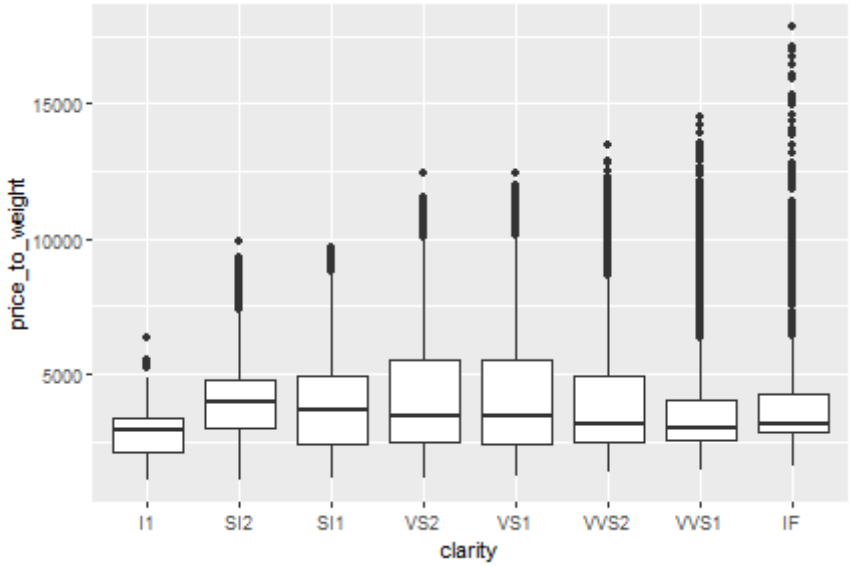
- It is often useful to add new columns that are functions of existing columns. This is the job of mutate():

```
new_diamonds = mutate(diamonds, price_to_weight  = price/carat)
head(new_diamonds[,"price_to_weight"])
```

```
## # A tibble: 6 x 1
##   price_to_weight
##             <dbl>
## 1           1417.
## 2           1552.
## 3           1422.
## 4           1152.
## 5           1081.
## 6           1400.
```

R:dplyr package
mutate()

```
ggplot(new_diamonds, aes(x = clarity, y = price_to_weight)) +
  geom_boxplot()
```



R:dplyr package
group_by()

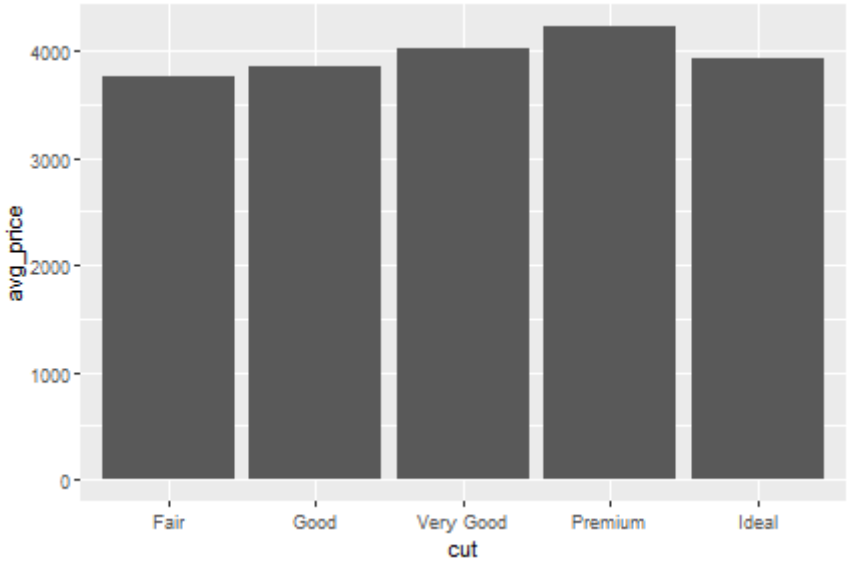
- The group_by() function splits a dataset into several groups, and then we can calculate for example, the mean, max, min of each group.

```
# Grouped by chas, and then get the sample size of each group
summarise(group_by(new_diamonds, factor(clarity)), n())
```

```
## # A tibble: 8 x 2
##   `factor(clarity)` `n()`
##   <ord>             <int>
## 1 I1                  741
## 2 SI2                9194
## 3 SI1               13065
## 4 VS2               12258
## 5 VS1                8171
## 6 VVS2               5066
## 7 VVS1               3655
## 8 IF                 1790
```

R:dplyr package
group_by()

```
# Grouped by chas, and then get the mean of each group
sumary_diamond1 = summarise(group_by(new_diamonds, factor(cut)),
                            mean(price_to_weight))
colnames(sumary_diamond1) = c("cut","avg_price")
ggplot(sumary_diamond1, aes(x=cut, y=avg_price)) +
geom_bar(stat="identity")
```

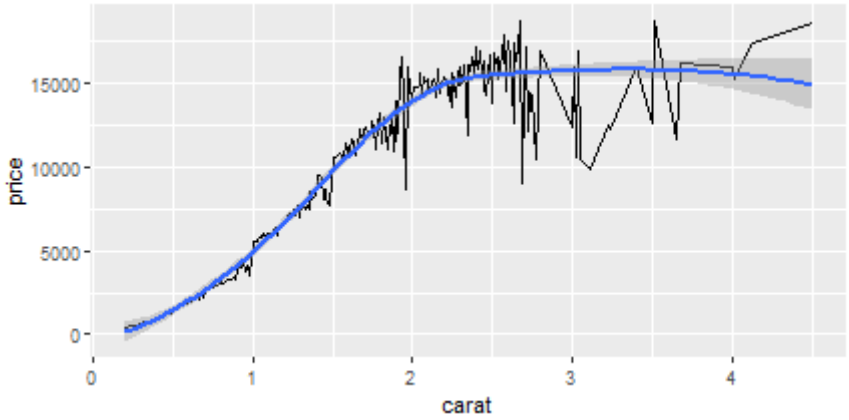


Pipeline %>%

Powerful trick for coding a sequence of operations.

- Example 1

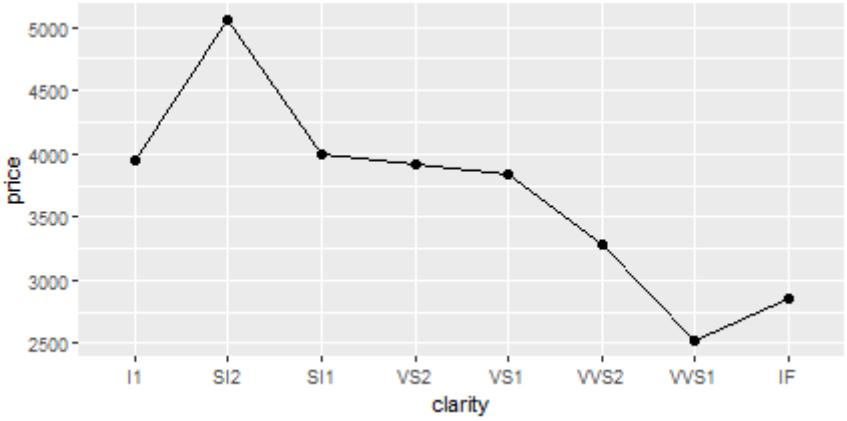
```
diamonds %>%
  filter(x>0, y>0, y<20) %>%
  mutate(sym=x-y, size=pi*((x+y)/2/2)^2) %>%
  filter(abs(sym)<0.2) %>%
  group_by(carat) %>%
  summarize(price=mean(price)) %>%
  ggplot(aes(carat, price)) + geom_line() + geom_smooth()
```



Pipeline %>%

- Example 2

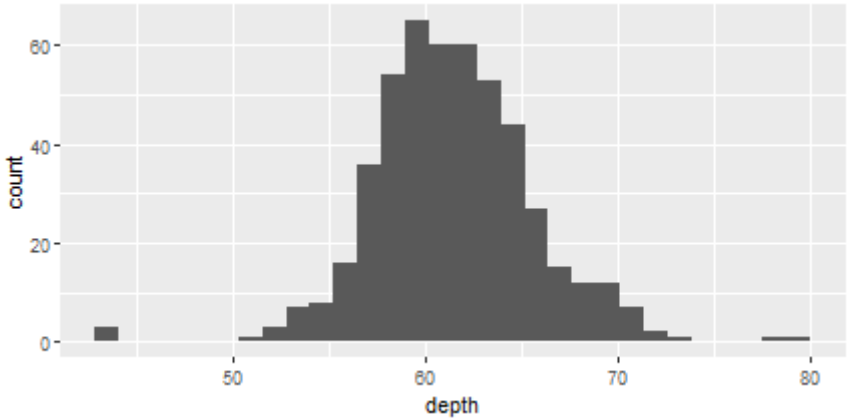
```
diamonds %>%
  filter(x>0, y>0, y<20) %>%
  mutate(sym=x-y, size=pi*((x+y)/2/2)^2) %>%
  filter(abs(sym)<0.2) %>%
  group_by(clarity) %>%
  summarize(price=mean(price)) %>%
  ggplot( aes(clarity, price))+
  geom_line(aes(group=1))+ geom_point(size=2)
```



Pipeline %>%

- Example 3

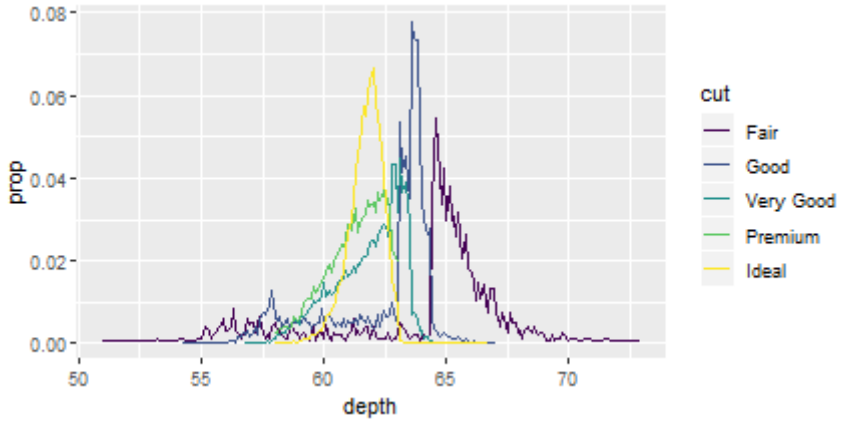
```
diamonds %>%
  filter(x>0, y>0, y<20) %>%
  mutate(sym=x-y, size=pi*((x+y)/2/2)^2) %>%
  filter(abs(sym)<0.2) %>%
  group_by(cut, depth) %>%
  summarize(n=n()) %>%
  ggplot(aes(depth)) + geom_histogram(bins=30)
```



Pipeline %>%

- Example 4

```
diamonds %>%
  filter(x>0, y>0, y<20) %>%
  mutate(sym=x-y, size=pi*((x+y)/2/2)^2) %>%
  filter(abs(sym)<0.2) %>%
  group_by(cut, depth) %>%
  summarize(n=n()) %>%
  filter(depth>50, depth<75) %>%
  mutate( prop=n/sum(n)) %>%
  ggplot(aes(depth, prop, color=cut)) + geom_line()
```



Exercises

- We have two built-in data table2.
- They record the number of TB cases documented by the World Health Organization in Afghanistan, Brazil, and China between 1999 and 2000.

```
library(tidyr)
table2
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <int> <chr>           <int>
##  1 Afghanistan  1999 cases             745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000 cases            2666
##  4 Afghanistan  2000 population   20595360
##  5 Brazil       1999 cases           37737
##  6 Brazil       1999 population  172006362
##  7 Brazil       2000 cases           80488
##  8 Brazil       2000 population  174504898
##  9 China        1999 cases          212258
## 10 China        1999 population 1272915272
## 11 China        2000 cases          213766
## 12 China        2000 population 1280428583
```

Exercises

- Extract the number of cases per country per year.
- Extract the matching population per country per year.
- Divide cases by population, and multiply by 10000.

Q&A