Lesson_3: Chapter 5 Data transformation

Hao Wang

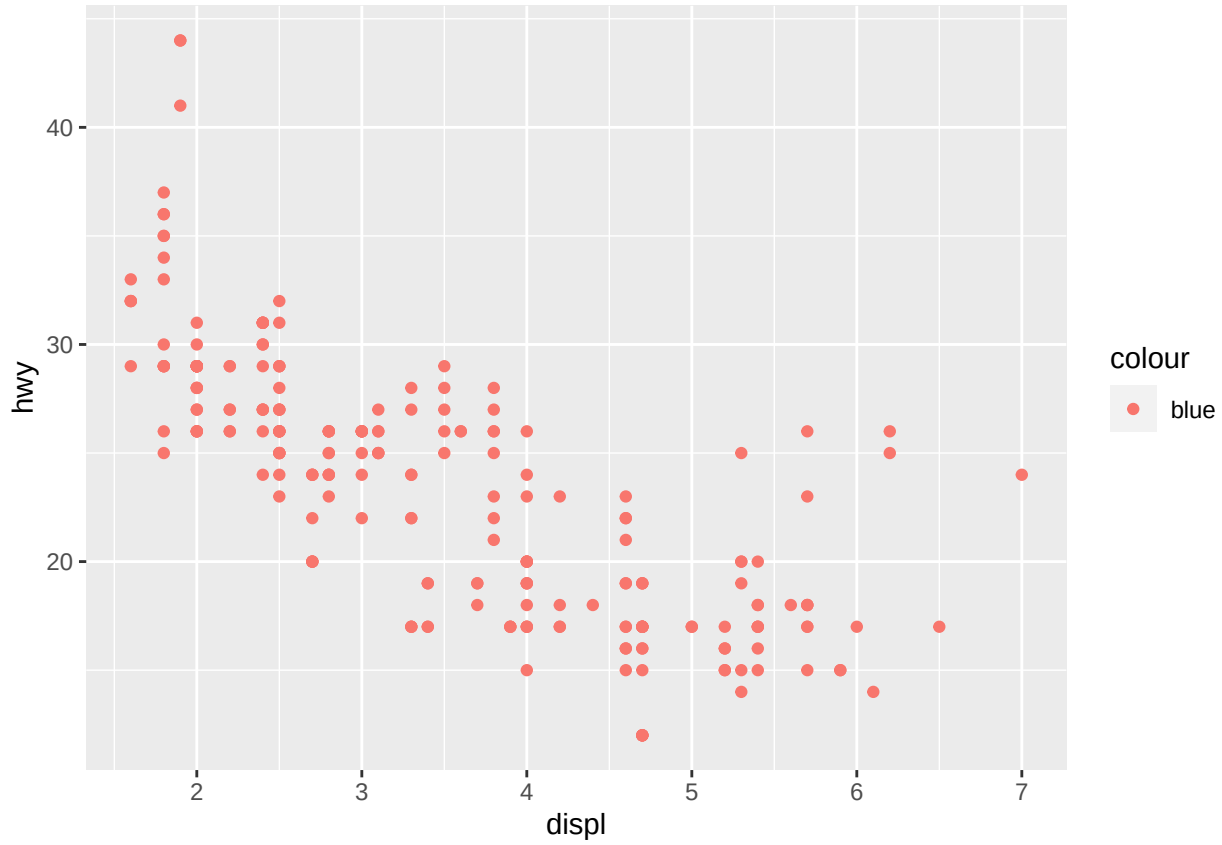
1/26/2022

1. Review Chapter 3.

$3.3.1 Exercises

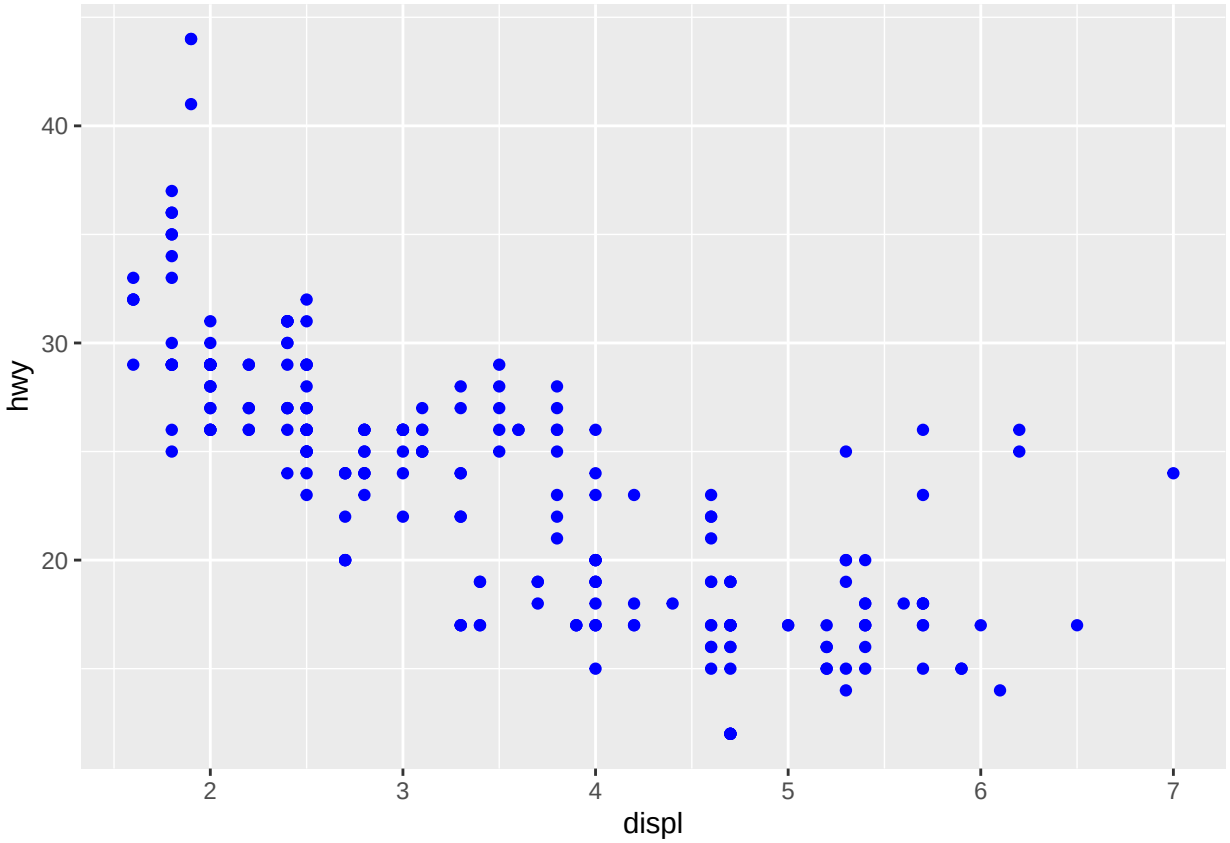
1.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = "blue"))
```



Ans: The error is "blue" is not a column in mpg. You need to put color outside of aes.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```

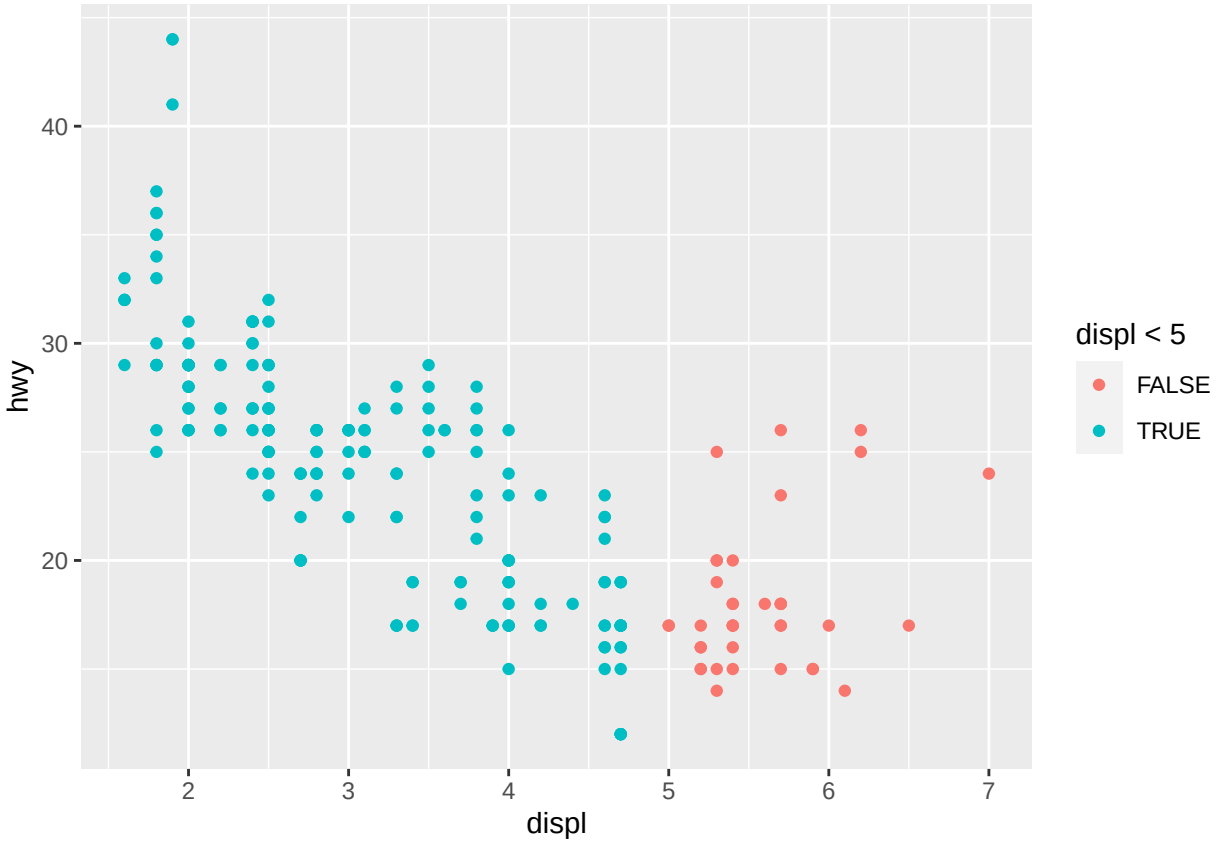


3. Map a continuous variable to color, size, and shape. How do these aesthetics behave differently for categorical vs. continuous variables?

Ans: Continuous variables can be mapped to *color* and *size*, only categorical variables can be mapped to *shape*.

6. What happens if you map an aesthetic to something other than a variable name, like aes(colour = displ < 5)? Note, you'll also need to specify x and y.

```
ggplot(mpg, aes(x = displ, y = hwy, colour = displ < 5)) +
  geom_point()
```



$3.5.1 Exercises

1. What happens if you facet on a continuous variable?

Ans: The continuous variable is converted to a categorical variable, and the plot contains a facet for each distinct value.

2. Chapter 5 Data transformation

We will learn how to use the package **dplyr**.

-Key functions in **dplyr**:

1. filter(): selct rows.
2. select(): select columns.
3. arrange(): sort by.
4. mutate(): create new columns.
5. summmarise(): usually used with group_by() to create summarized tables by columns.

Explore data set flights in nycflights13:

```
?flights
```

```
## starting httpd help server ... done
```

```
str(flights)
```

```
## tibble [336,776 x 19] (S3: tbl_df/tbl/data.frame)
##  $ year          : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int [1:336776] 517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int [1:336776] 515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int [1:336776] 830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int [1:336776] 819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr [1:336776] "UA" "UA" "AA" "B6" ...
##  $ flight        : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time      : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : num [1:336776] 1400 1416 1089 1576 762 ...
##  $ hour          : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

2.1 filter()

```
filter(flights, day == 1, dep_delay >= 180)
```

```
## # A tibble: 131 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      848           1835       853     1001           1950
##  2  2013     1     1     1815           1325       290     2120           1542
##  3  2013     1     1     1842           1422       260     1958           1535
##  4  2013     1     1     2006           1630       216     2230           1848
##  5  2013     1     1     2115           1700       255     2330           1920
##  6  2013     1     1     2205           1720       285       46           2040
##  7  2013     1     1     2312           2000       192       21           2110
##  8  2013     1     1     2343           1724       379      314           1938
##  9  2013    10     1     1342            815       327     1458            947
## 10  2013    11     1     1310            845       265     1423           1030
## # ... with 121 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

2.2 comparison operators and logical operators, NA

==, !=, etc.

&, |, !

%in%

is.na()

2.3 arrange()

```
flights_filtered <- filter(flights, day == 1, dep_delay >= 180)
arrange(flights_filtered, month, dep_delay)
```

```
## # A tibble: 131 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1     2312           2000       192       21           2110
##  2  2013     1     1     2006           1630       216     2230           1848
##  3  2013     1     1     2115           1700       255     2330           1920
##  4  2013     1     1     1842           1422       260     1958           1535
##  5  2013     1     1     2205           1720       285       46           2040
##  6  2013     1     1     1815           1325       290     2120           1542
##  7  2013     1     1     2343           1724       379      314           1938
##  8  2013     1     1      848           1835       853     1001           1950
##  9  2013     2     1     1133            822       191     1324           1019
## 10  2013     2     1     1518           1205       193     1724           1345
## # ... with 121 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

2.4 select()

```
select(flights, year, month, day)
```

```
## # A tibble: 336,776 x 3
##     year month   day
##    <int> <int> <int>
##  1  2013     1     1
##  2  2013     1     1
##  3  2013     1     1
##  4  2013     1     1
##  5  2013     1     1
##  6  2013     1     1
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # ... with 336,766 more rows
```

There are number of ways to select columns. It can be very useful when you have many columns.

2.5 mutate()

```
mutate(flights_filtered,
       gain = dep_delay - arr_delay,
       speed = distance / air_time * 60
       )
```

```
## # A tibble: 131 x 21
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      848           1835       853     1001           1950
##  2  2013     1     1     1815           1325       290     2120           1542
##  3  2013     1     1     1842           1422       260     1958           1535
##  4  2013     1     1     2006           1630       216     2230           1848
##  5  2013     1     1     2115           1700       255     2330           1920
##  6  2013     1     1     2205           1720       285       46           2040
##  7  2013     1     1     2312           2000       192       21           2110
##  8  2013     1     1     2343           1724       379      314           1938
##  9  2013    10     1     1342            815       327     1458            947
## 10  2013    11     1     1310            845       265     1423           1030
## # ... with 121 more rows, and 13 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>,
## #   gain <dbl>, speed <dbl>
```

A lot of functions can be used in mutate() to create new columns.

2.6 summarise() with group_by()

Mean of dep_delay.

```
summarise(flights, delay = mean(dep_delay, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   delay
##   <dbl>
## 1  12.6
```

Question: what would happen if na.rm = TRUE is removed?

2.7 Use the pipe to connect operations

The pipe %>% is a very useful operator in tidyverse.

```
delays <- flights %>%
  group_by(dest) %>%
  summarise(
    count = n(),
    dist = mean(distance, na.rm = TRUE),
```

```
    delay = mean(arr_delay, na.rm = TRUE)
  ) %>%
  filter(count > 20, dest != "HNL")

delays
```

```
## # A tibble: 96 x 4
##    dest  count  dist delay
##    <chr> <int> <dbl> <dbl>
##  1 ABQ     254 1826   4.38
##  2 ACK     265  199   4.85
##  3 ALB     439  143  14.4
##  4 ATL   17215  757. 11.3
##  5 AUS    2439 1514.  6.02
##  6 AVL     275  584.  8.00
##  7 BDL     443  116   7.05
##  8 BGR     375  378   8.03
##  9 BHM     297  866. 16.9
## 10 BNA    6333  758. 11.8
## # ... with 86 more rows
```

Explore the summary functions in R.

You can group by multiple variables.

Here is the summary table for number of flights every month, day.

```
daily <- flights %>%
  group_by(year, month, day) %>%
  summarise(flights = n())
```

```
daily
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [12]
##     year month   day flights
##    <int> <int> <int>   <int>
##  1  2013     1     1     842
##  2  2013     1     2     943
##  3  2013     1     3     914
##  4  2013     1     4     915
##  5  2013     1     5     720
##  6  2013     1     6     832
##  7  2013     1     7     933
##  8  2013     1     8     899
##  9  2013     1     9     902
## 10  2013     1    10     932
## # ... with 355 more rows
```

group_by() can also be used with mutate() and filter().

3. Exercises

§5.2.4: 1, 3

§5.5.2: 4

§5.6.7: 4. 5.

§5.7.1: 2. 3. 6. 8