Environmental Health Big Data Analysis – R4ds (1) Exploring Data using R

Il-Youp Kwak

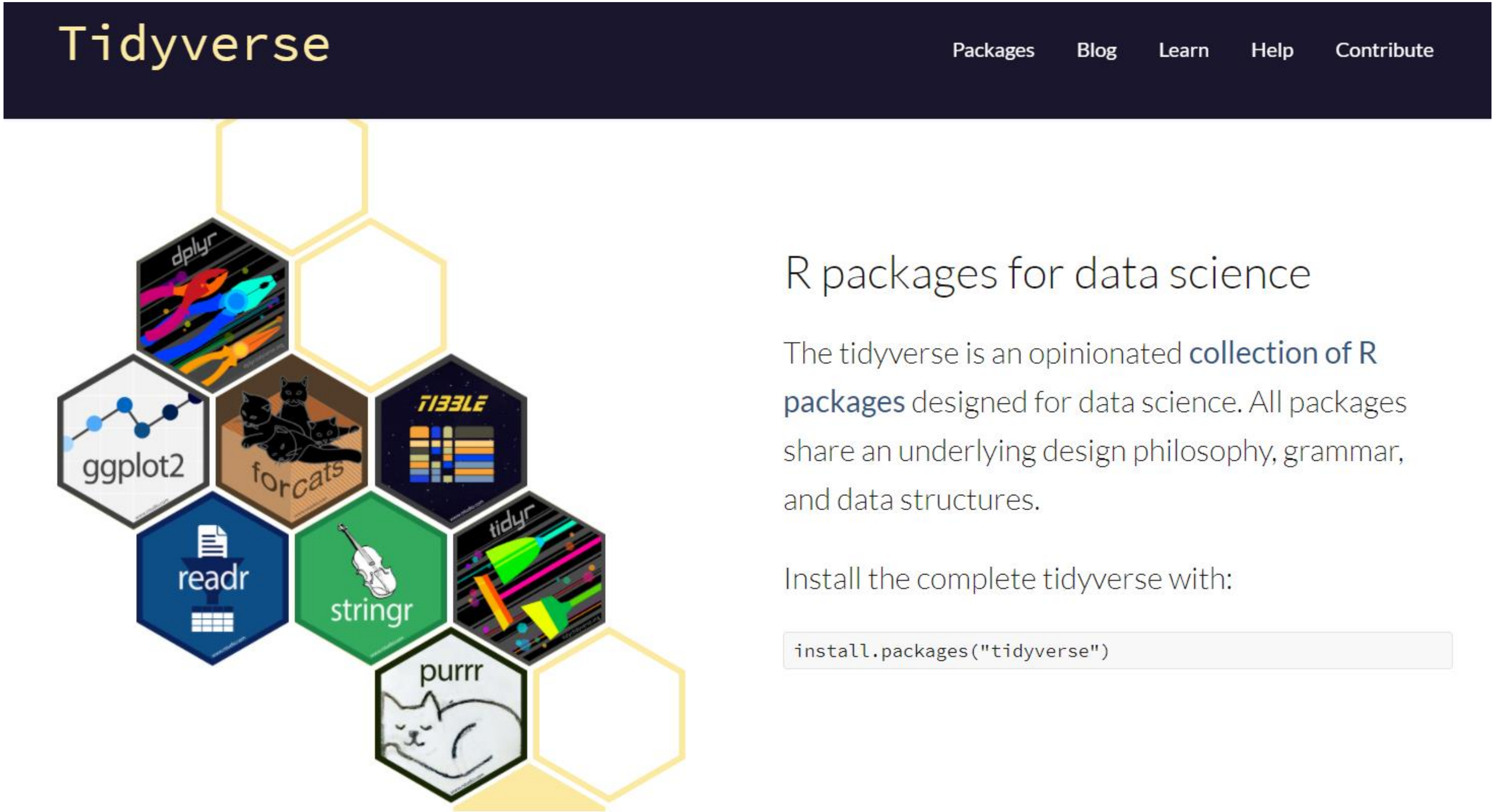
Chung-Ang University

What we will learn

1. **Data transformation**
2. **Data visualization**
3. **Data wrangling**
4. **Functional programming with R**

R for Data Science: https://r4ds.had.co.nz/

What is tidyverse?



Tidyverse

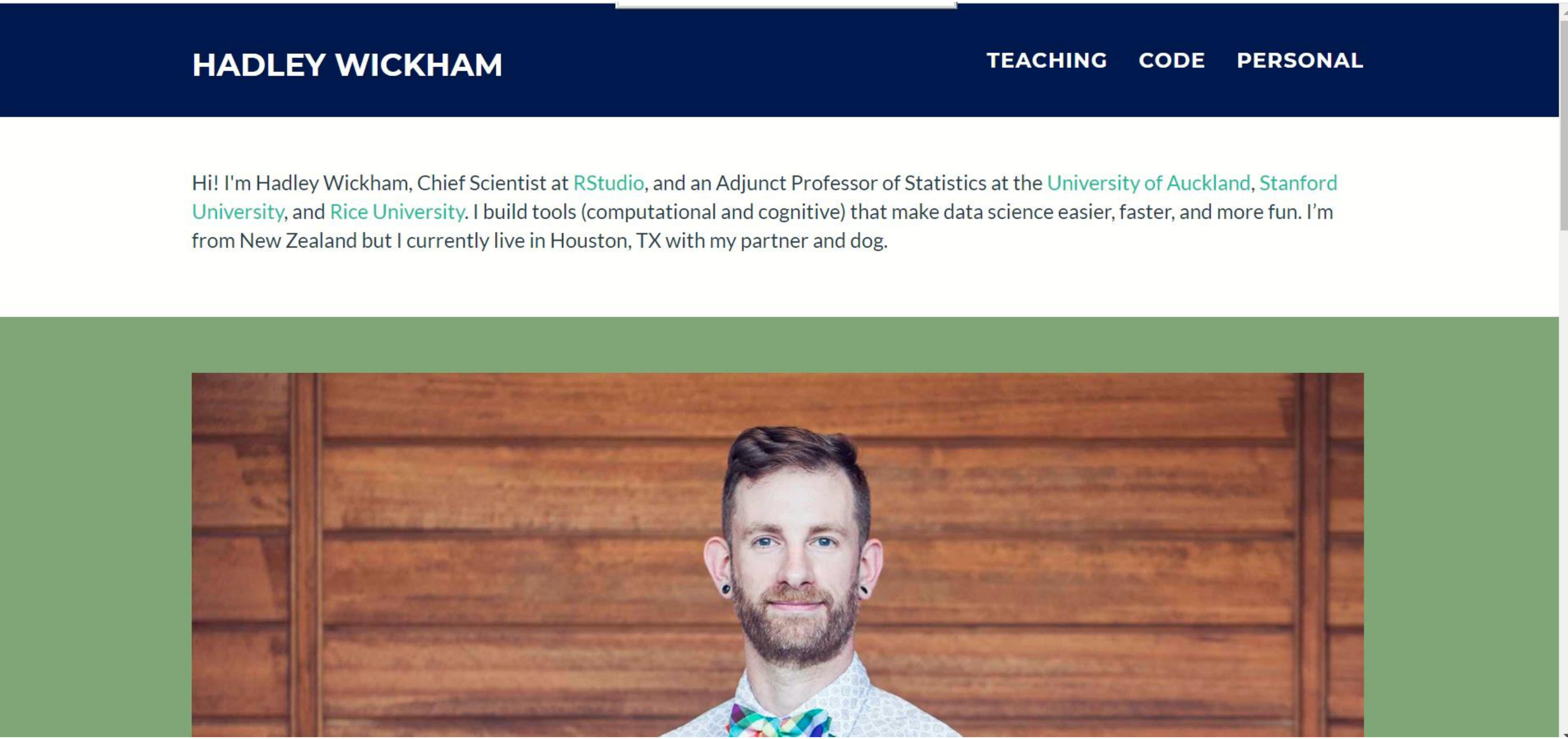
Packages Blog Learn Help Contribute

R packages for data science

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

```
install.packages("tidyverse")
```



HADLEY WICKHAM

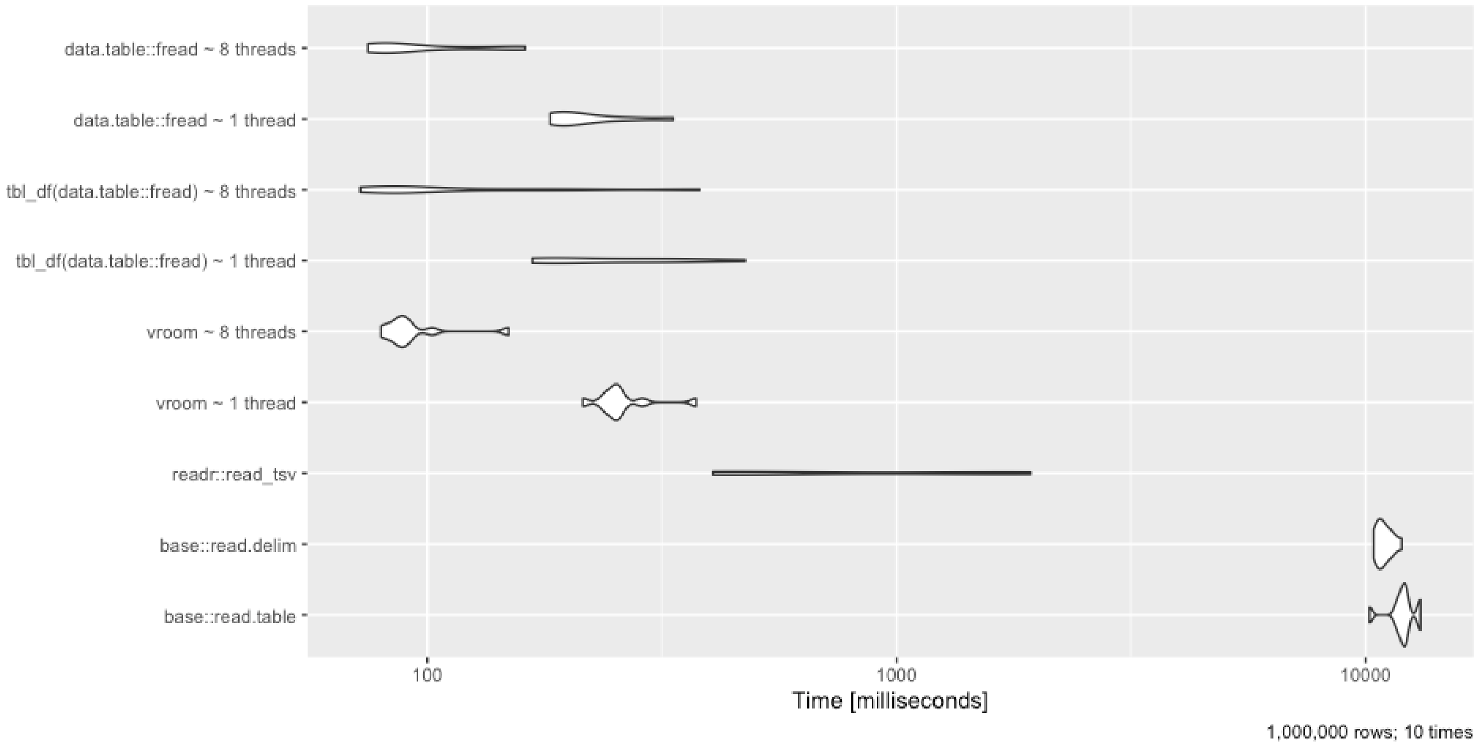
Hi! I'm Hadley Wickham, Chief Scientist at RStudio, and an Adjunct Professor of Statistics at the University of Auckland, Stanford University, and Rice University. I build tools (computational and cognitive) that make data science easier, faster, and more fun. I'm from New Zealand but I currently live in Houston, TX with my partner and dog.

Why tidyverse?

Easy to learn

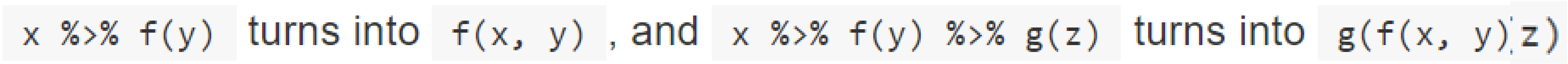
Faster than traditional R

Ex)



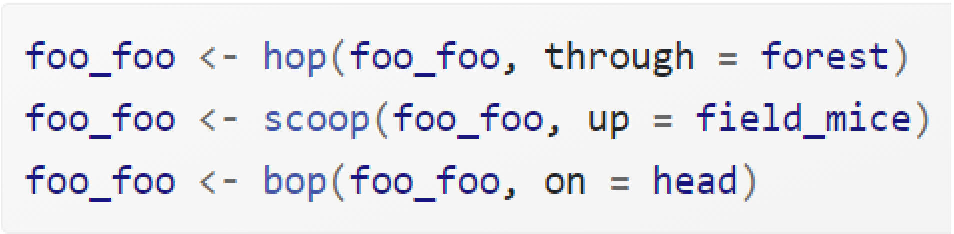
Pipes (%>%)

Pipes are a powerful tool for clearly expressing a sequence of multiple operations.

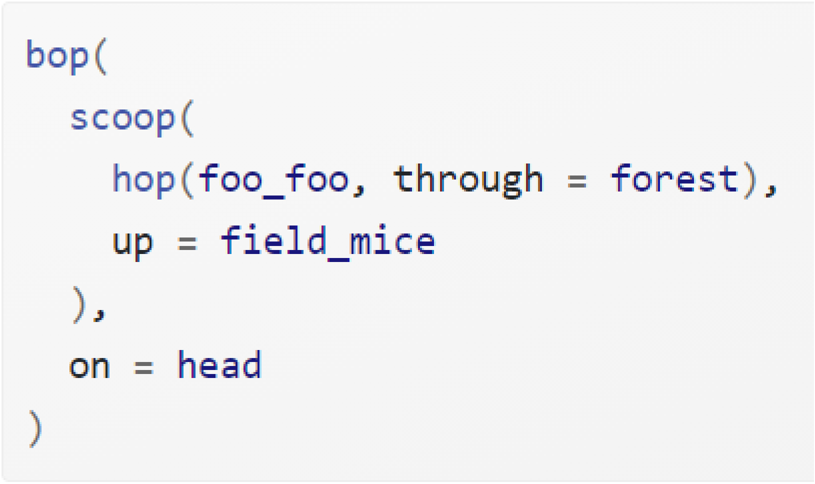


`x %>% f(y)` turns into `f(x, y)`, and `x %>% f(y) %>% g(z)` turns into `g(f(x, y), z)`

Without pipes

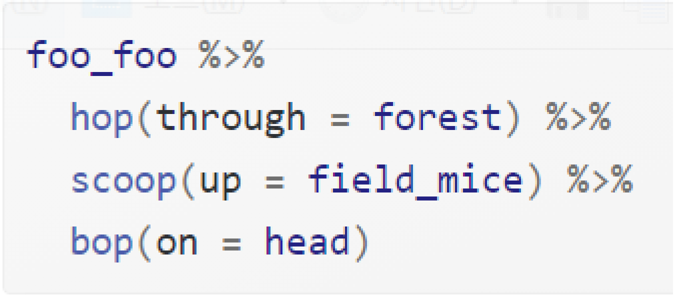


```
foo_foo <- hop(foo_foo, through = forest)
foo_foo <- scoop(foo_foo, up = field_mice)
foo_foo <- bop(foo_foo, on = head)
```



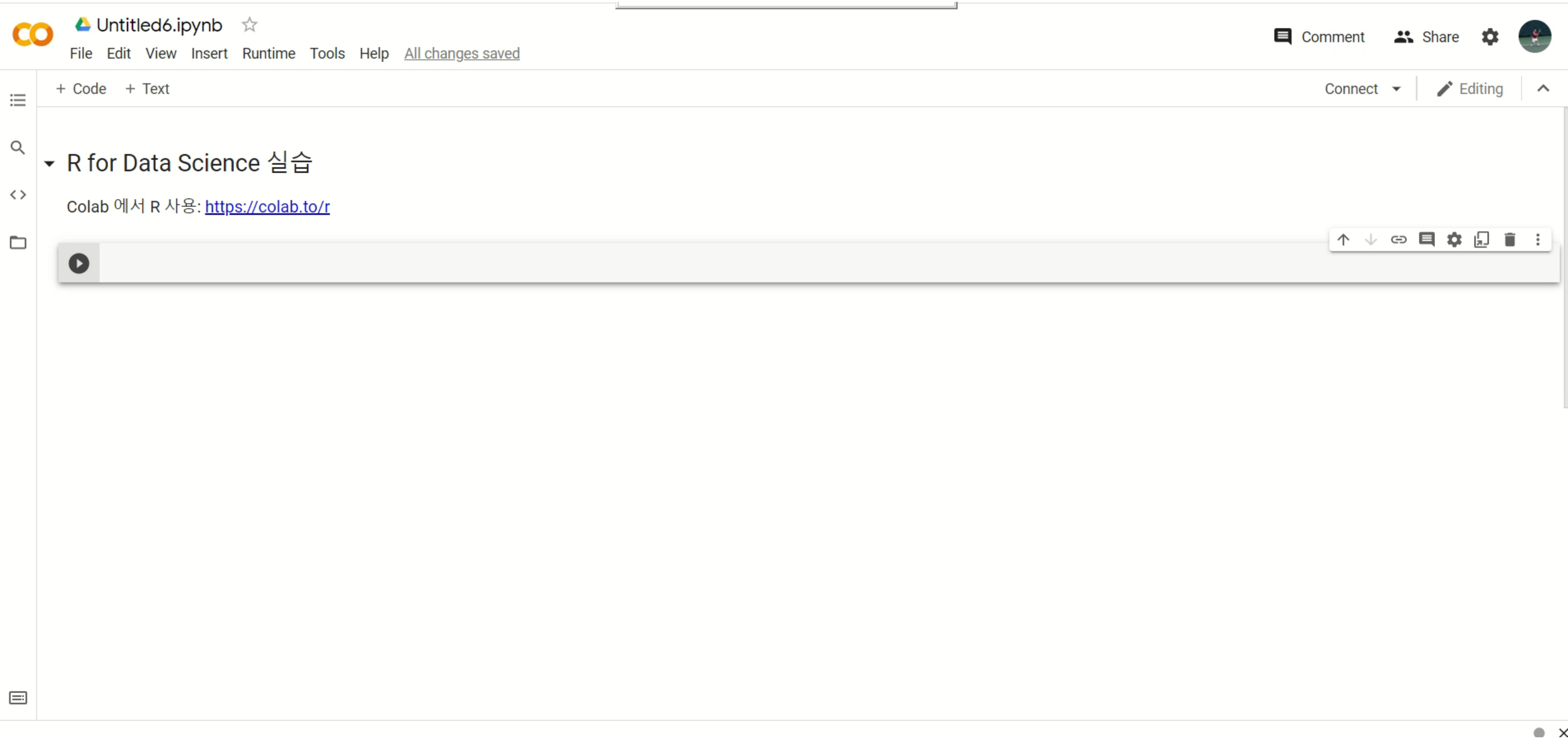
```
bop(
  scoop(
    hop(foo_foo, through = forest),
    up = field_mice
  ),
  on = head
)
```

With pipes



```
foo_foo %>%
  hop(through = forest) %>%
  scoop(up = field_mice) %>%
  bop(on = head)
```

https://r4ds.had.co.nz/pipes.html#pipes



R for Data Science 실습

Colab 에서 R 사용: https://colab.to/r

Data Transformation - dplyr

- **is a package for data manipulation**

- **Functions are coded in C++**

- **Fast and efficient**

Alternatives: data.table package, R base functions

https://r4ds.had.co.nz/transform.html

dplyr basics (five key functions)

filter(): Pick observations by their values.

arrange(): Reorder the rows.

select(): Pick variables by their names.

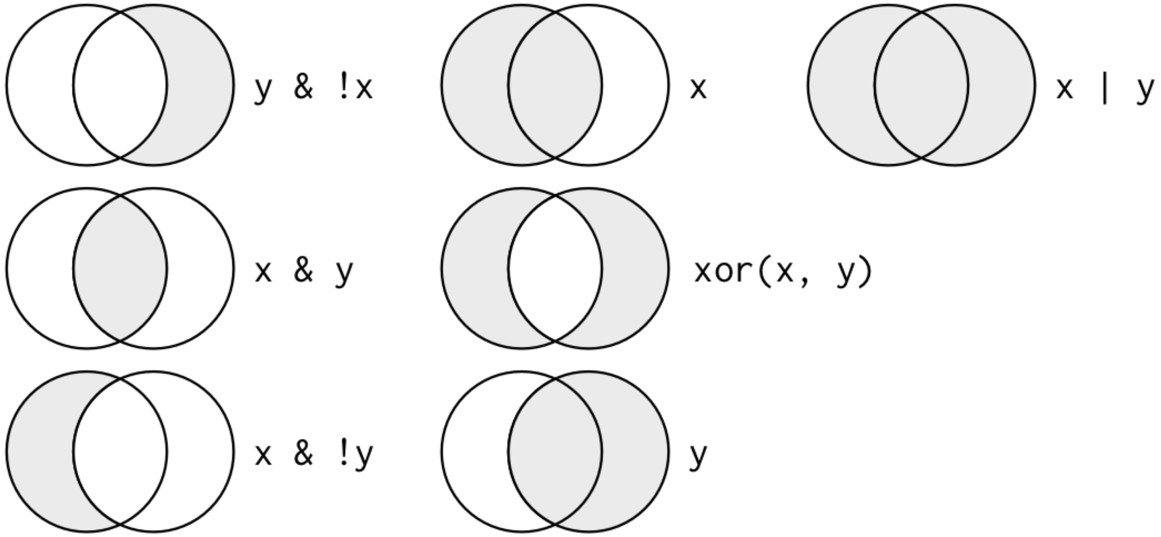
mutate(): Create new variables with existing variables

summarize(): Collapse many values down to a single summary

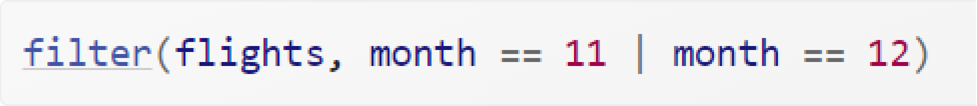
Filter rows with filter()

```
(dec25 <- filter(flights, month == 12, day == 25))
#> # A tibble: 719 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    12    25      456            500        -4      649            651
#> 2  2013    12    25      524            515         9      805            814
#> 3  2013    12    25      542            540         2      832            850
#> 4  2013    12    25      546            550        -4     1022           1027
#> 5  2013    12    25      556            600        -4      730            745
#> 6  2013    12    25      557            600        -3      743            752
#> # … with 713 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>
```

Logical Operations in filter()

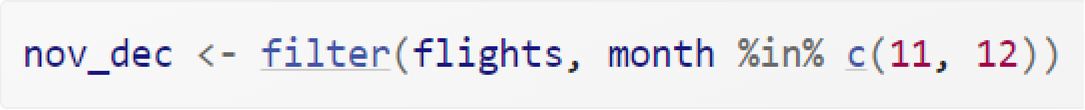


Ex)

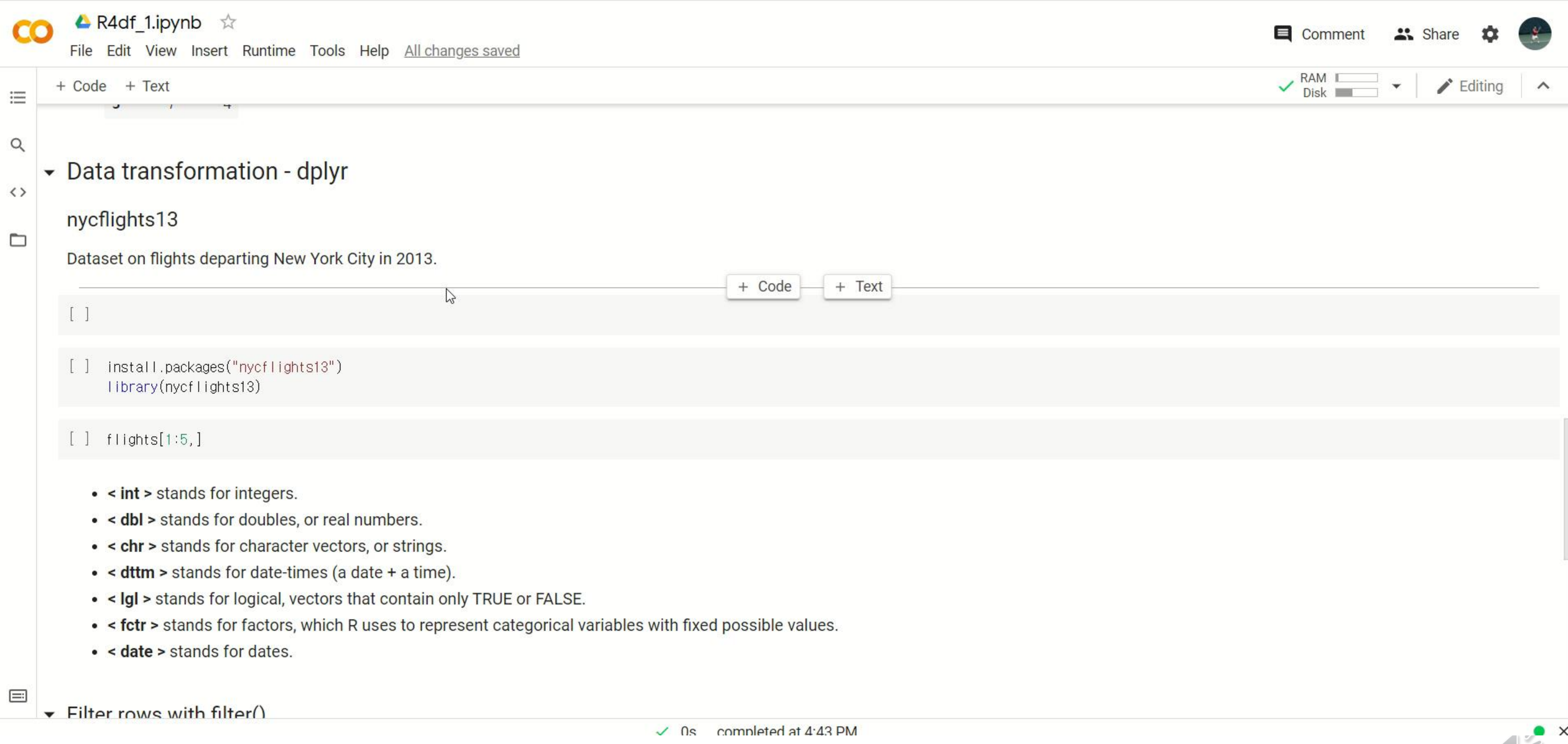


```
filter(flights, month == 11 | month == 12)
```

With x %in% y, it select every row where x is one of the values in y



```
nov_dec <- filter(flights, month %in% c(11, 12))
```



Data transformation - dplyr

nycflights13

Dataset on flights departing New York City in 2013.

```
[ ]
```

```
[ ] install.packages("nycflights13")
    library(nycflights13)
```

```
[ ] flights[1:5,]
```

- **< int >** stands for integers.
- **< dbl >** stands for doubles, or real numbers.
- **< chr >** stands for character vectors, or strings.
- **< dttm >** stands for date-times (a date + a time).
- **< lgl >** stands for logical, vectors that contain only TRUE or FALSE.
- **< fctr >** stands for factors, which R uses to represent categorical variables with fixed possible values.
- **< date >** stands for dates.

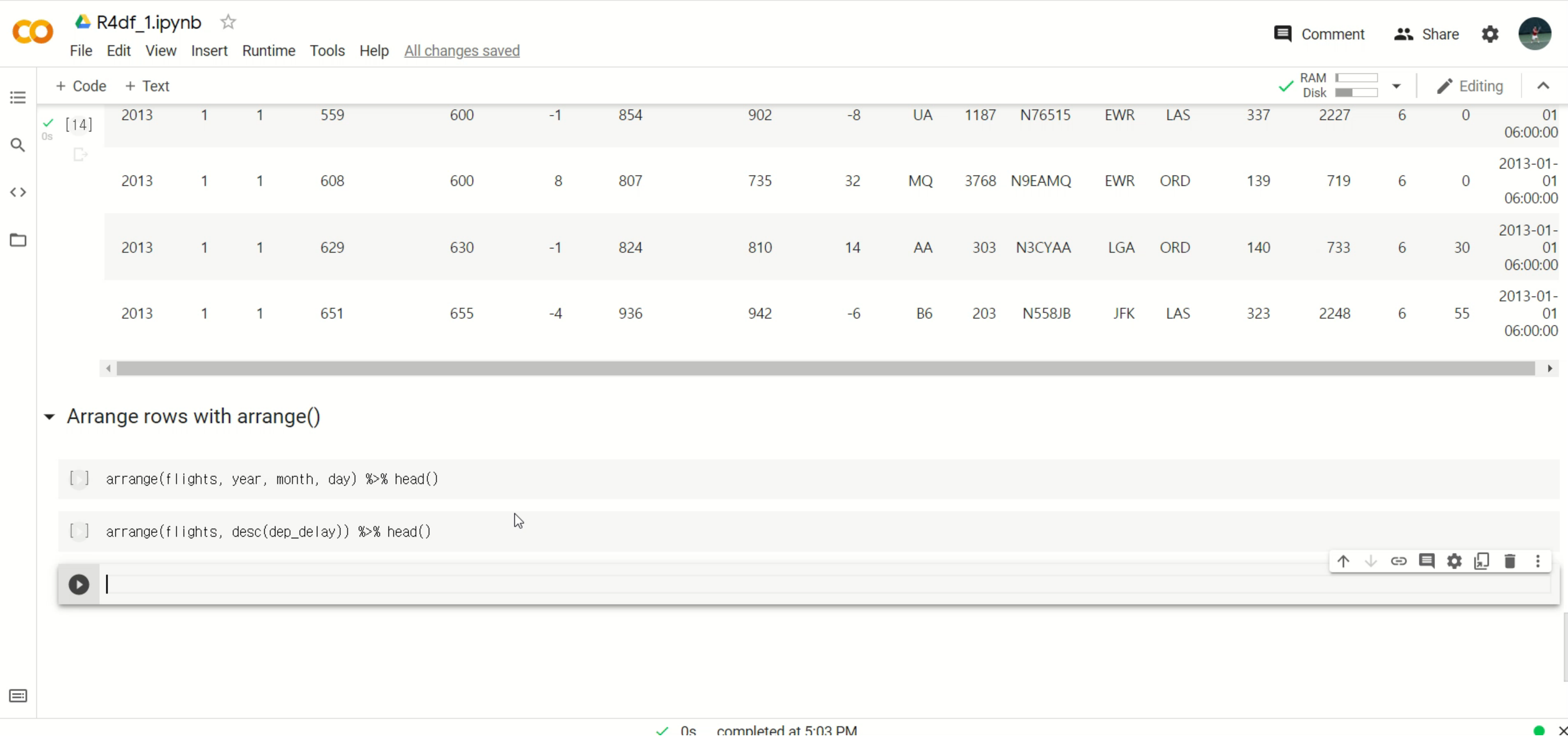
Filter rows with filter()

Arrange rows with arrange()

```
arrange(flights, year, month, day)
#> # A tibble: 336,776 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      517            515         2      830            819
#> 2  2013     1     1      533            529         4      850            830
#> 3  2013     1     1      542            540         2      923            850
#> 4  2013     1     1      544            545        -1     1004           1022
…
```

Use desc() for descending order

```
arrange(flights, desc(dep_delay))
#> # A tibble: 336,776 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     9      641            900      1301     1242           1530
#> 2  2013     6    15     1432           1935      1137     1607           2120
#> 3  2013     1    10     1121           1635      1126     1239           1810
…
```



```
2013  1  1  559  600  -1  854  902  -8  UA  1187  N76515  EWR  LAS  337  2227  6   0             01
                                                                                                06:00:00
2013  1  1  608  600   8  807  735  32  MQ  3768  N9EAMQ  EWR  ORD  139   719  6   0  2013-01-
                                                                                              01
                                                                                        06:00:00
2013  1  1  629  630  -1  824  810  14  AA   303  N3CYAA  LGA  ORD  140   733  6  30  2013-01-
                                                                                              01
                                                                                        06:00:00
2013  1  1  651  655  -4  936  942  -6  B6   203  N558JB  JFK  LAS  323  2248  6  55  2013-01-
                                                                                              01
                                                                                        06:00:00
```

Arrange rows with arrange()

```
arrange(flights, year, month, day) %>% head()
```

```
arrange(flights, desc(dep_delay)) %>% head()
```

Select columns with select()

Select columns by name

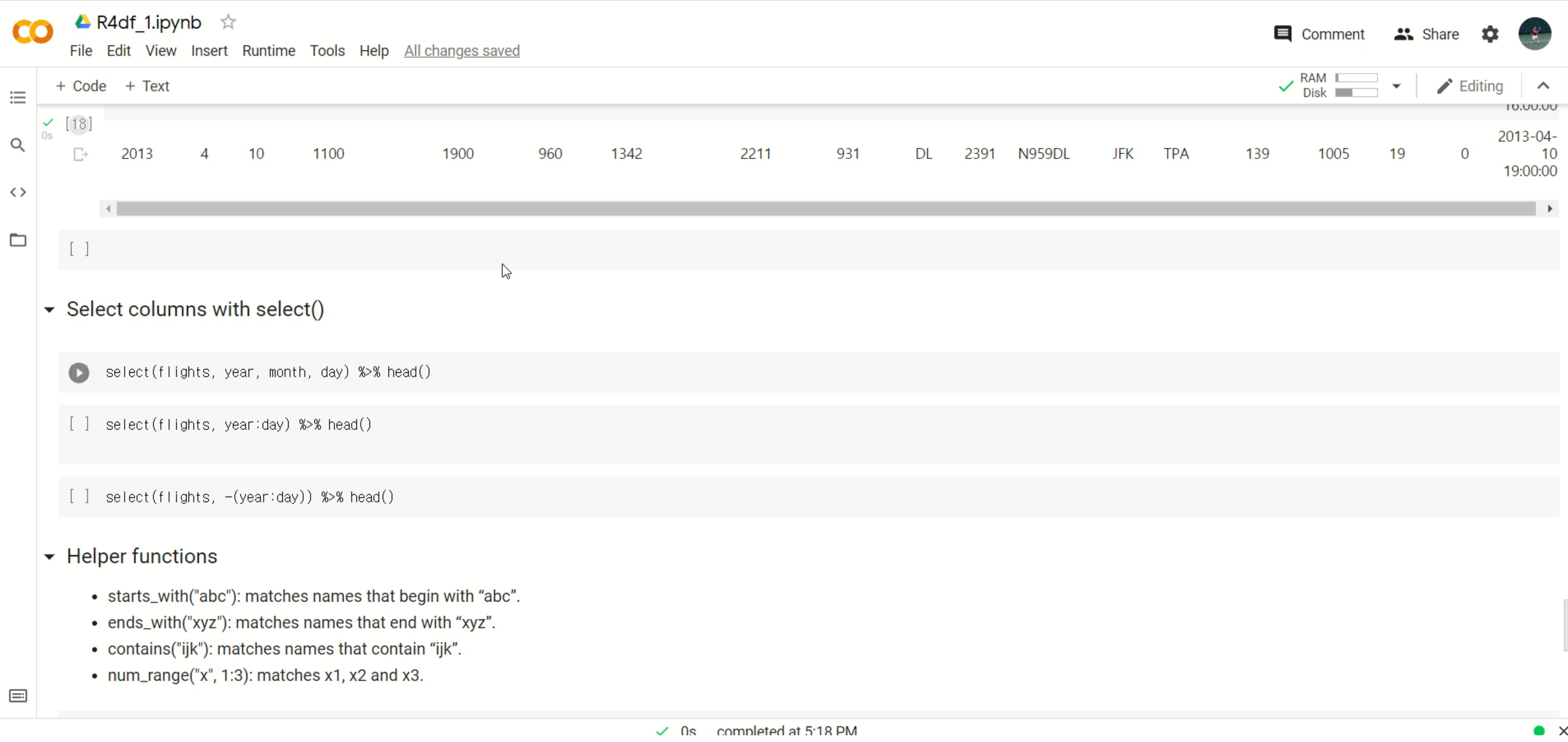
```
select(flights, year, month, day)
#> # A tibble: 336,776 x 3
#>    year month   day
#>   <int> <int> <int>
#> 1  2013     1     1
#> 2  2013     1     1
#> 3  2013     1     1
…
```

Select all columns between year and day (inclusive)

```
select(flights, year: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
…
```

Select all columns except those from year to day (inclusive)

```
select(flights, -(year:day))
#> # A tibble: 336,776 x 16
#>   dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
#>      <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>
#> 1      517            515         2      830            819        11 UA
#> 2      533            529         4      850            830        20 UA
#> 3      542            540         2      923            850        33 AA
#> 4      544            545        -1     1004           1022       -18 B6
#> 5      554            600        -6      812            837       -25 DL
#> 6      554            558        -4      740            728        12 UA
#> # … with 336,770 more rows, and 9 more variables: flight <int>, tailnum <chr>,
#> #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#> #   minute <dbl>, time_hour <dttm>
```



```
                                                                                                                16:00:00
 2013   4   10   1100   1900   960   1342   2211   931   DL   2391   N959DL   JFK   TPA   139   1005   19   0   2013-04-
                                                                                                                      10
                                                                                                                19:00:00
```

Select columns with select()

```
select(flights, year, month, day) %>% head()
```

```
select(flights, year:day) %>% head()
```

```
select(flights, -(year:day)) %>% head()
```

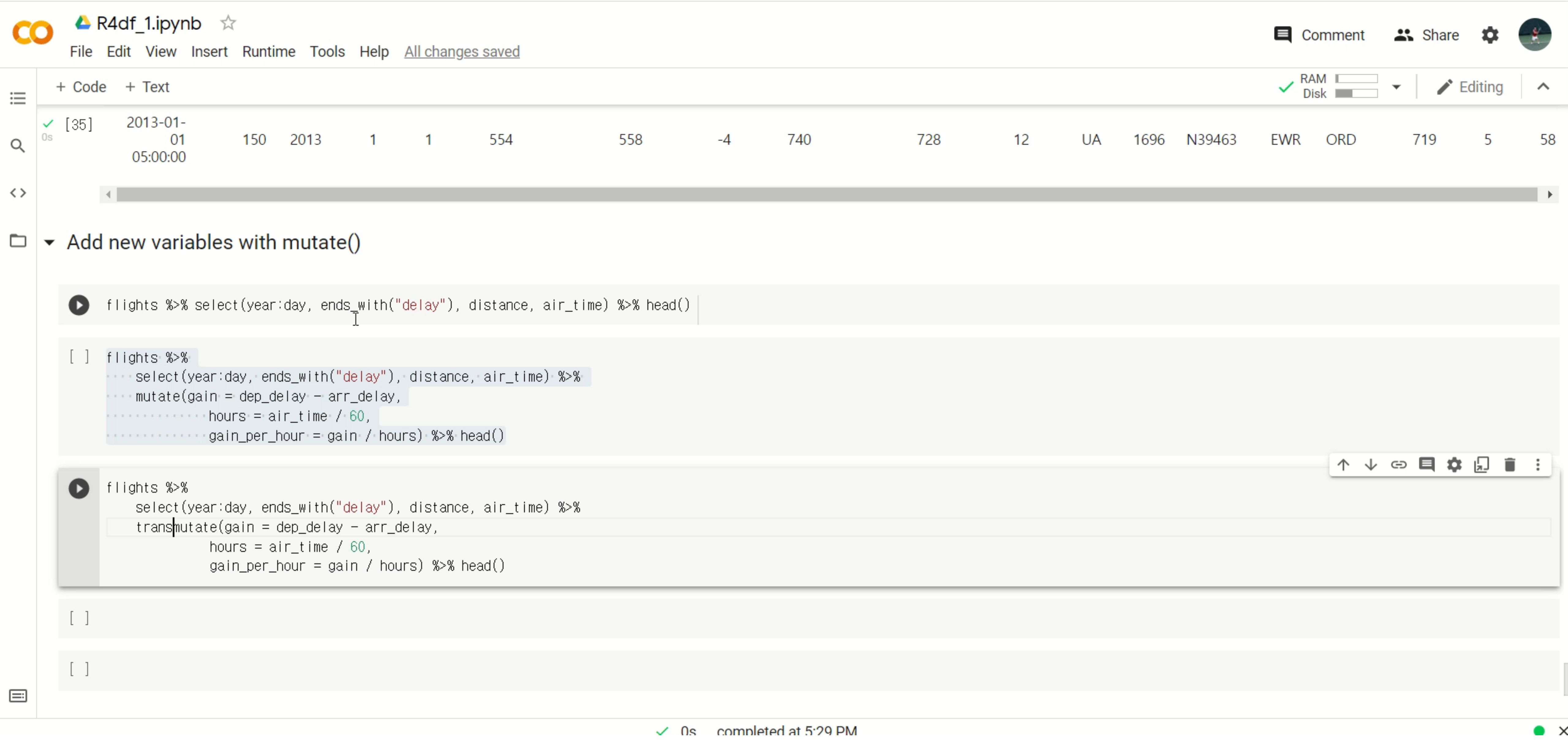
Helper functions

- starts_with("abc"): matches names that begin with “abc”.
- ends_with("xyz"): matches names that end with “xyz”.
- contains("ijk"): matches names that contain “ijk”.
- num_range("x", 1:3): matches x1, x2 and x3.

Add new variables with mutate()

```
flights %>%
   select(year:day, ends_with("delay"), distance, air_time) %>%
   mutate(gain = dep_delay - arr_delay,
           hours = air_time / 60,
           gain_per_hour = gain / hours)
#> # A tibble: 336,776 x 10
#>    year month   day dep_delay arr_delay distance air_time  gain hours
#>   <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl> <dbl>
#> 1  2013     1     1         2        11     1400      227    -9  3.78
#> 2  2013     1     1         4        20     1416      227   -16  3.78
#> 3  2013     1     1         2        33     1089      160   -31  2.67
#> 4  2013     1     1        -1       -18     1576      183    17  3.05
#> 5  2013     1     1        -6       -25      762      116    19  1.93
#> 6  2013     1     1        -4        12      719      150   -16  2.5
#> # … with 336,770 more rows, and 1 more variable: gain_per_hour <dbl>
```

Note that you can refer to columns that you've just created:



```
[35]  2013-01-
        01     150  2013  1  1  554  558  -4  740  728  12  UA  1696  N39463  EWR  ORD  719  5  58
      05:00:00
```

Add new variables with mutate()

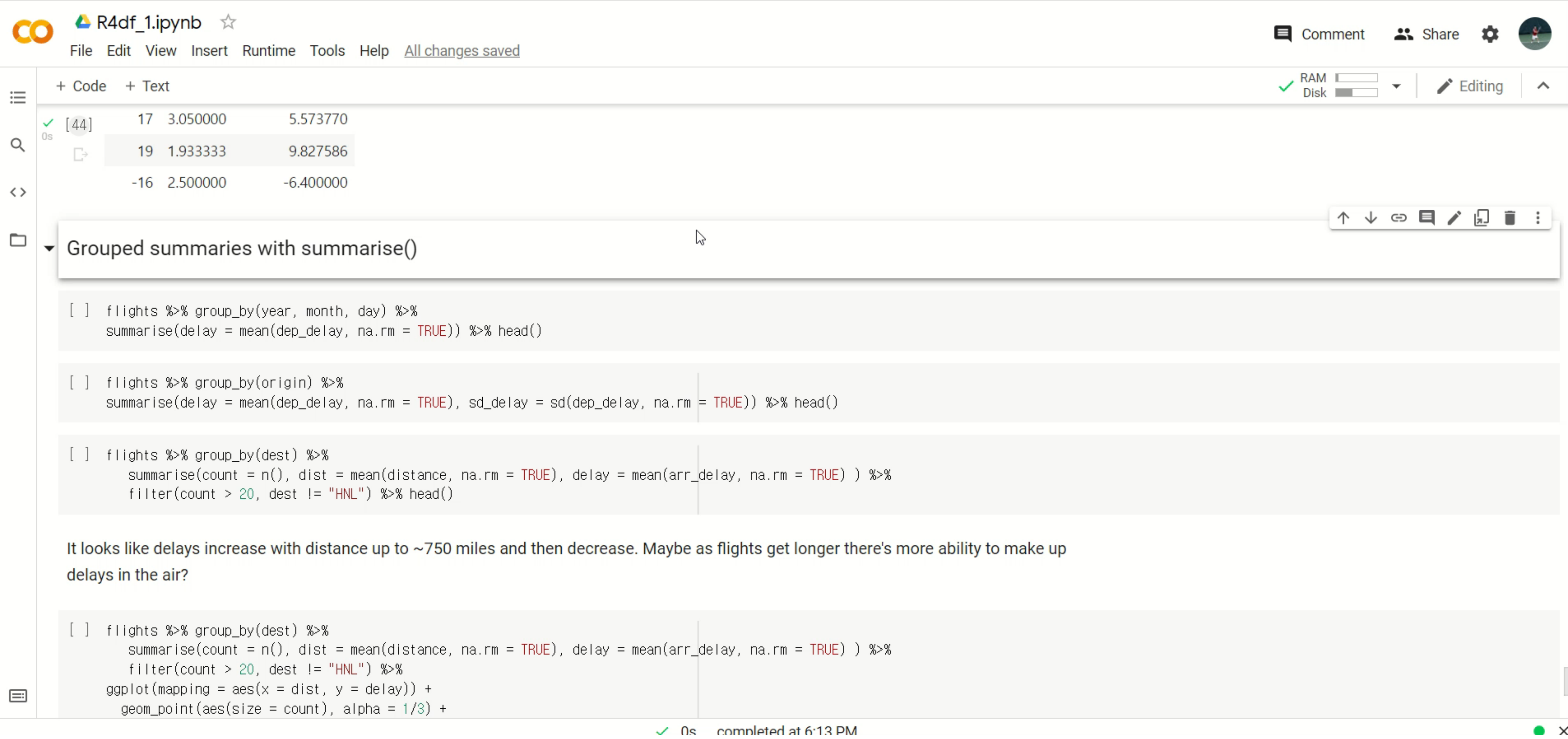
```
flights %>% select(year:day, ends_with("delay"), distance, air_time) %>% head()
```

```
flights %>%
    select(year:day, ends_with("delay"), distance, air_time) %>%
    mutate(gain = dep_delay - arr_delay,
               hours = air_time / 60,
               gain_per_hour = gain / hours) %>% head()
```

```
flights %>%
    select(year:day, ends_with("delay"), distance, air_time) %>%
    transmutate(gain = dep_delay - arr_delay,
               hours = air_time / 60,
               gain_per_hour = gain / hours) %>% head()
```

Grouped summaries with summarise()

```
flights %>% group_by(year, month, day) %>%
summarise(delay = mean(dep_delay, na.rm = TRUE))
#> `summarise()` regrouping output by 'year', 'month' (override with `.groups`
argument)
#> # A tibble: 365 x 4
#> # Groups:   year, month [12]
#>    year month   day delay
#>   <int> <int> <int> <dbl>
#> 1  2013     1     1 11.5
#> 2  2013     1     2 13.9
#> 3  2013     1     3 11.0
#> 4  2013     1     4  8.95
#> 5  2013     1     5  5.73
#> 6  2013     1     6  7.15
#> # … with 359 more rows
```



```
 17   3.050000      5.573770
 19   1.933333      9.827586
-16   2.500000     -6.400000
```

Grouped summaries with summarise()

```
[ ] flights %>% group_by(year, month, day) %>%
    summarise(delay = mean(dep_delay, na.rm = TRUE)) %>% head()
```

```
[ ] flights %>% group_by(origin) %>%
    summarise(delay = mean(dep_delay, na.rm = TRUE), sd_delay = sd(dep_delay, na.rm = TRUE)) %>% head()
```

```
[ ] 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") %>% head()
```

It looks like delays increase with distance up to ~750 miles and then decrease. Maybe as flights get longer there's more ability to make up delays in the air?

```
[ ] 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") %>%
    ggplot(mapping = aes(x = dist, y = delay)) +
      geom_point(aes(size = count), alpha = 1/3) +
```

✓ 0s completed at 6:13 PM

Data Visualization - ggplot

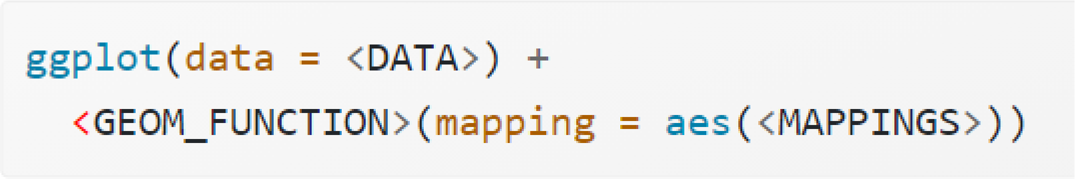
- is a package for data visualization

- Use grammar of graphics, a coherent system for describing and building graphs.

Alternatives: R base functions

https://r4ds.had.co.nz/data-visualisation.html

Aesthetic mapping



```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

**ggplot(data = , mapping = aes(x = , y= , color= , size= ,
alpha= , shape=))**

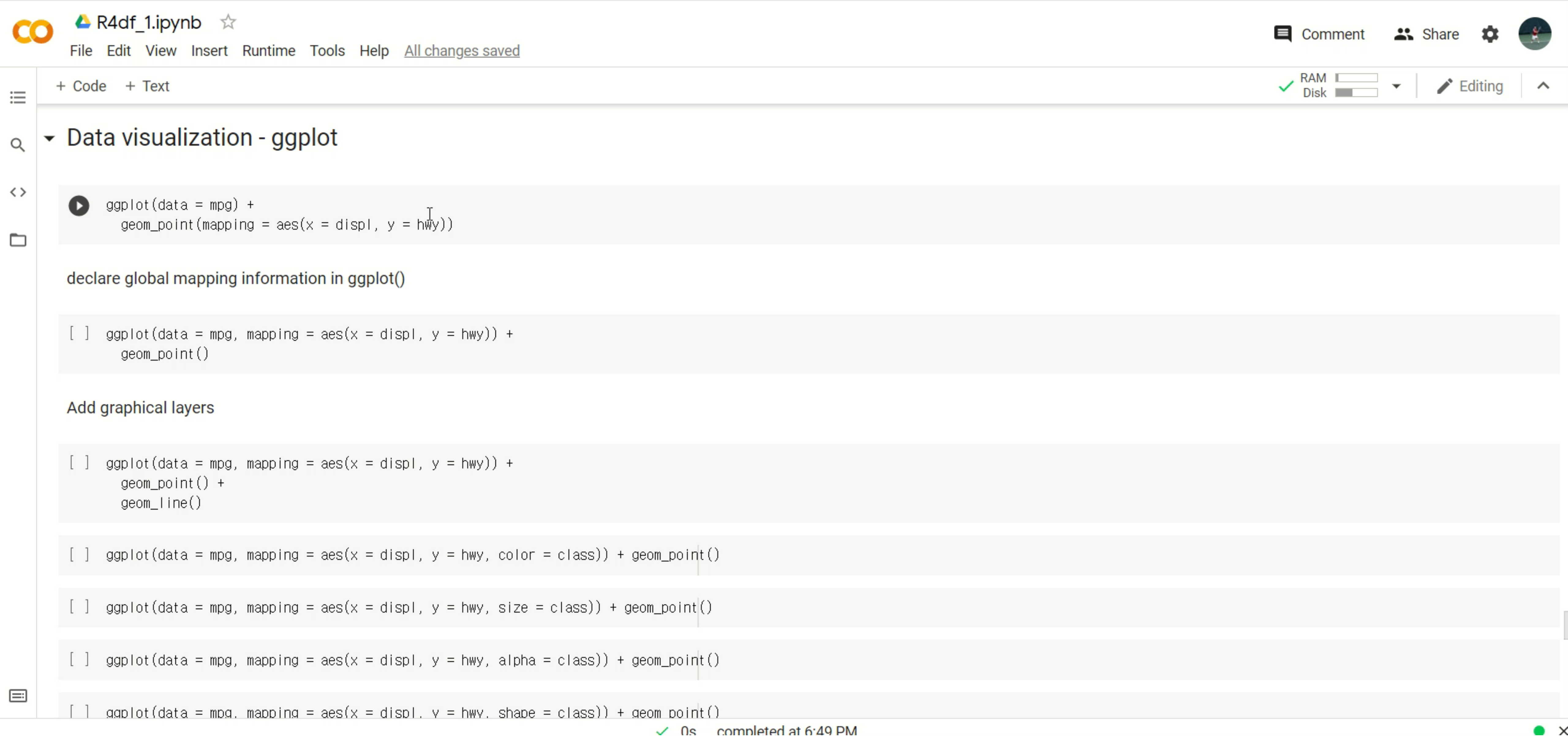
Facet

Split your plot into facets (facet_wrap, facet_grid)

Geometric objects

geom stand for geometrical objects

geom_point, geom_smooth, geom_bar, geom_violin, geom_abline, etc



Data visualization - ggplot

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

declare global mapping information in ggplot()

```
[ ] ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
      geom_point()
```

Add graphical layers

```
[ ] ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
      geom_point() +
      geom_line()
```

```
[ ] ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) + geom_point()
```

```
[ ] ggplot(data = mpg, mapping = aes(x = displ, y = hwy, size = class)) + geom_point()
```

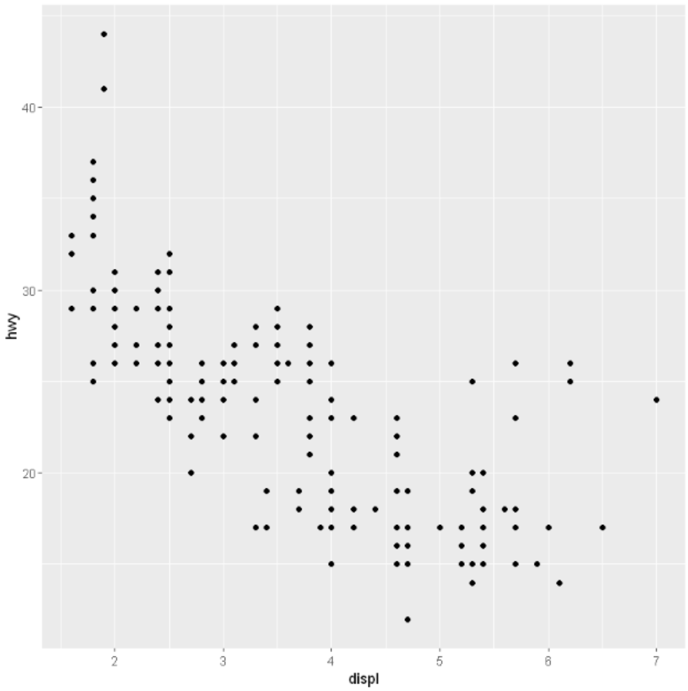
```
[ ] ggplot(data = mpg, mapping = aes(x = displ, y = hwy, alpha = class)) + geom_point()
```

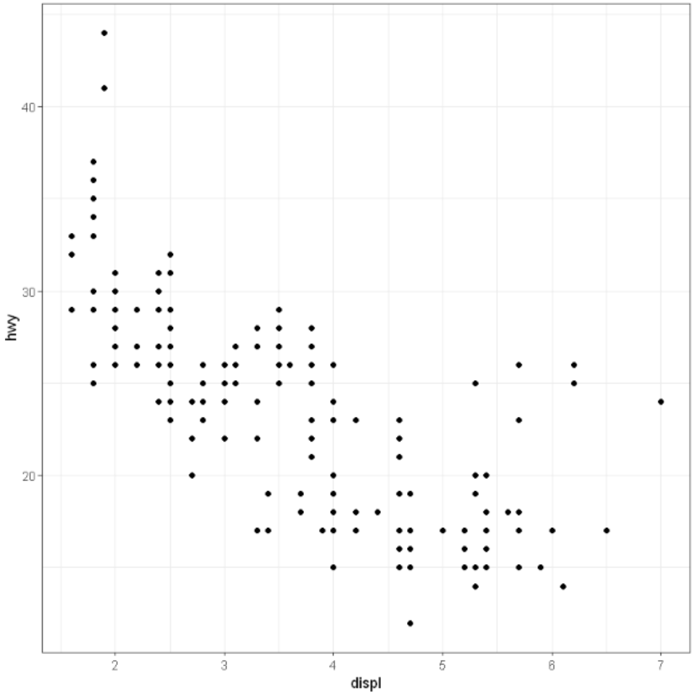
```
[ ] ggplot(data = mpg, mapping = aes(x = displ, y = hwy, shape = class)) + geom_point()
```

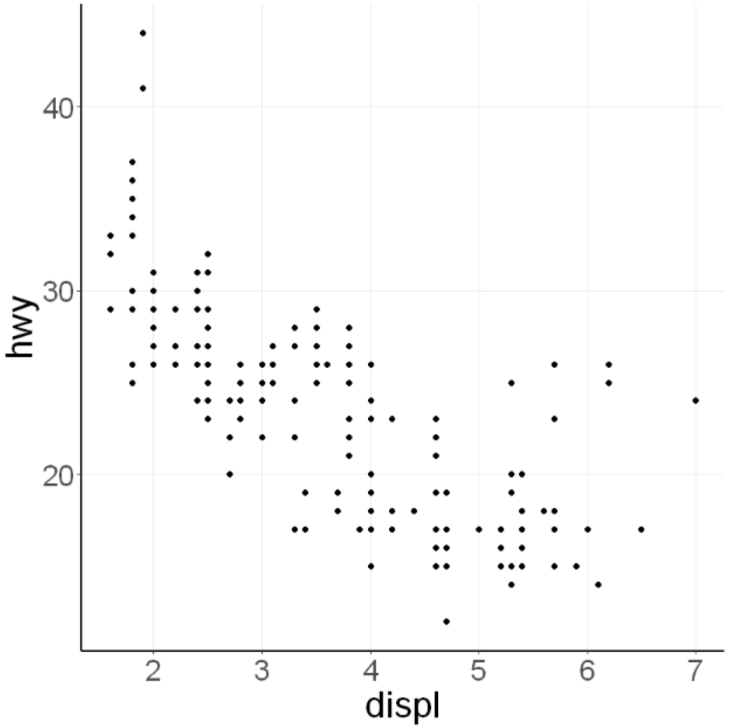
✓ 0s completed at 6:49 PM

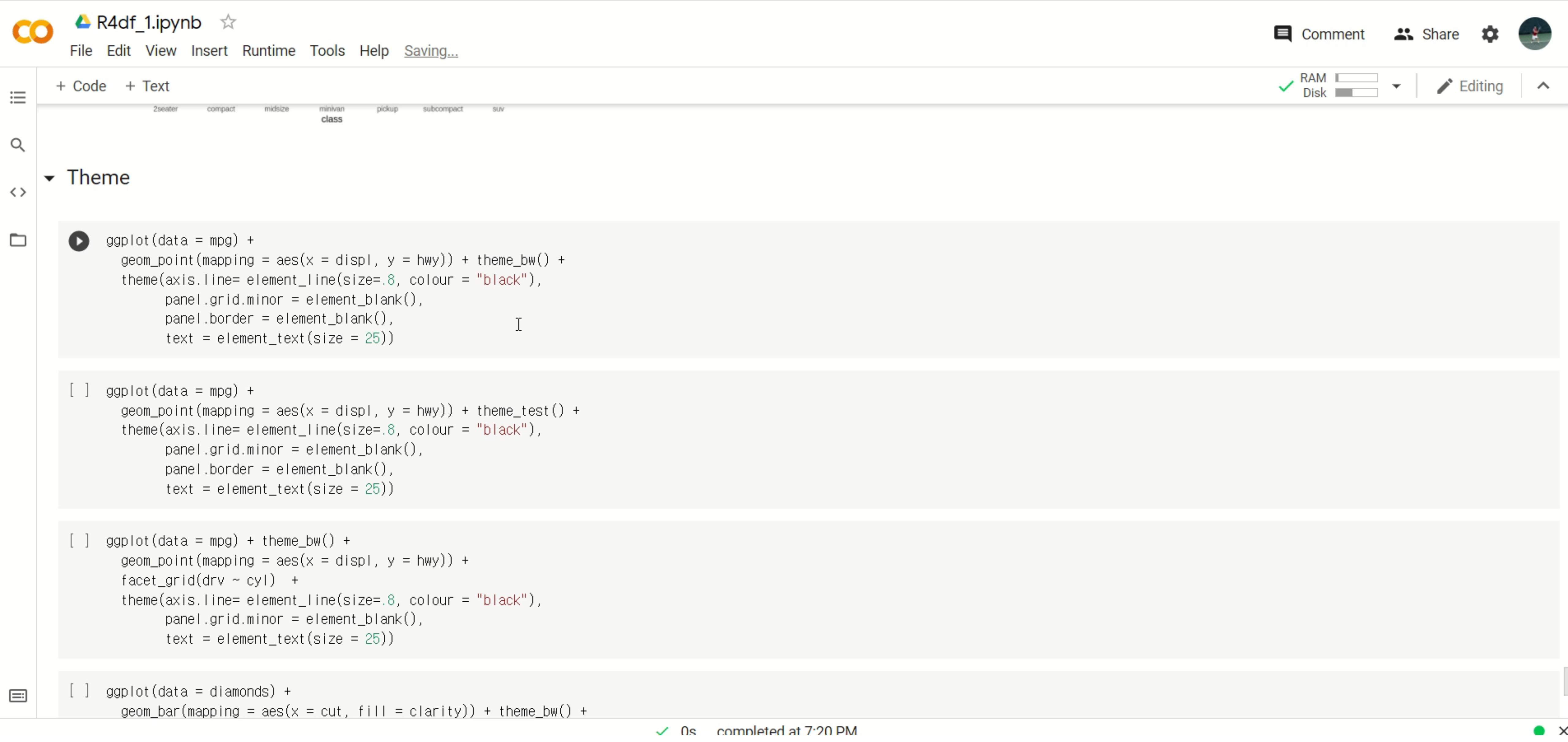
Theme

- What will you choose?









Theme

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) + theme_bw() +
  theme(axis.line= element_line(size=.8, colour = "black"),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        text = element_text(size = 25))
```

```
[ ] ggplot(data = mpg) +
      geom_point(mapping = aes(x = displ, y = hwy)) + theme_test() +
      theme(axis.line= element_line(size=.8, colour = "black"),
            panel.grid.minor = element_blank(),
            panel.border = element_blank(),
            text = element_text(size = 25))
```

```
[ ] ggplot(data = mpg) + theme_bw() +
      geom_point(mapping = aes(x = displ, y = hwy)) +
      facet_grid(drv ~ cyl)  +
      theme(axis.line= element_line(size=.8, colour = "black"),
            panel.grid.minor = element_blank(),
            text = element_text(size = 25))
```

```
[ ] ggplot(data = diamonds) +
      geom_bar(mapping = aes(x = cut, fill = clarity)) + theme_bw() +
```

Thank you! ☺