# DATA 100,Week 3 (A)

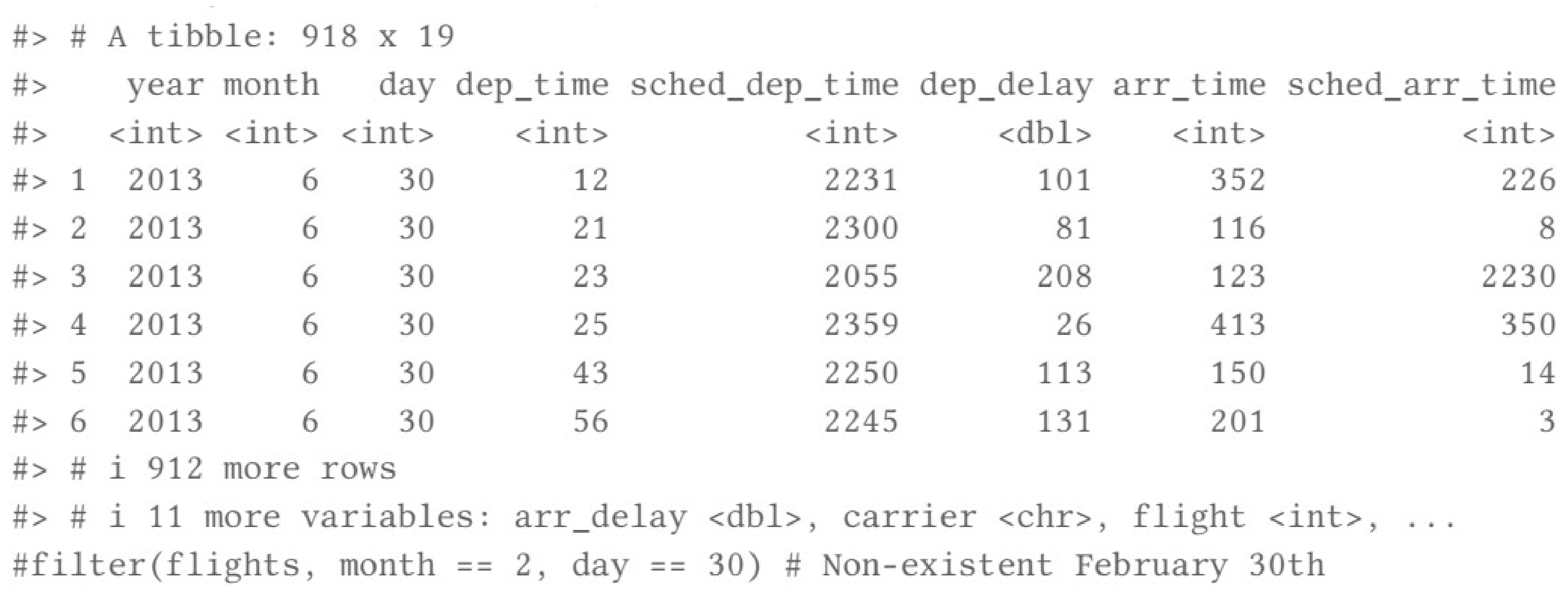
***Recall*** library(tidyverse) should appear in the beginning of every .qmd (or .rmd)file we have from now on

**3.3.4.** filter()

Get flights going out on a given day. **Note** that the column names month and day are not in quotation marks

```
filter(flights, month == 6, day == 30) # June 30th
```

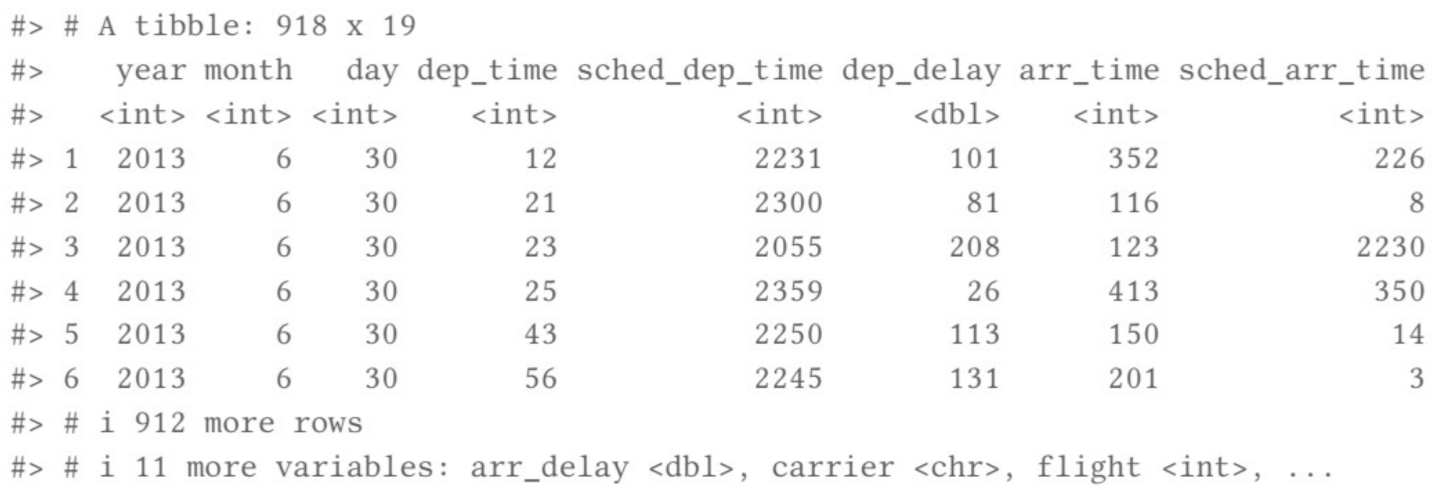
# DATA 100,Week 3 (A)



```
#> # A tibble: 918 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     6    30       12           2231       101      352            226
#> 2  2013     6    30       21           2300        81      116              8
#> 3  2013     6    30       23           2055       208      123           2230
#> 4  2013     6    30       25           2359        26      413            350
#> 5  2013     6    30       43           2250       113      150             14
#> 6  2013     6    30       56           2245       131      201              3
#> # i 912 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
#filter(flights, month == 2, day == 30) # Non-existent February 30th
```

## DATA 100,Week 3 (A)

```
# use pipe
flights |>
filter(month == 6, day == 30)
```



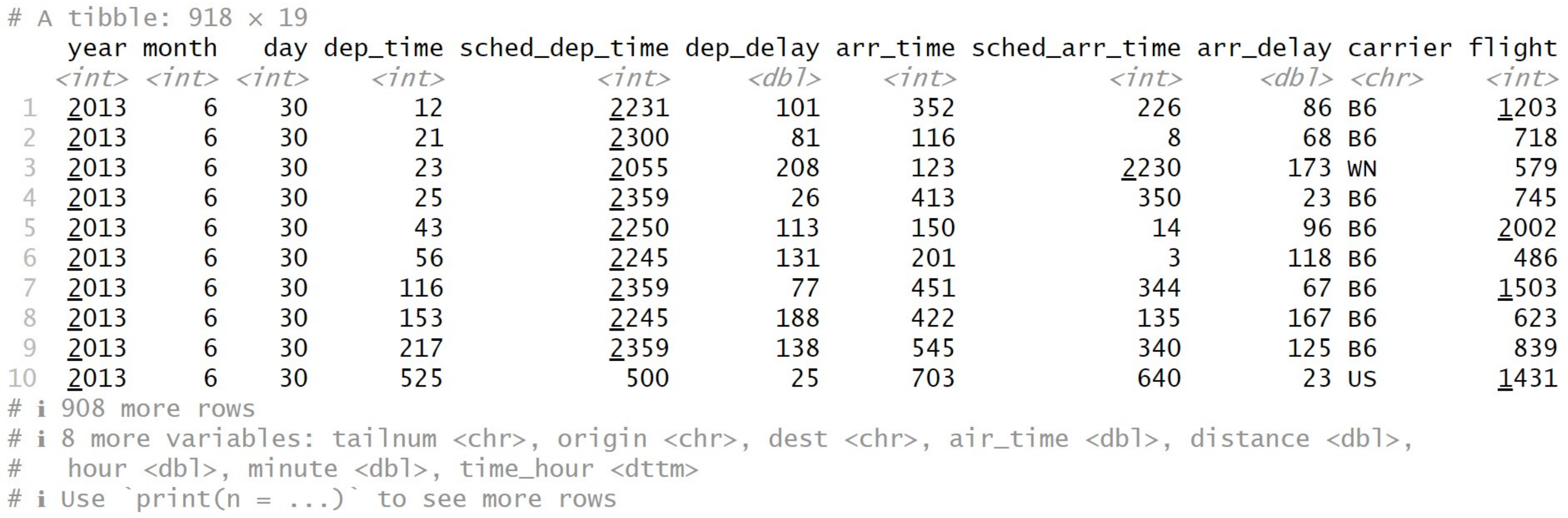
```
#> # A tibble: 918 x 19
#>    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
#>   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
#> 1  2013     6    30       12           2231       101      352            226
#> 2  2013     6    30       21           2300        81      116              8
#> 3  2013     6    30       23           2055       208      123           2230
#> 4  2013     6    30       25           2359        26      413            350
#> 5  2013     6    30       43           2250       113      150             14
#> 6  2013     6    30       56           2245       131      201              3
#> # i 912 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
```

Since the input dataframe is not modified, if the filtered result is needed later, should assign it to a new variable to keep the filtered result.

# DATA 100,Week 3 (A)

**Comment**: The name of the variable should reflect the meaning of its contents. If possible, should avoid *update in-place* as much as possible.

```
Jun30 <- flights |> filter(month == 6, day == 30)
Jun30
```



```
# A tibble: 918 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int>
 1  2013     6    30       12           2231       101      352            226        86 B6        1203
 2  2013     6    30       21           2300        81      116              8        68 B6         718
 3  2013     6    30       23           2055       208      123           2230       173 WN         579
 4  2013     6    30       25           2359        26      413            350        23 B6         745
 5  2013     6    30       43           2250       113      150             14        96 B6        2002
 6  2013     6    30       56           2245       131      201              3       118 B6         486
 7  2013     6    30      116           2359        77      451            344        67 B6        1503
 8  2013     6    30      153           2245       188      422            135       167 B6         623
 9  2013     6    30      217           2359       138      545            340       125 B6         839
10  2013     6    30      525            500        25      703            640        23 US        1431
# ℹ 908 more rows
# ℹ 8 more variables: tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
# ℹ Use `print(n = ...)` to see more rows
```

## DATA 100,Week 3 (A)

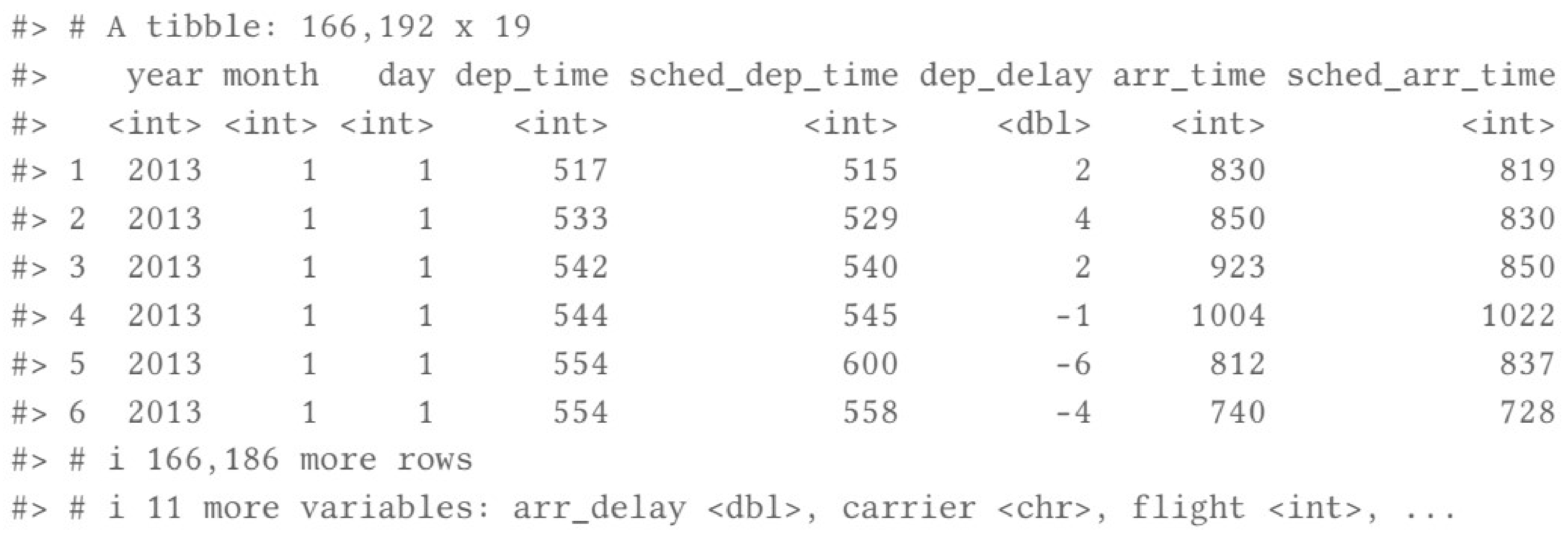
Surrounding the assign statement by a pair of round parenthesis () accomplishes assignment and print-out in one go.

```
(September <- flights |> filter(month == 9))
```

```
#> # A tibble: 27,574 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     9     1        9           2359        10      343            340
#> 2  2013     9     1      117           2245       152      218           2359
#> 3  2013     9     1      508            516        -8      717            800
#> 4  2013     9     1      537            545        -8      849            855
#> 5  2013     9     1      537            545        -8      906            921
#> 6  2013     9     1      549            600       -11      815            850
#> # i 27,568 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
```

# DATA 100,Week 3 (A)

```r
# get flights during the first half of the months
flights |> filter(day %in% 1:15)
```



```
#> # A tibble: 166,192 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
#> 5  2013     1     1      554            600        -6      812            837
#> 6  2013     1     1      554            558        -4      740            728
#> # i 166,186 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
```

## DATA 100,Week 3 (A)

**Rules of logical operations:**

• DeMorgan's Laws

**– !(x | y) = (!x) & (!y)**

**– !(x & y) = (!x) | (!y)**

• Distributivity

**– x & (y | z) = (x & y) | (x & z)**

**– x | (y & z) = (x | y) & (x | z)**

## DATA 100,Week 3 (A)

The following commands give the same list of flights

• In filter's parameter list: and can be represented by , and & in the filtering conditions.

```
flights |> filter(!(arr_delay > 120 | dep_delay > 120))

flights |> filter(arr_delay <= 120 & dep_delay <= 120)

flights |> filter(arr_delay <= 120, dep_delay <= 120)
```

## DATA 100,Week 3 (A)

**Filter out (for) missing values**

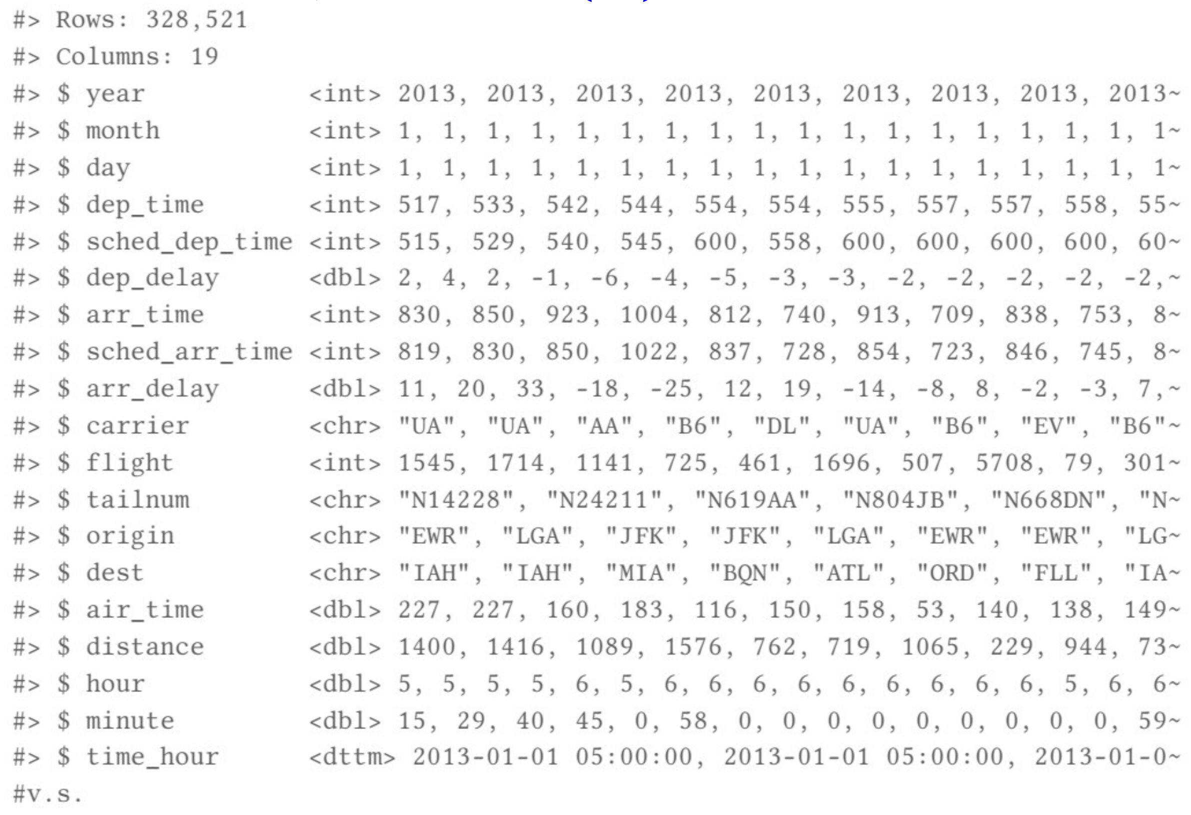
- == NA or != NA do not work
- … since as we saw NA == NA is still NA, (as is NA != NA)
- Use is.na()

## DATA 100,Week 3 (A)

The flights that departed:

```
glimpse(flights |> filter(!is.na(dep_time)))
```

# DATA 100,Week 3 (A)



```
#> Rows: 328,521
#> Columns: 19
#> $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013~
#> $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 55~
#> $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 60~
#> $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2,~
#> $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 8~
#> $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 8~
#> $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7,~
#> $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6"~
#> $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301~
#> $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N~
#> $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LG~
#> $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IA~
#> $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149~
#> $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 73~
#> $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6~
#> $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59~
#> $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-0~
#v.s.
```

```
glimpse(flights |> filter(dep_time != NA))
```

# DATA 100,Week 3 (A)

```
Rows: 0
Columns: 19
$ year           <int>
$ month          <int>
$ day            <int>
$ dep_time       <int>
$ sched_dep_time <int>
$ dep_delay      <dbl>
$ arr_time       <int>
$ sched_arr_time <int>
$ 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>
```

The flights that never arrived:

## DATA 100,Week 3 (A)

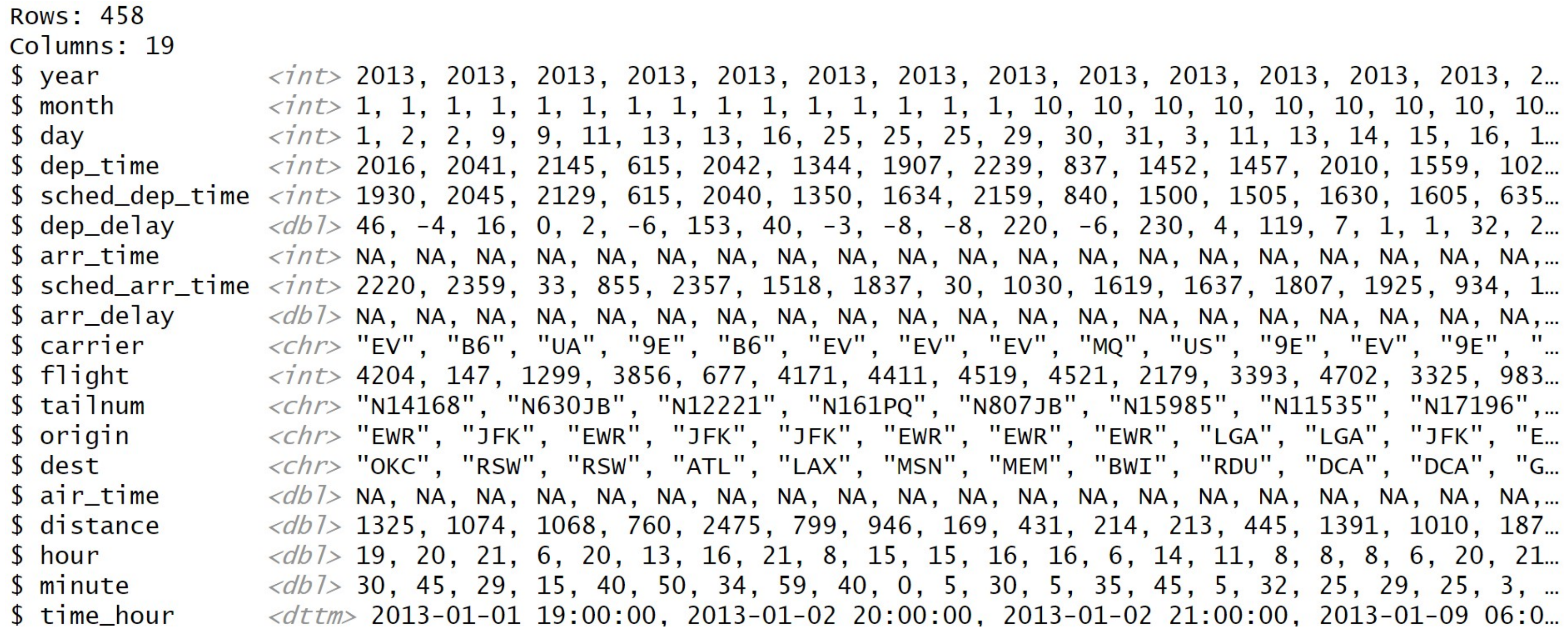
glimpse(flights |> filter(is.na(arr_time)))

```
#> Rows: 8,713
#> Columns: 19
#> $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013~
#> $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ day            <int> 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3~
#> $ dep_time       <int> 2016, NA, NA, NA, NA, 2041, 2145, NA, NA, NA, NA, NA~
#> $ sched_dep_time <int> 1930, 1630, 1935, 1500, 600, 2045, 2129, 1540, 1620,~
#> $ dep_delay      <dbl> 46, NA, NA, NA, NA, -4, 16, NA, NA, NA, NA, NA, NA, ~
#> $ arr_time       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ sched_arr_time <int> 2220, 1815, 2240, 1825, 901, 2359, 33, 1747, 1746, 1~
#> $ arr_delay      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ carrier        <chr> "EV", "EV", "AA", "AA", "B6", "B6", "UA", "EV", "EV"~
#> $ flight         <int> 4204, 4308, 791, 1925, 125, 147, 1299, 4352, 4406, 4~
#> $ tailnum        <chr> "N14168", "N18120", "N3EHAA", "N3EVAA", "N618JB", "N~
#> $ origin         <chr> "EWR", "EWR", "LGA", "LGA", "JFK", "JFK", "EWR", "EW~
#> $ dest           <chr> "OKC", "RDU", "DFW", "MIA", "FLL", "RSW", "RSW", "CV~
#> $ air_time       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ distance       <dbl> 1325, 416, 1389, 1096, 1069, 1074, 1068, 569, 319, 2~
#> $ hour           <dbl> 19, 16, 19, 15, 6, 20, 21, 15, 16, 13, 14, 13, 15, 1~
#> $ minute         <dbl> 30, 30, 35, 0, 0, 45, 29, 40, 20, 55, 20, 21, 45, 30~
#> $ time_hour      <dttm> 2013-01-01 19:00:00, 2013-01-01 16:00:00, 2013-01-0~
```

The flight that departed but never arrived:

## DATA 100,Week 3 (A)

glimpse(flights |> filter(!is.na(dep_time) & is.na(arr_time)))

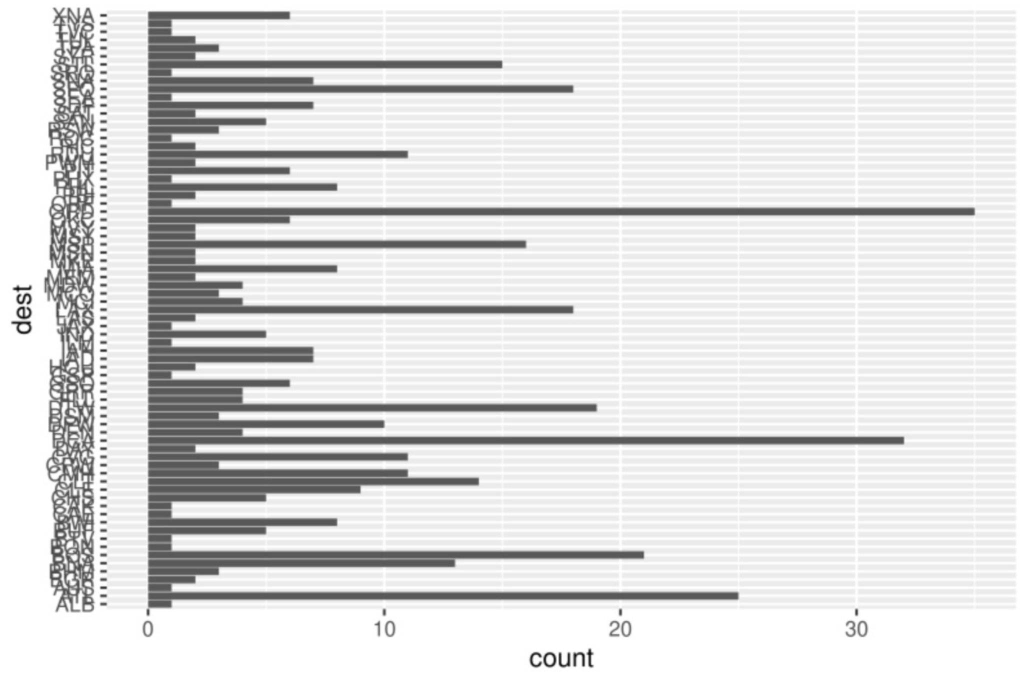


```
Rows: 458
Columns: 19
$ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2…
$ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 10, 10, 10, 10, 10, 10, 10, 10, 10…
$ day            <int> 1, 2, 2, 9, 9, 11, 13, 13, 16, 25, 25, 25, 29, 30, 31, 3, 11, 13, 14, 15, 16, 1…
$ dep_time       <int> 2016, 2041, 2145, 615, 2042, 1344, 1907, 2239, 837, 1452, 1457, 2010, 1559, 102…
$ sched_dep_time <int> 1930, 2045, 2129, 615, 2040, 1350, 1634, 2159, 840, 1500, 1505, 1630, 1605, 635…
$ dep_delay      <dbl> 46, -4, 16, 0, 2, -6, 153, 40, -3, -8, -8, 220, -6, 230, 4, 119, 7, 1, 1, 32, 2…
$ arr_time       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
$ sched_arr_time <int> 2220, 2359, 33, 855, 2357, 1518, 1837, 30, 1030, 1619, 1637, 1807, 1925, 934, 1…
$ arr_delay      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
$ carrier        <chr> "EV", "B6", "UA", "9E", "B6", "EV", "EV", "EV", "MQ", "US", "9E", "EV", "9E", "…
$ flight         <int> 4204, 147, 1299, 3856, 677, 4171, 4411, 4519, 4521, 2179, 3393, 4702, 3325, 983…
$ tailnum        <chr> "N14168", "N630JB", "N12221", "N161PQ", "N807JB", "N15985", "N11535", "N17196",…
$ origin         <chr> "EWR", "JFK", "EWR", "JFK", "JFK", "EWR", "EWR", "EWR", "LGA", "LGA", "JFK", "E…
$ dest           <chr> "OKC", "RSW", "RSW", "ATL", "LAX", "MSN", "MEM", "BWI", "RDU", "DCA", "DCA", "G…
$ air_time       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
$ distance       <dbl> 1325, 1074, 1068, 760, 2475, 799, 946, 169, 431, 214, 213, 445, 1391, 1010, 187…
$ hour           <dbl> 19, 20, 21, 6, 20, 13, 16, 21, 8, 15, 15, 16, 16, 6, 14, 11, 8, 8, 8, 6, 20, 21…
$ minute         <dbl> 30, 45, 29, 15, 40, 50, 34, 59, 40, 0, 5, 30, 5, 35, 45, 5, 32, 25, 29, 25, 3, …
$ time_hour      <dttm> 2013-01-01 19:00:00, 2013-01-02 20:00:00, 2013-01-02 21:00:00, 2013-01-09 06:0…
```

dep_no_arr <- flights |>

# DATA 100,Week 3 (A)

```
filter(!is.na(dep_time), is.na(arr_time)) ggplot(data = dep_no_arr, mapping =
aes(x = dest)) +
geom_bar() + coord_flip()
```



## DATA 100,Week 3 (A)

filter **basic syntax:**

```
filter(DATAFRAME, CONDITIONS separated by comma)
```

or using pipe

```
DATAFRAME |> filter(CONDITIONS separated by comma)
```

use ?filter to learn more

## DATA 100,Week 3 (A)

filter **mistakes:**

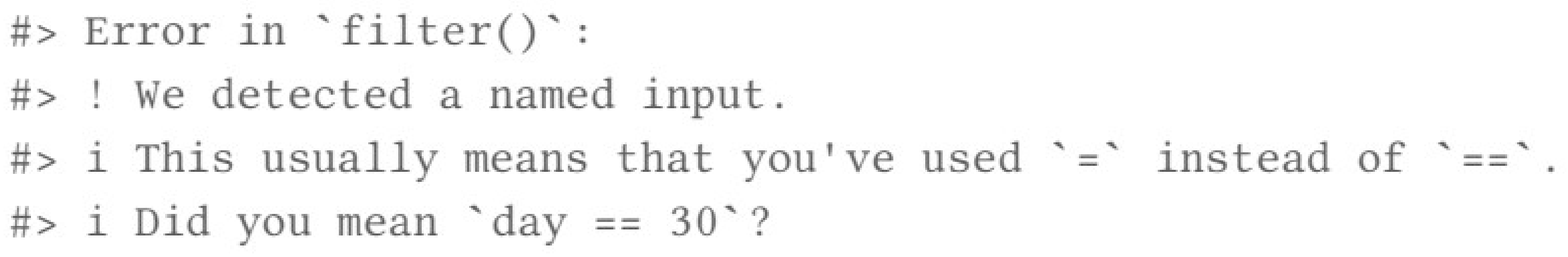
Mix up == and =: one is comparison (==) which gives TRUE or FALSE, the other is assignment (=) which puts a value into a variable

```
flights |>
filter(month == 6, day == 30)
```

```
#> # A tibble: 918 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     6    30       12           2231       101      352            226
#> 2   2013     6    30       21           2300        81      116              8
#> 3   2013     6    30       23           2055       208      123           2230
#> 4   2013     6    30       25           2359        26      413            350
#> 5   2013     6    30       43           2250       113      150             14
#> 6   2013     6    30       56           2245       131      201              3
#> # i 912 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
# v.s.
```

## DATA 100,Week 3 (A)

```
flights |>
filter(month == 6, day = 30)
```



```
#> Error in `filter()`:
#> ! We detected a named input.
#> i This usually means that you've used `=` instead of `==`.
#> i Did you mean `day == 30`?
```

## DATA 100,Week 3 (A)

### arrange()

Get data in order according to the values in some variables (columns)
• Usually in *ascending* order
• Use desc() to specify in descending order

Why ordering with respect to a collection of variables?

• Quickly see the range of data directly, combined with glimpse() for example

• Have some idea about the extreme cases

## DATA 100,Week 3 (A)

```
flights |>
arrange(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    12     7     2040           2123       -43       40           2352
#> 2  2013     2     3     2022           2055       -33     2240           2338
#> 3  2013    11    10     1408           1440       -32     1549           1559
#> 4  2013     1    11     1900           1930       -30     2233           2243
#> 5  2013     1    29     1703           1730       -27     1947           1957
#> 6  2013     8     9      729            755       -26     1002            955
#> # i 336,770 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
```

## DATA 100,Week 3 (A)

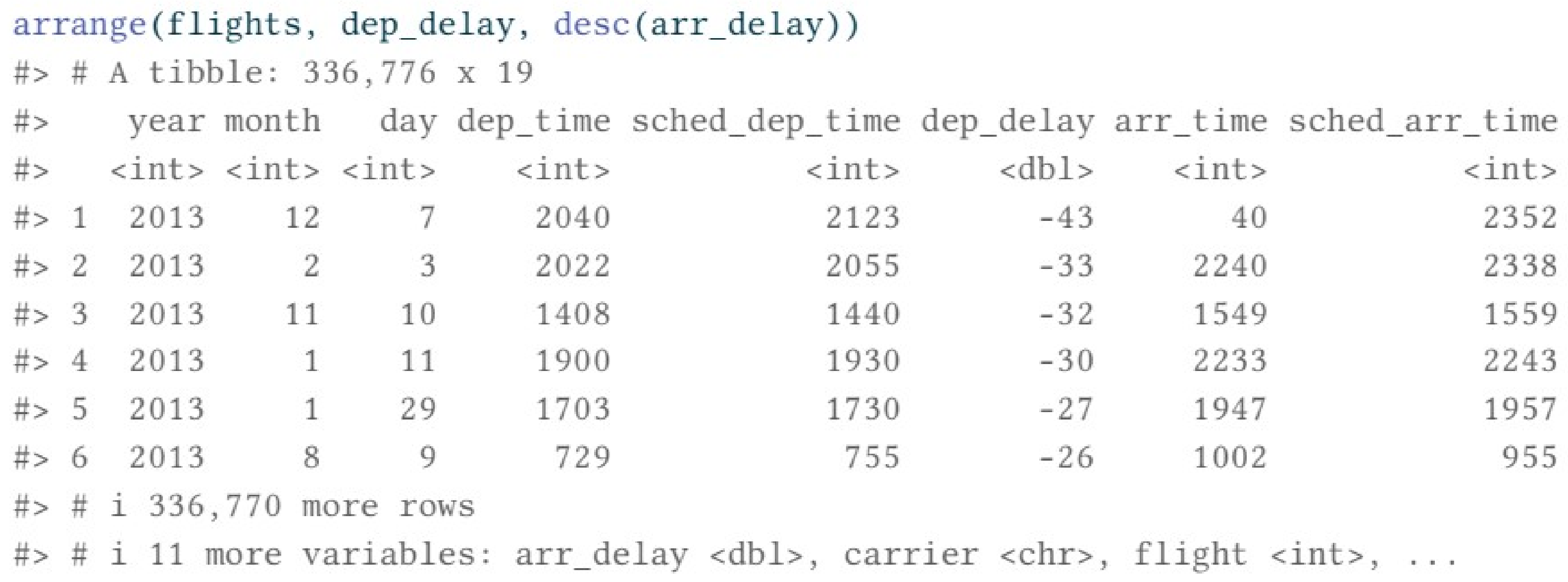
flights |> arrange(desc(arr_delay))

```
#> # 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
#> 4  2013     9    20     1139           1845      1014     1457           2210
#> 5  2013     7    22      845           1600      1005     1044           1815
#> 6  2013     4    10     1100           1900       960     1342           2211
#> # i 336,770 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
```

# DATA 100,Week 3 (A)

Multiple columns: sorted from left to right

arrange(flights, dep_delay, desc(arr_delay))



```
arrange(flights, dep_delay, desc(arr_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    12     7     2040           2123       -43       40           2352
#> 2  2013     2     3     2022           2055       -33     2240           2338
#> 3  2013    11    10     1408           1440       -32     1549           1559
#> 4  2013     1    11     1900           1930       -30     2233           2243
#> 5  2013     1    29     1703           1730       -27     1947           1957
#> 6  2013     8     9      729            755       -26     1002            955
#> # i 336,770 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
```

## DATA 100,Week 3 (A)

Where did **NA** go?

• **NA** values in dep_delay are all arranged to the end no matter what order we use.

• tail gives the last given number of rows

• head gives the first given number of rows

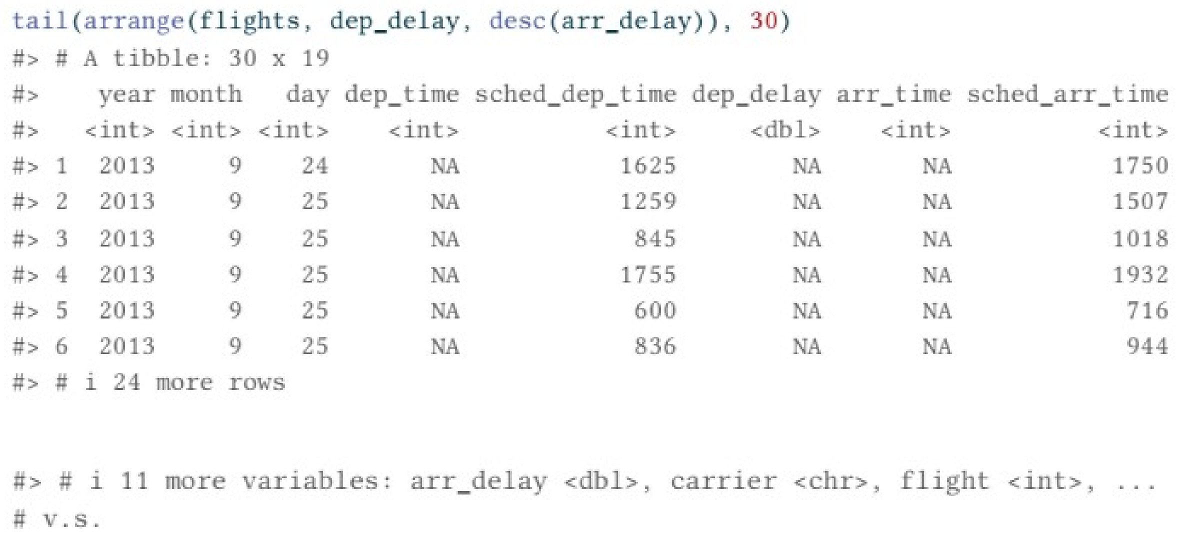
• check them out by ?tail or ?head

Page 75

## DATA 100,Week 3 (A)

The following comparison illustrates some benefits (in terms of readability) of using pipe:

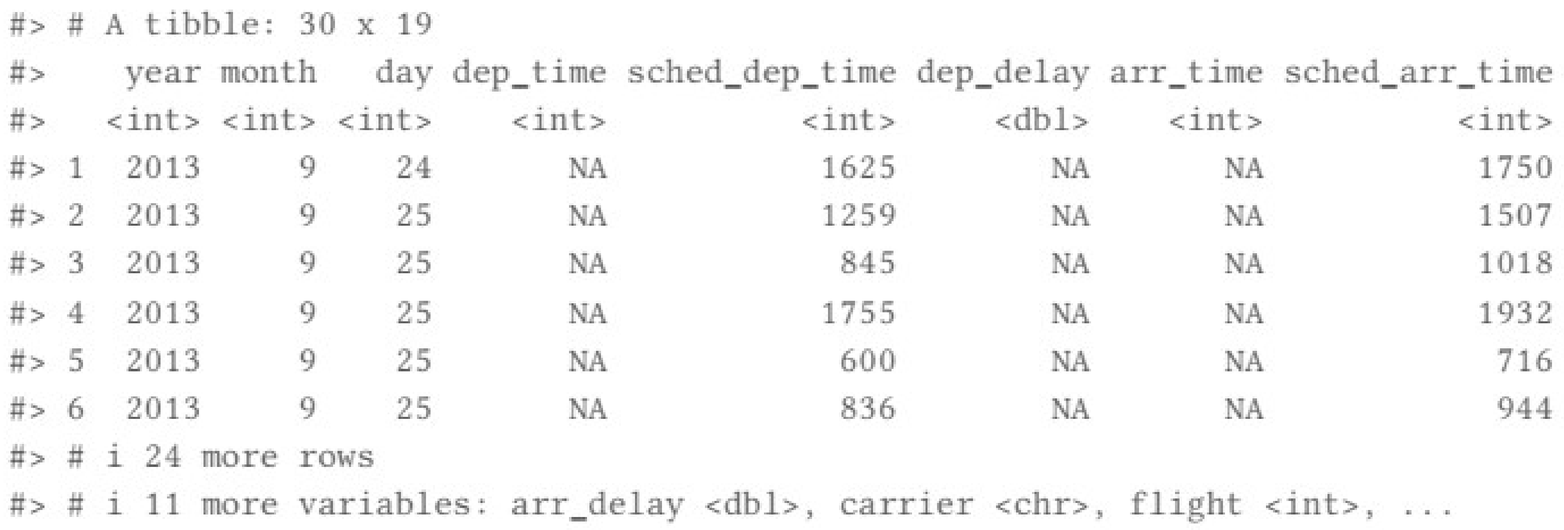
```r
tail(arrange(flights, dep_delay, desc(arr_delay)), 30)
```



```
tail(arrange(flights, dep_delay, desc(arr_delay)), 30)
#> # A tibble: 30 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     9    24       NA           1625        NA       NA           1750
#> 2  2013     9    25       NA           1259        NA       NA           1507
#> 3  2013     9    25       NA            845        NA       NA           1018
#> 4  2013     9    25       NA           1755        NA       NA           1932
#> 5  2013     9    25       NA            600        NA       NA            716
#> 6  2013     9    25       NA            836        NA       NA            944
#> # i 24 more rows


#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
# v.s.
```

## DATA 100,Week 3 (A)

```
flights |>
arrange(dep_delay, desc(arr_delay)) |>
tail(30)
```

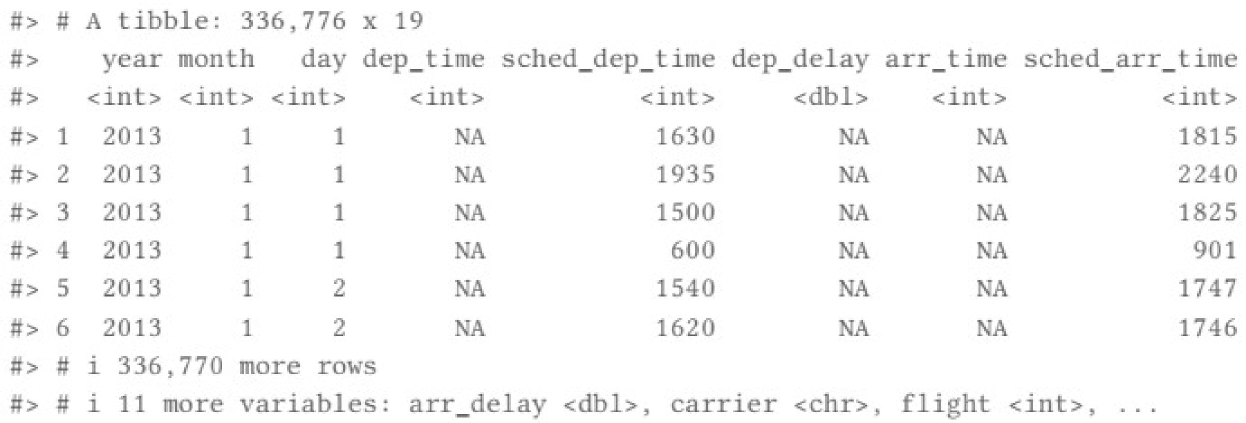


```
#> # A tibble: 30 x 19
#>     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
#>    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
#> 1   2013     9    24       NA           1625        NA       NA           1750
#> 2   2013     9    25       NA           1259        NA       NA           1507
#> 3   2013     9    25       NA            845        NA       NA           1018
#> 4   2013     9    25       NA           1755        NA       NA           1932
#> 5   2013     9    25       NA            600        NA       NA            716
#> 6   2013     9    25       NA            836        NA       NA            944
#> # i 24 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
```

## DATA 100,Week 3 (A)

Also, formatting the code well makes it easy to try and see:

```
flights |>
arrange(
desc(is.na(dep_delay)),
#dep_delay, # try desc()
desc(arr_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     1       NA           1630        NA       NA           1815
#> 2  2013     1     1       NA           1935        NA       NA           2240
#> 3  2013     1     1       NA           1500        NA       NA           1825
#> 4  2013     1     1       NA            600        NA       NA            901
#> 5  2013     1     2       NA           1540        NA       NA           1747
#> 6  2013     1     2       NA           1620        NA       NA           1746
#> # i 336,770 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
```

## DATA 100,Week 3 (A)

arrange **basic syntax:**

DATAFRAME |> arrange(COLUMNS to be reordered separated by comma)

Use desc(COLUMN) when necessary. ?arrange() for more.

Can arrange by computed values as well (# 3 in 2e, 4.2.5 Exercises).

Use desc(is.na(COLUMN)) appropriately to bring NA up top.

## DATA 100,Week 3 (A)

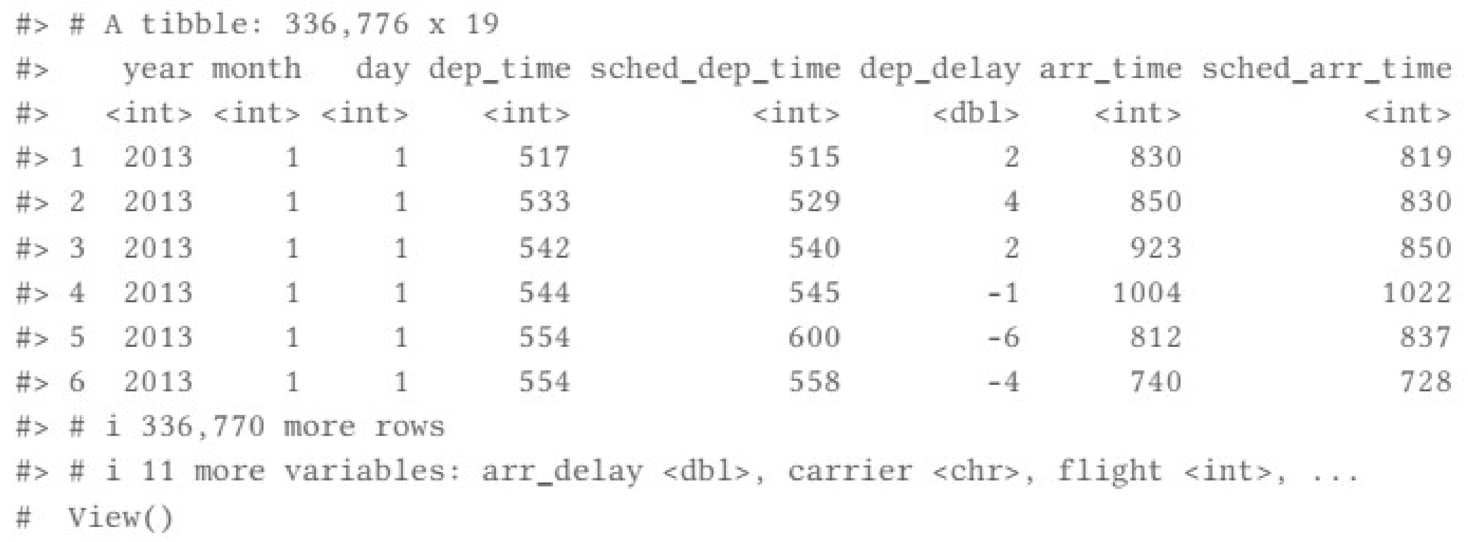
distinct()

- ✓ As the name suggests, the output of distinct() **contains rows that are distinct**, maybe only in values of a collection of columns that are specified in the parameters.

- ✓ Sometimes **whole rows may be duplicated during manipulation of data** and this functions can be very handy.

# DATA 100,Week 3 (A)

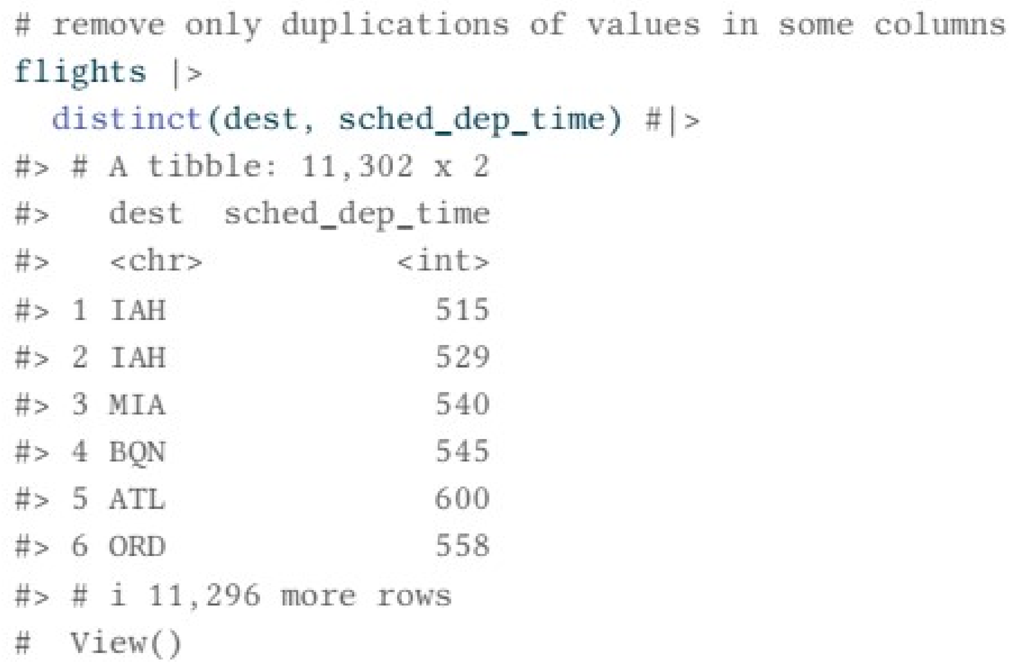
```
# remove duplicated rows, if there are any

flights |>
distinct() #|>
```



```
#> # 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
#> 5  2013     1     1      554            600        -6      812            837
#> 6  2013     1     1      554            558        -4      740            728
#> # i 336,770 more rows
#> # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, ...
#  View()
```

## DATA 100,Week 3 (A)

```
# remove only duplications of values in some columns
flights |>
distinct(dest, sched_dep_time) #|>
```



```
# remove only duplications of values in some columns
flights |>
  distinct(dest, sched_dep_time) #|>
#> # A tibble: 11,302 x 2
#>   dest  sched_dep_time
#>   <chr>          <int>
#> 1 IAH              515
#> 2 IAH              529
#> 3 MIA              540
#> 4 BQN              545
#> 5 ATL              600
#> 6 ORD              558
#> # i 11,296 more rows
#  View()
```

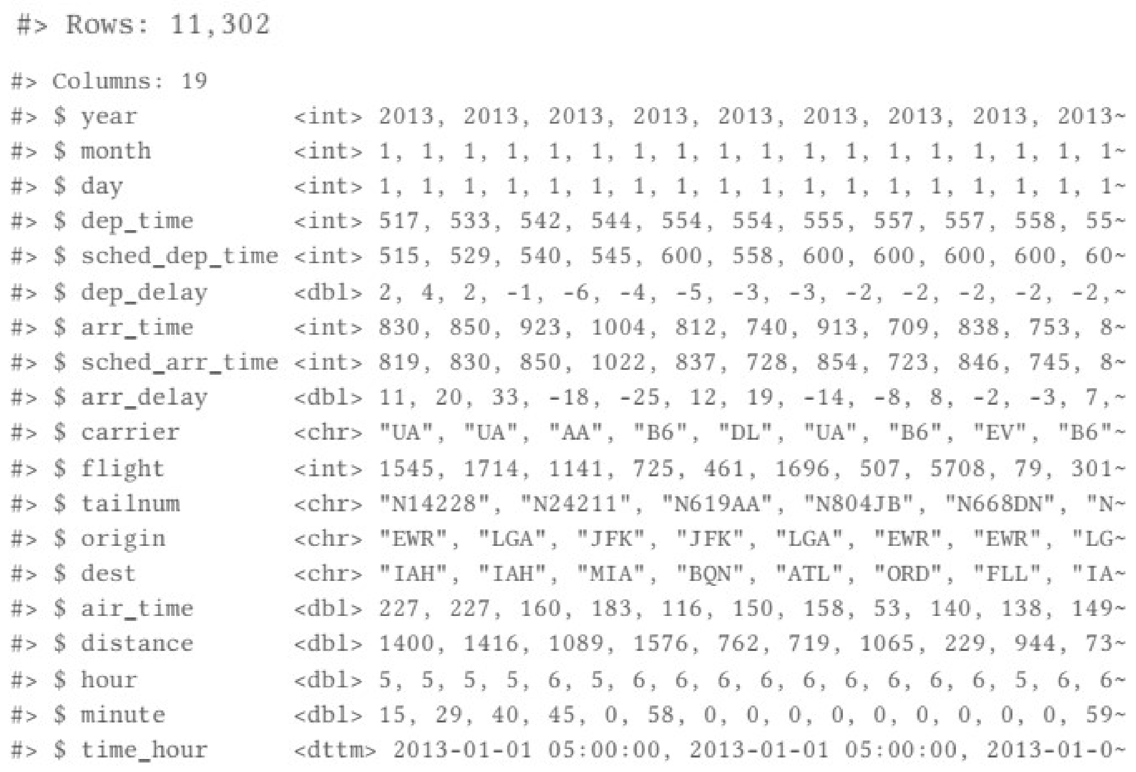
## DATA 100,Week 3 (A)

**Notice** that when columns are specified, the *default* is to only keep the specified columns in the output.

• To keep all the columns, use .keep_all = TRUE. Some internal algorithms are used to determine which rows are included in the result

# DATA 100,Week 3 (A)

```
flights |>
distinct(dest, sched_dep_time, .keep_all = TRUE) |>
glimpse()
```



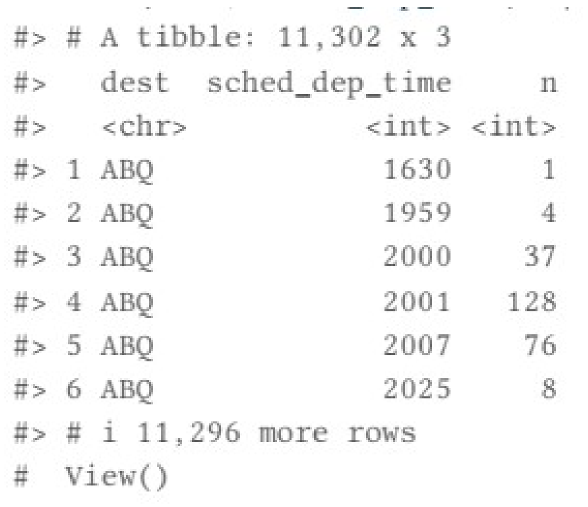
```
#> Rows: 11,302
#> Columns: 19
#> $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013~
#> $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 55~
#> $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 60~
#> $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2,~
#> $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 8~
#> $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 8~
#> $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7,~
#> $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6"~
#> $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301~
#> $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N~
#> $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LG~
#> $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IA~
#> $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149~
#> $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 73~
#> $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6~
#> $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59~
#> $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-0~
```

## DATA 100,Week 3 (A)

count()

Related to distinct(), using count() can get the *number* of rows that has the same values in given columns.

```
# count how many flights are scheduled at a given time of the day to a given destination flights |>
count(dest, sched_dep_time) #|>
```



```
#> # A tibble: 11,302 x 3
#>   dest  sched_dep_time     n
#>   <chr>          <int> <int>
#> 1 ABQ             1630     1
#> 2 ABQ             1959     4
#> 3 ABQ             2000    37
#> 4 ABQ             2001   128
#> 5 ABQ             2007    76
#> 6 ABQ             2025     8
#> # i 11,296 more rows
#  View()
```

## DATA 100,Week 3 (A)

mutate()

Analysis means to answer questions, which implies to **create new data from existing ones**

- the average speed in air
- the gain in air time
- the number of flights that were delayed on any given day
- the number of flights that were delayed going to any given destination
- … etc

## DATA 100,Week 3 (A)

**Operations are generally *vectorized*, i.e. automatically done for all observations / rows**

• the formula **distance/air_time** would compute the **average speed** for all flights

• the formula **arr_delay - dep_delay** would compute the **gain in air time** for all flights

To keep the computed results in the dataframe, use **mutate()**

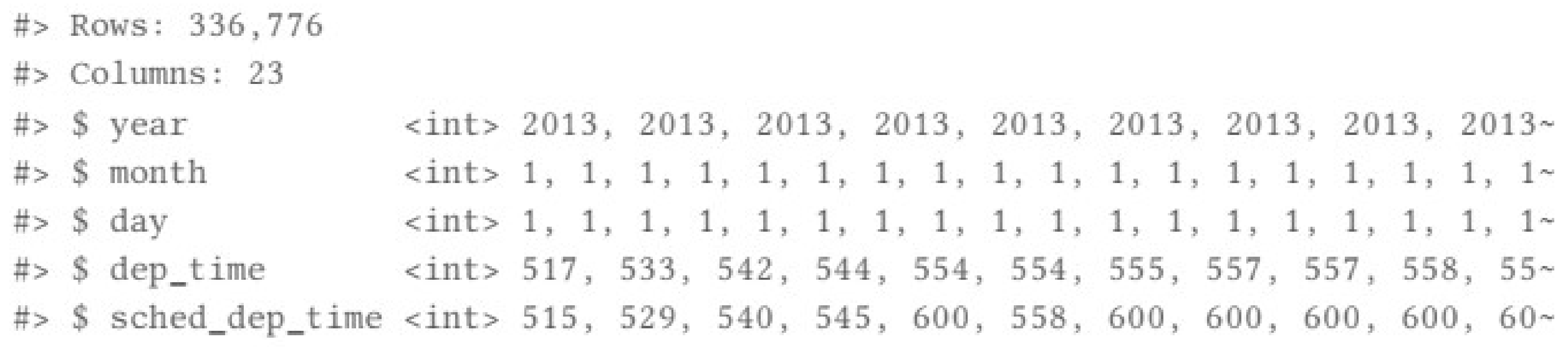
• with meaningful names for the new variables

# DATA 100,Week 3 (A)

```
flights_speed <- flights |>
mutate(
gain = arr_delay - dep_delay, hours = air_time / 60,
speed = distance / air_time * 60, # average speed per hour, could use `/ hour`
gain_per_hour = gain / hours )
```

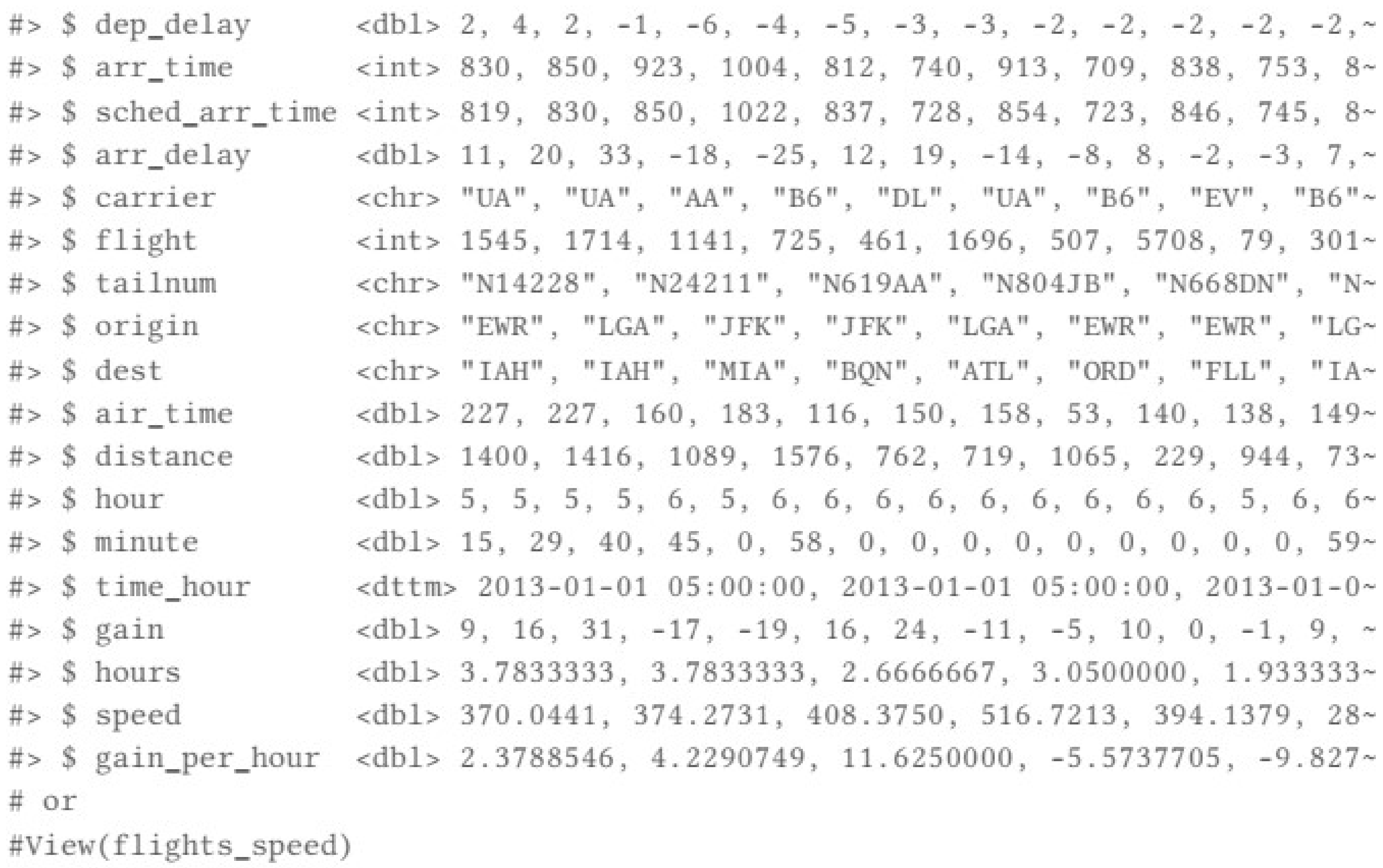
Take a look at the new dataframe

```
glimpse(flights_speed)
```



```
#> Rows: 336,776
#> Columns: 23
#> $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013~
#> $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 55~
#> $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 60~
```

## DATA 100,Week 3 (A)



```
#> $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2,~
#> $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 8~
#> $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 8~
#> $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7,~
#> $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6"~
#> $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301~
#> $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N~
#> $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LG~
#> $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IA~
#> $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149~
#> $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 73~
#> $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6~
#> $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59~
#> $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-0~
#> $ gain           <dbl> 9, 16, 31, -17, -19, 16, 24, -11, -5, 10, 0, -1, 9, ~
#> $ hours          <dbl> 3.7833333, 3.7833333, 2.6666667, 3.0500000, 1.933333~
#> $ speed          <dbl> 370.0441, 374.2731, 408.3750, 516.7213, 394.1379, 28~
#> $ gain_per_hour  <dbl> 2.3788546, 4.2290749, 11.6250000, -5.5737705, -9.827~
# or
#View(flights_speed)
```

## DATA 100,Week 3 (A)

More detailed control of the columns in the new dataframe, using parameters with . in their names.

Again, use ?mutate() for more information.

• Position: like relocate()

– Default position of the newly created columns are at the (right) end

– .before add the newly created columns right before (to the left of) a certain column

– .after add the newly created columns right after (to the right of) a certain column

## DATA 100,Week 3 (A)

• Choice: like select()

– .keep: can have value

✓ all (default),

✓ used (only the ones listed when calling the function),

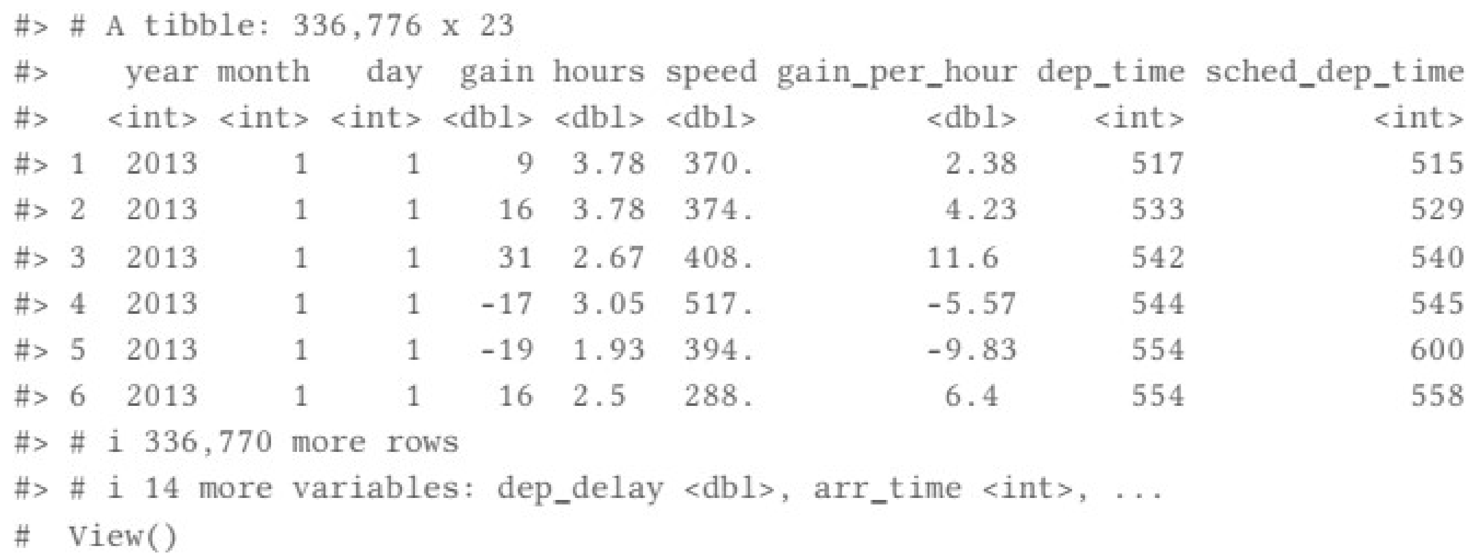
✓ unused (remove the ones listed when calling the function) and

✓ none (only the newly created columns)

## DATA 100,Week 3 (A)

```
flights |>
mutate(
gain = arr_delay - dep_delay, hours = air_time / 60,
speed = distance / air_time * 60, # average speed per hour, could use `/ hour`
gain_per_hour = gain / hours,
.before = 4,
# .after = tailnum,
# .keep = "used"
) #|>
```

# DATA 100,Week 3 (A)



```
#> # A tibble: 336,776 x 23
#>    year month   day  gain hours speed gain_per_hour dep_time sched_dep_time
#>   <int> <int> <int> <dbl> <dbl> <dbl>         <dbl>    <int>          <int>
#> 1  2013     1     1     9  3.78  370.          2.38      517            515
#> 2  2013     1     1    16  3.78  374.          4.23      533            529
#> 3  2013     1     1    31  2.67  408.         11.6       542            540
#> 4  2013     1     1   -17  3.05  517.         -5.57      544            545
#> 5  2013     1     1   -19  1.93  394.         -9.83      554            600
#> 6  2013     1     1    16  2.5   288.          6.4       554            558
#> # i 336,770 more rows
#> # i 14 more variables: dep_delay <dbl>, arr_time <int>, ...
#  View()
```

## DATA 100,Week 3 (A)

select()

Too many columns to get *quickly* to the variable that we are interested in

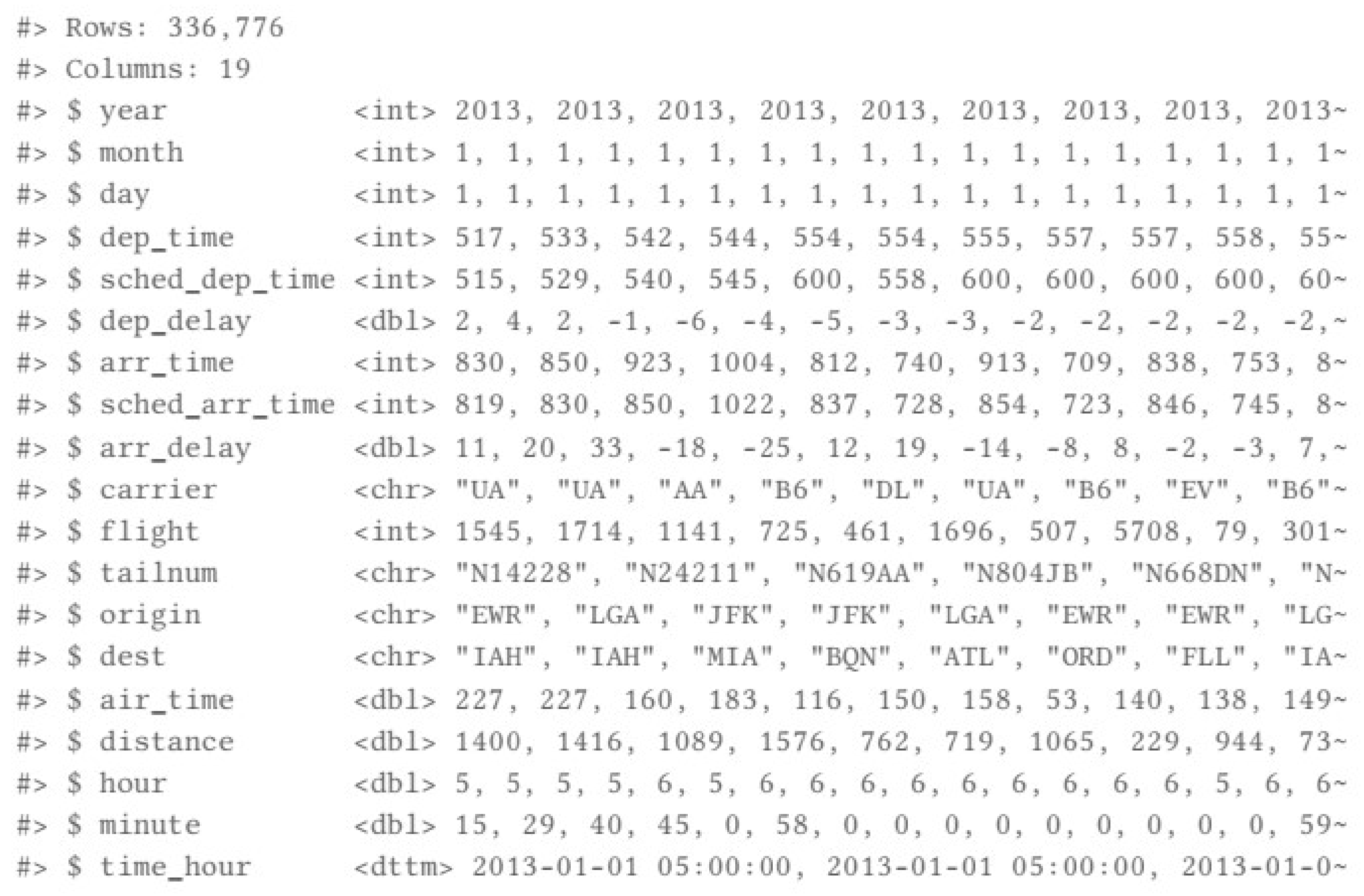
• select() operates on the *names* of the variable, or equivalently, the column head

– Thus can use string functions for comparison and matching

Compare to filter() and arrange() that operates on the *values* of the variables, i.e. on the *rows*

## DATA 100,Week 3 (A)

glimpse(flights)



```
#> Rows: 336,776
#> Columns: 19
#> $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013~
#> $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 55~
#> $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 60~
#> $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2,~
#> $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 8~
#> $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 8~
#> $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7,~
#> $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6"~
#> $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301~
#> $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N~
#> $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LG~
#> $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IA~
#> $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149~
#> $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 73~
#> $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6~
#> $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59~
#> $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-0~
```

## DATA 100,Week 3 (A)

select **the fields related to how fast the flight goes**

The **order matters** inside the bracket of select().

The output dataframe will have the columns in the order the way they are specified in select()

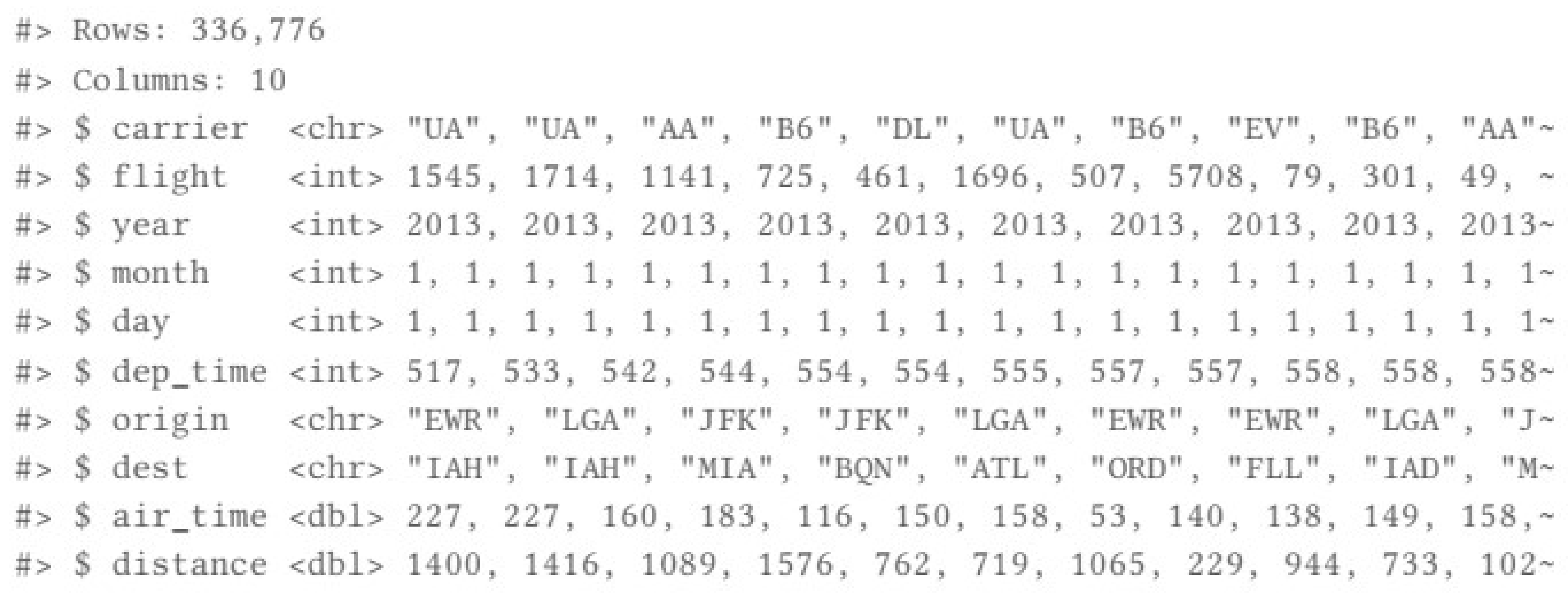
• all the **un**selected fields are dropped from the output

COLUMN_NAME:COLUMN_NAME: takes all columns between (and including) the two columns given.

• Just like 1:15 takes all integers between 1 and 15, including 1 and 15

## DATA 100,Week 3 (A)

```
flights |>
select(carrier, flight, year:dep_time, origin:distance) |>
glimpse()
```

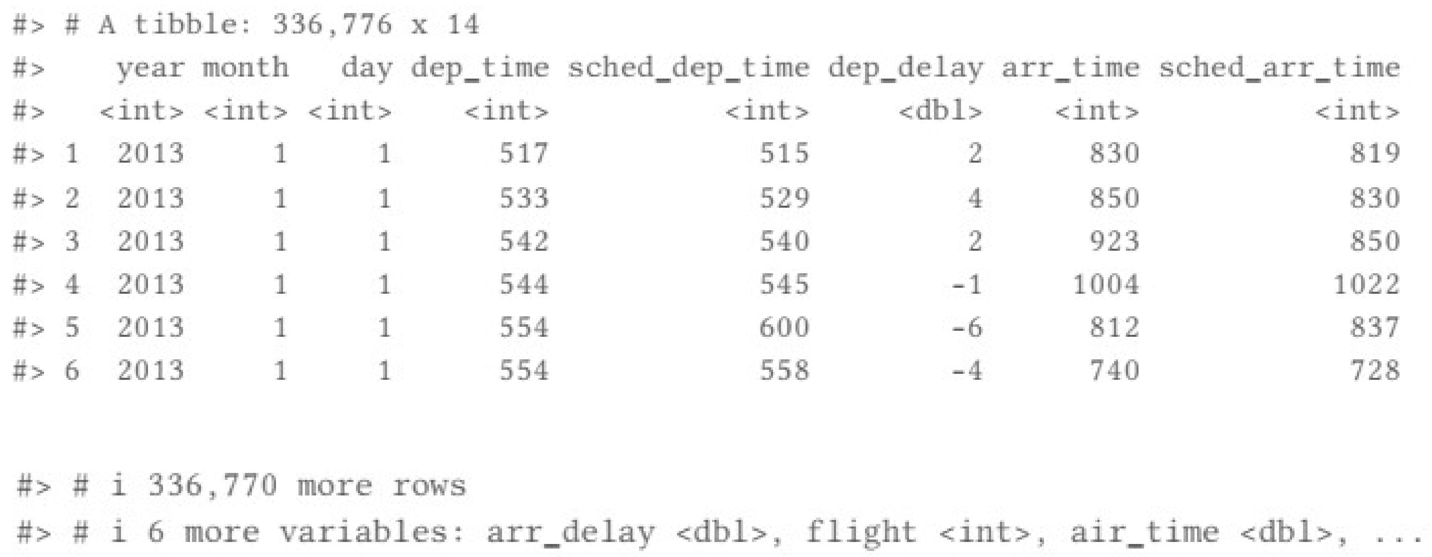


```
#> Rows: 336,776
#> Columns: 10
#> $ carrier  <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "AA"~
#> $ flight   <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 49, ~
#> $ year     <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013~
#> $ month    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ day      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ dep_time <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, 558~
#> $ origin   <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA", "J~
#> $ dest     <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD", "M~
#> $ air_time <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 158,~
#> $ distance <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, 102~
```

## DATA 100,Week 3 (A)

Can select by type of values using where(is.____).

```
flights |>
select( where(is.numeric),
# where(is.character),
# where(is.Date)
)
```



```
#> # A tibble: 336,776 x 14
#>    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
#> 5  2013     1     1      554            600        -6      812            837
#> 6  2013     1     1      554            558        -4      740            728

#> # i 336,770 more rows
#> # i 6 more variables: arr_delay <dbl>, flight <int>, air_time <dbl>, ...
```

## DATA 100,Week 3 (A)

### Removing some columns that won’t be used

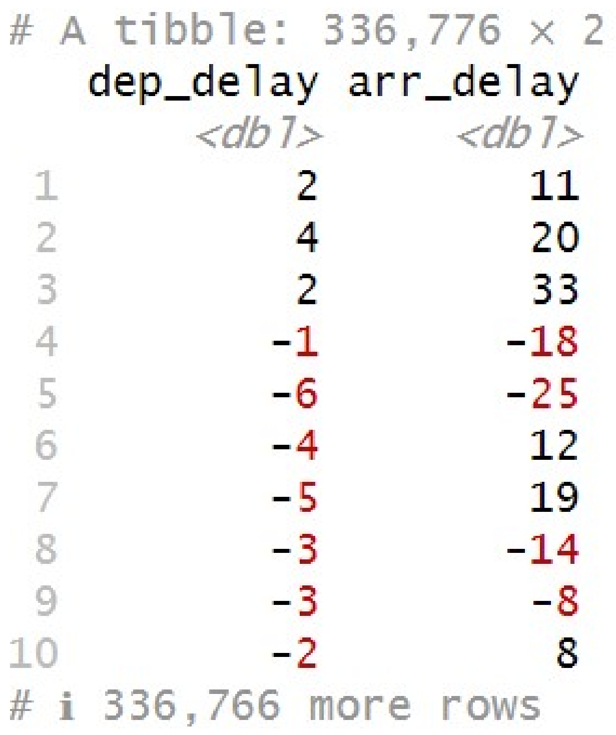
Use !c(...) (or -c(...)) to remove columns in the parenthesis, i.e. select *against* them.

- **Cannot** mix select *for* and select *against*

- i.e.: flights |> select(dep_delay, !c(hour:time_hour)) is **not expected to work**

# DATA 100,Week 3 (A)

```
flights |> select(contains("delay"))
```



```
# A tibble: 336,776 × 2
   dep_delay arr_delay
       <dbl>     <dbl>
 1         2        11
 2         4        20
 3         2        33
 4        -1       -18
 5        -6       -25
 6        -4        12
 7        -5        19
 8        -3       -14
 9        -3        -8
10        -2         8
# ℹ 336,766 more rows
```

## DATA 100,Week 3 (A)

• Other useful string functions, as included in 2e, section 4.3.2

– starts_with()

– matches(): regular expression, which will be discussed later with stringr

• and choice functions such as one_of()

• num_range("x", 1:3) can be useful, but hopefully not in select() – bad variable name, IMO.

All from package tidy-select … try ?contains() in the Console

## DATA 100,Week 3 (A)

***Reusability*, or** try not to repeat

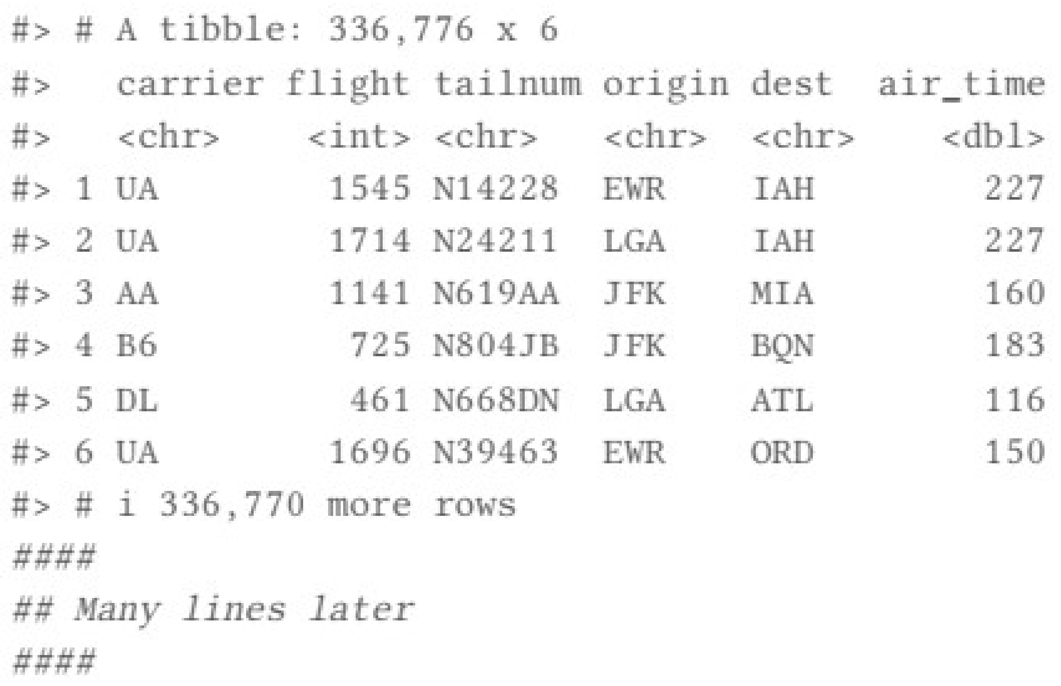
Suppose that a decision is made to always keep the columns carrier, flight and tailnum in all the analysis

• while only keep necessary variables for various different tasks

Could do

```
# select a bunch of columns and work with them
flights |>
select(carrier, flight, tailnum, origin, dest, air_time)
```

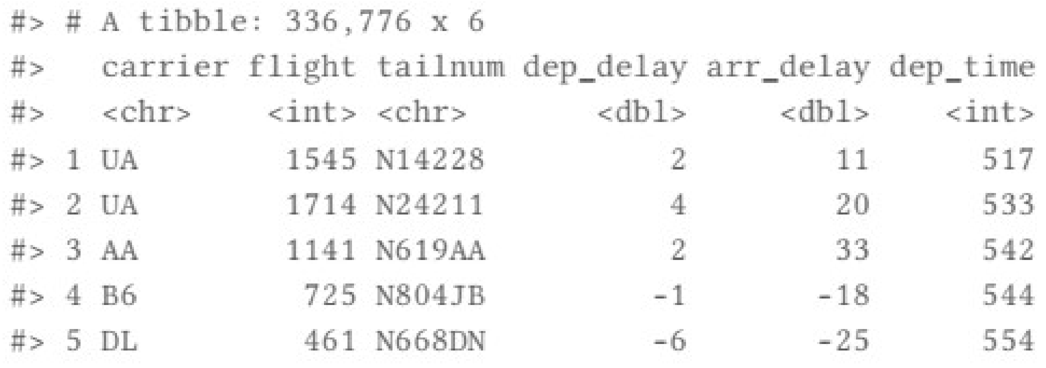
# DATA 100,Week 3 (A)



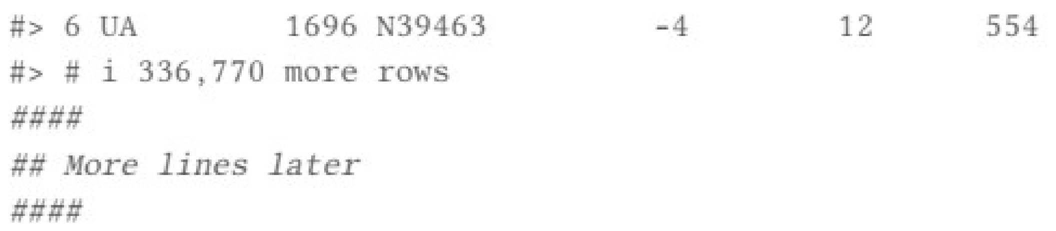
```
#> # A tibble: 336,776 x 6
#>   carrier flight tailnum origin dest  air_time
#>   <chr>    <int> <chr>   <chr>  <chr>    <dbl>
#> 1 UA        1545 N14228  EWR    IAH        227
#> 2 UA        1714 N24211  LGA    IAH        227
#> 3 AA        1141 N619AA  JFK    MIA        160
#> 4 B6         725 N804JB  JFK    BQN        183
#> 5 DL         461 N668DN  LGA    ATL        116
#> 6 UA        1696 N39463  EWR    ORD        150
#> # i 336,770 more rows
####
## Many lines later
####
```

# DATA 100,Week 3 (A)

```
flights |>
select(carrier, flight, tailnum, dep_delay, arr_delay, dep_time)
```



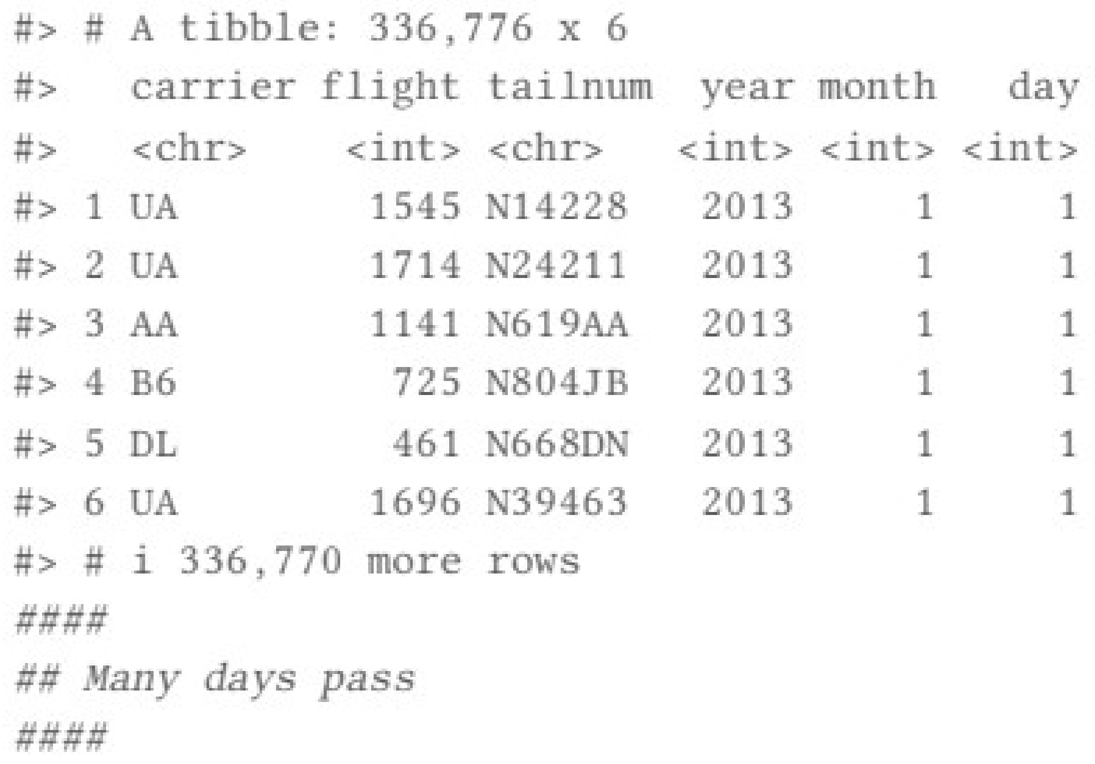
```
#> # A tibble: 336,776 x 6
#>   carrier flight tailnum dep_delay arr_delay dep_time
#>   <chr>    <int> <chr>       <dbl>     <dbl>    <int>
#> 1 UA        1545 N14228          2        11      517
#> 2 UA        1714 N24211          4        20      533
#> 3 AA        1141 N619AA          2        33      542
#> 4 B6         725 N804JB         -1       -18      544
#> 5 DL         461 N668DN         -6       -25      554
```



```
#> 6 UA        1696 N39463         -4        12      554
#> # i 336,770 more rows
####
## More lines later
####
```

## DATA 100,Week 3 (A)

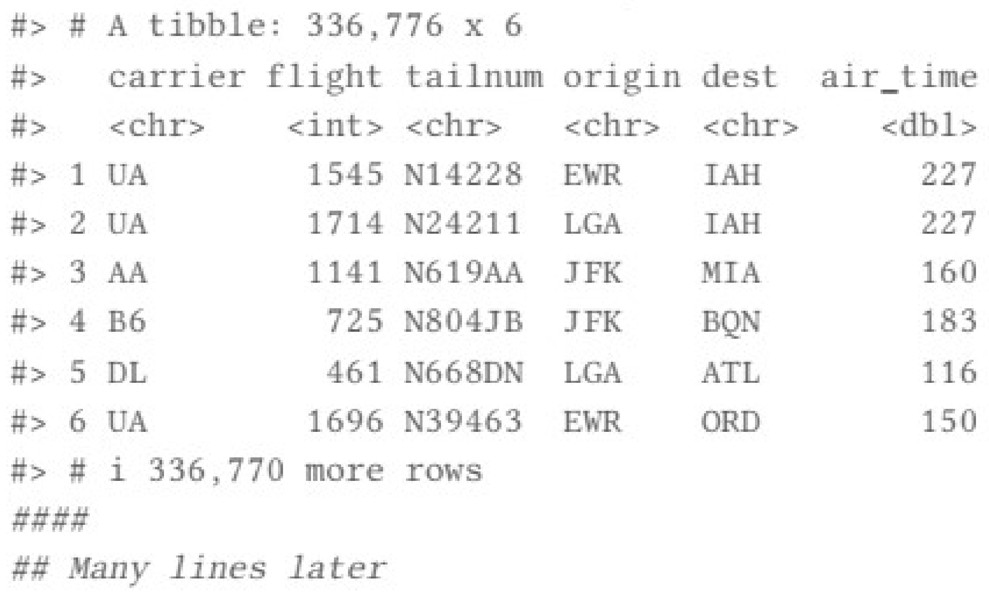
```
flights |>
select(carrier, flight, tailnum, year, month, day)
```



```
#> # A tibble: 336,776 x 6
#>   carrier flight tailnum  year month   day
#>   <chr>    <int> <chr>   <int> <int> <int>
#> 1 UA        1545 N14228   2013     1     1
#> 2 UA        1714 N24211   2013     1     1
#> 3 AA        1141 N619AA   2013     1     1
#> 4 B6         725 N804JB   2013     1     1
#> 5 DL         461 N668DN   2013     1     1
#> 6 UA        1696 N39463   2013     1     1
#> # i 336,770 more rows
####
## Many days pass
####
```

# DATA 100,Week 3 (A)

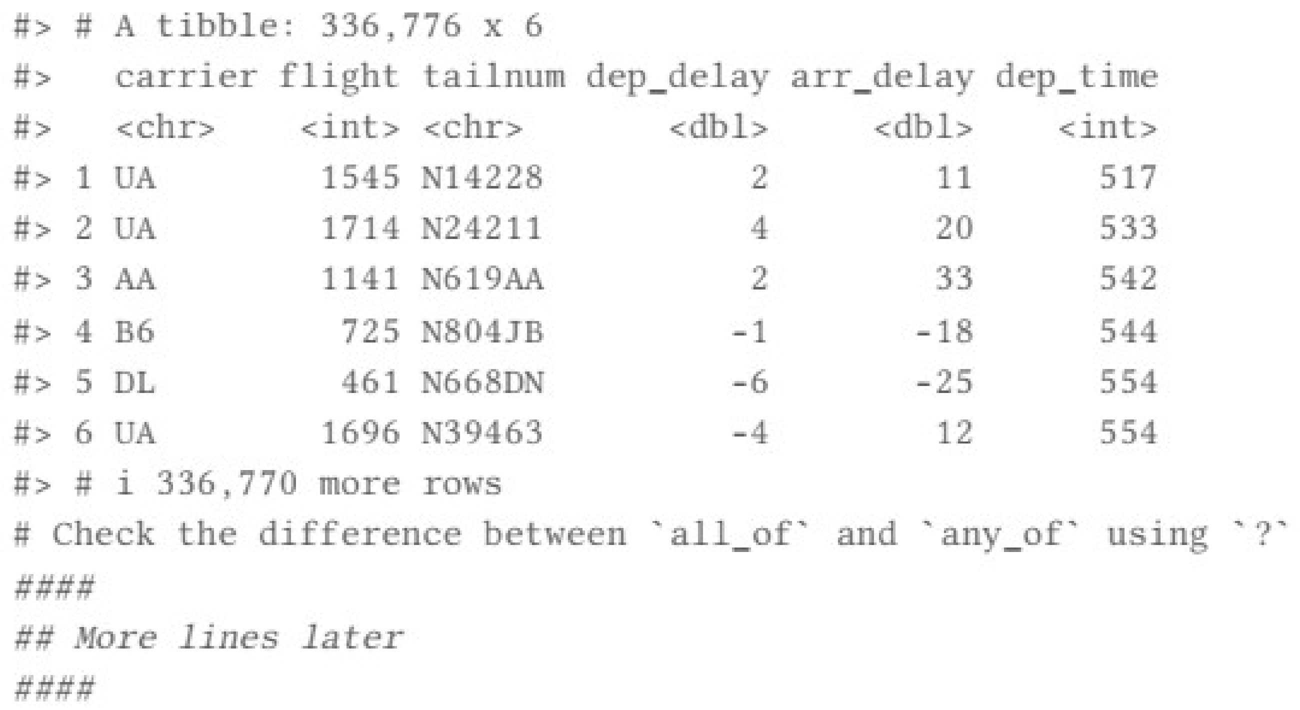
```
must_have <- c("carrier", "flight", "tailnum")
####
## whatever happens ####

flights |>
select(all_of(must_have), origin, dest, air_time)
```



```
#> # A tibble: 336,776 x 6
#>   carrier flight tailnum origin dest  air_time
#>   <chr>    <int> <chr>   <chr>  <chr>    <dbl>
#> 1 UA        1545 N14228  EWR    IAH        227
#> 2 UA        1714 N24211  LGA    IAH        227
#> 3 AA        1141 N619AA  JFK    MIA        160
#> 4 B6         725 N804JB  JFK    BQN        183
#> 5 DL         461 N668DN  LGA    ATL        116
#> 6 UA        1696 N39463  EWR    ORD        150
#> # i 336,770 more rows
####
## Many lines later
```

# DATA 100,Week 3 (A)

```
flights |>
select(any_of(must_have), dep_delay, arr_delay, dep_time)
```



```
#> # A tibble: 336,776 x 6
#>   carrier flight tailnum dep_delay arr_delay dep_time
#>   <chr>    <int> <chr>       <dbl>     <dbl>    <int>
#> 1 UA        1545 N14228          2        11      517
#> 2 UA        1714 N24211          4        20      533
#> 3 AA        1141 N619AA          2        33      542
#> 4 B6         725 N804JB         -1       -18      544
#> 5 DL         461 N668DN         -6       -25      554
#> 6 UA        1696 N39463         -4        12      554
#> # i 336,770 more rows
# Check the difference between `all_of` and `any_of` using `?`
####
## More lines later
####
```

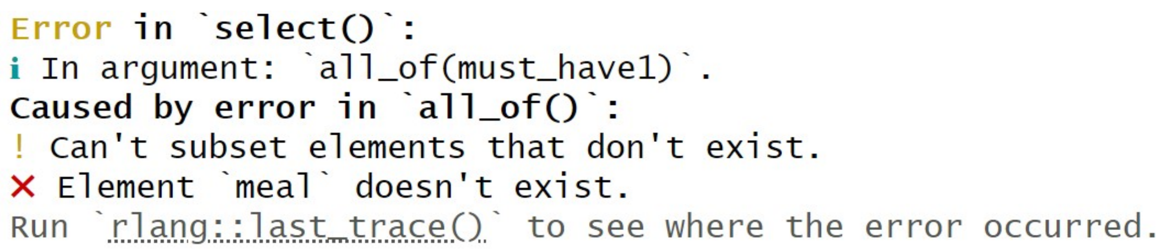
# DATA 100,Week 3 (A)

all_of() requires that all specified variable names exist in the data frame;

if some of the specified variables may not exist, and you want to select those that do exist, you should use any_of()

For example

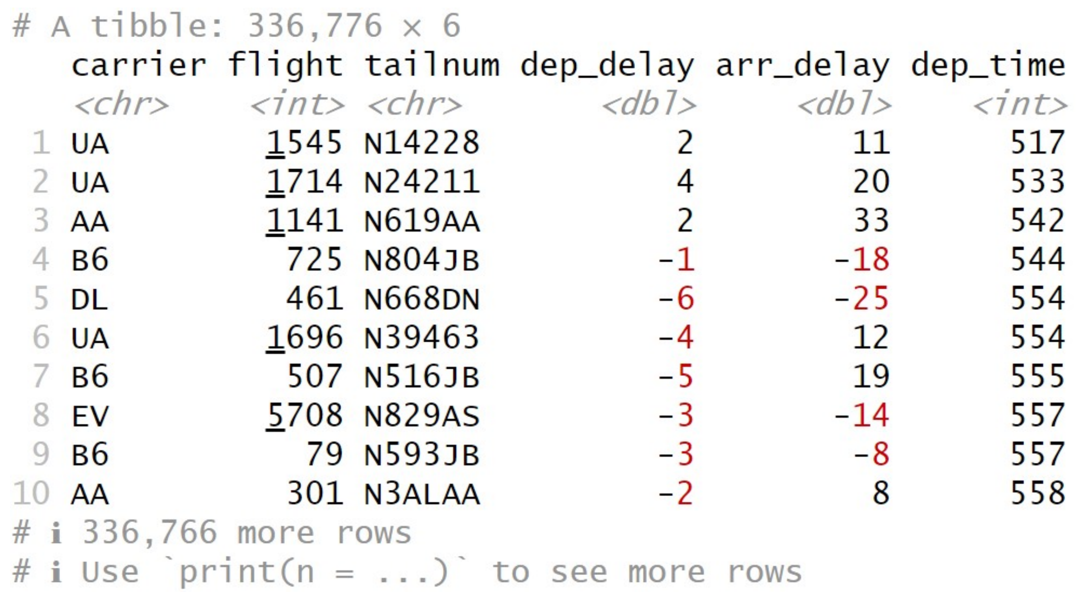
```
must_have1 <- c("carrier", "flight", "tailnum", "meal")

flights |>     select(all_of(must_have1), dep_delay, arr_delay, dep_time)
```



```
Error in `select()`:
i In argument: `all_of(must_have1)`.
Caused by error in `all_of()`:
! Can't subset elements that don't exist.
✕ Element `meal` doesn't exist.
Run `rlang::last_trace()` to see where the error occurred.
```

# DATA 100,Week 3 (A)

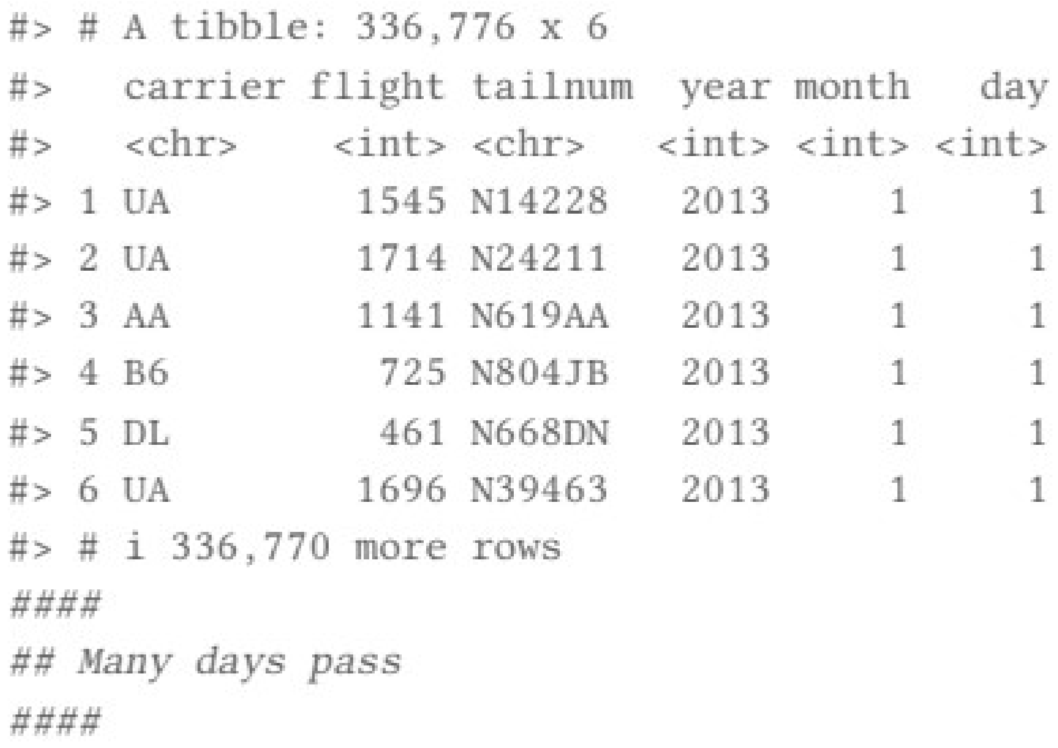
flights |>     select(any_of(must_have1), dep_delay, arr_delay, dep_time)



```
# A tibble: 336,776 × 6
   carrier flight tailnum dep_delay arr_delay dep_time
   <chr>    <int> <chr>       <dbl>     <dbl>    <int>
 1 UA        1545 N14228          2        11      517
 2 UA        1714 N24211          4        20      533
 3 AA        1141 N619AA          2        33      542
 4 B6         725 N804JB         -1       -18      544
 5 DL         461 N668DN         -6       -25      554
 6 UA        1696 N39463         -4        12      554
 7 B6         507 N516JB         -5        19      555
 8 EV        5708 N829AS         -3       -14      557
 9 B6          79 N593JB         -3        -8      557
10 AA         301 N3ALAA         -2         8      558
# ℹ 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
```

# DATA 100,Week 3 (A)

```
flights |>
select(all_of(must_have), year, month, day)
```



```
#> # A tibble: 336,776 x 6
#>   carrier flight tailnum  year month   day
#>   <chr>    <int> <chr>   <int> <int> <int>
#> 1 UA        1545 N14228   2013     1     1
#> 2 UA        1714 N24211   2013     1     1
#> 3 AA        1141 N619AA   2013     1     1
#> 4 B6         725 N804JB   2013     1     1
#> 5 DL         461 N668DN   2013     1     1
#> 6 UA        1696 N39463   2013     1     1
#> # i 336,770 more rows
####
## Many days pass
####
```

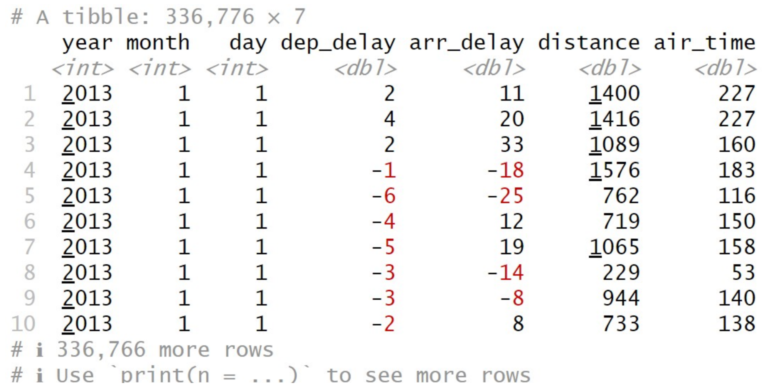
## DATA 100,Week 3 (A)

### Keep the changes

- Use <- to assign the revised dataframe to a new variable

```
flights_sml <- flights |>
select( year:day,
ends_with("delay"),
distance, air_time )
```

```
flights_sml
```



```
# A tibble: 336,776 × 7
    year month   day dep_delay arr_delay distance air_time
   <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl>
 1  2013     1     1         2        11     1400      227
 2  2013     1     1         4        20     1416      227
 3  2013     1     1         2        33     1089      160
 4  2013     1     1        -1       -18     1576      183
 5  2013     1     1        -6       -25      762      116
 6  2013     1     1        -4        12      719      150
 7  2013     1     1        -5        19     1065      158
 8  2013     1     1        -3       -14      229       53
 9  2013     1     1        -3        -8      944      140
10  2013     1     1        -2         8      733      138
# ℹ 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
```

## DATA 100,Week 3 (A)

rename()

Change column (variable) names.

It is useful for creating meaningful names, or making variable names conform to certain protocol.

• **Very** useful when dealing with data *in the wild*

– “Say what you mean and mean what you say”

– “Meaningful name is good name”

• We do not want to guess reading our own code
• We want others (read: **our future selves**) read our code with minimal amount of help

## DATA 100,Week 3 (A)

The following changes tailnum into tail_num —- the *new* name is on the left side of the = sign.

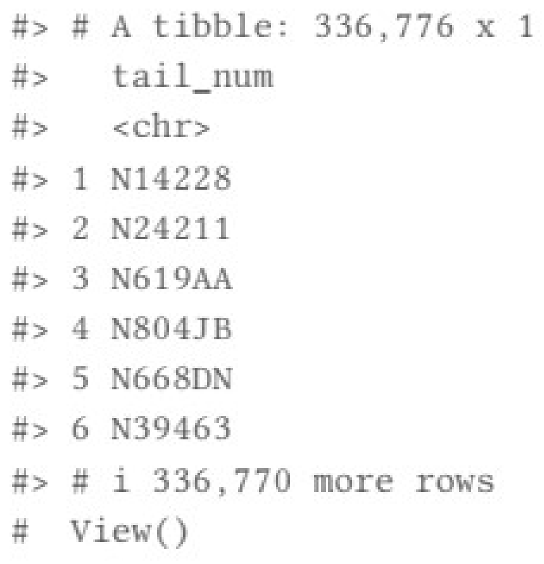
```
flights |>
rename(tail_num = tailnum) #|>
```

```
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tail_num origin
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>    <chr> 
 1  2013     1     1      517            515         2      830            819        11 UA        1545 N14228   EWR   
 2  2013     1     1      533            529         4      850            830        20 UA        1714 N24211   LGA   
 3  2013     1     1      542            540         2      923            850        33 AA        1141 N619AA   JFK   
 4  2013     1     1      544            545        -1     1004           1022       -18 B6         725 N804JB   JFK   
 5  2013     1     1      554            600        -6      812            837       -25 DL         461 N668DN   LGA   
 6  2013     1     1      554            558        -4      740            728        12 UA        1696 N39463   EWR   
 7  2013     1     1      555            600        -5      913            854        19 B6         507 N516JB   EWR   
 8  2013     1     1      557            600        -3      709            723       -14 EV        5708 N829AS   LGA   
 9  2013     1     1      557            600        -3      838            846        -8 B6          79 N593JB   JFK   
10  2013     1     1      558            600        -2      753            745         8 AA         301 N3ALAA   LGA   
# ℹ 336,766 more rows
# ℹ 6 more variables: dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
# ℹ Use `print(n = ...)` to see more rows
```

## DATA 100,Week 3 (A)

Can also rename *while* select. Compare the difference from above.

```
flights |>
select(tail_num = tailnum) #|>
```



```
#> # A tibble: 336,776 x 1
#>   tail_num
#>   <chr>
#> 1 N14228
#> 2 N24211
#> 3 N619AA
#> 4 N804JB
#> 5 N668DN
#> 6 N39463
#> # i 336,770 more rows
#  View()
```