STOR 320 Data Transformation I

Lecture 3

Yao Li

Department of Statistics and Operations Research

UNC Chapel Hill

Introduction

- Read Chapter 5
- Goal: Their Data → Your Data
- Covers:
  - Data Subsetting
  - Data Ordering
  - Variable Selecting
  - Variable Creating
- Help: dplyr Package in R

NYC Flights Meta Data

- Requirements:

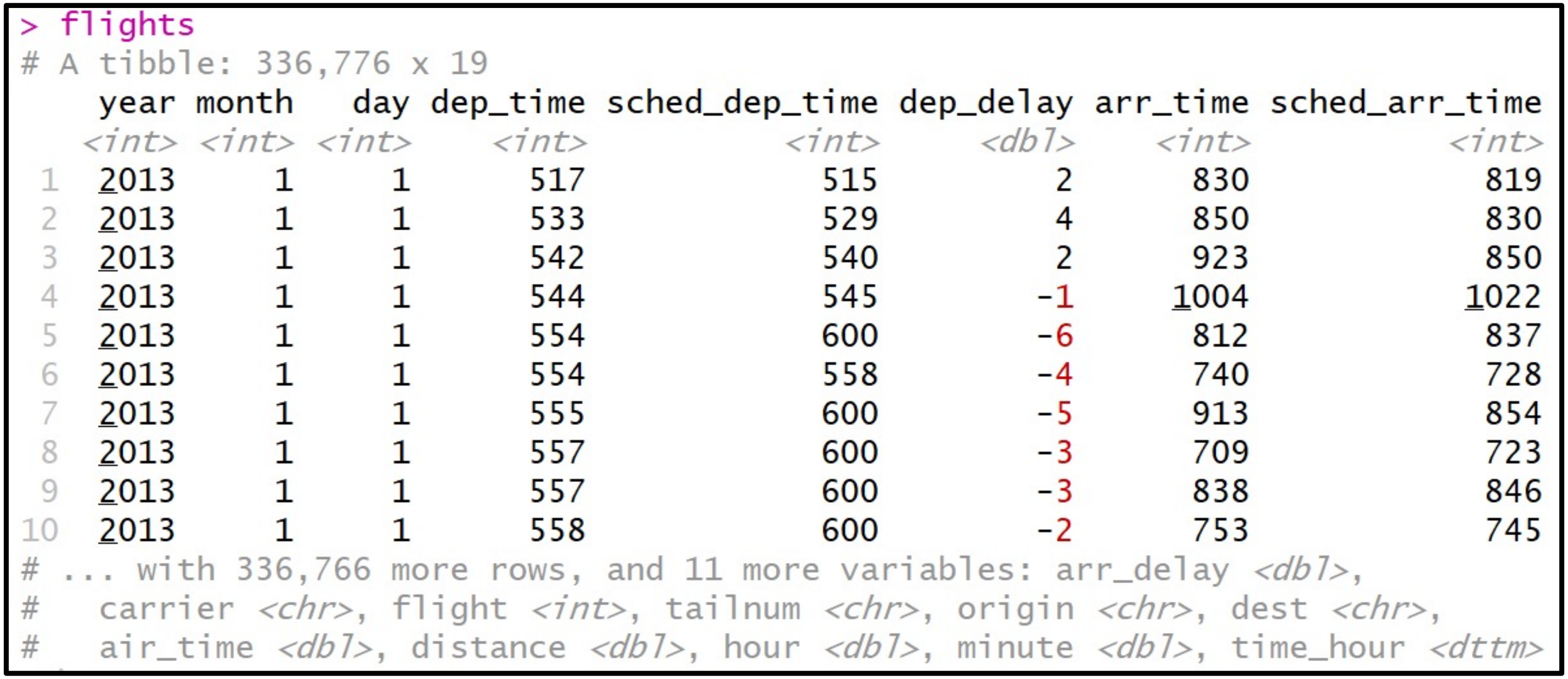
```
> install.packages(nycflights13)
> library(nycflights13)
```

- All 2013 Flights from NYC
  - US Bureau of Trans. Statistics

- To View all Data, Use `>  View(flights)`

- For more information, `> ?flights`

NYC Flights Data



```
> flights
# 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
 7  2013     1     1      555            600        -5      913            854
 8  2013     1     1      557            600        -3      709            723
 9  2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# ... with 336,766 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>
```

- Four Different Types of Variables
  - int = integer
  - dbl = double
  - chr = character
  - dttm = date and times

- Other Types of Variables
  - lgl = logical (TRUE or FALSE)
  - fctr = factor
  - date = dates

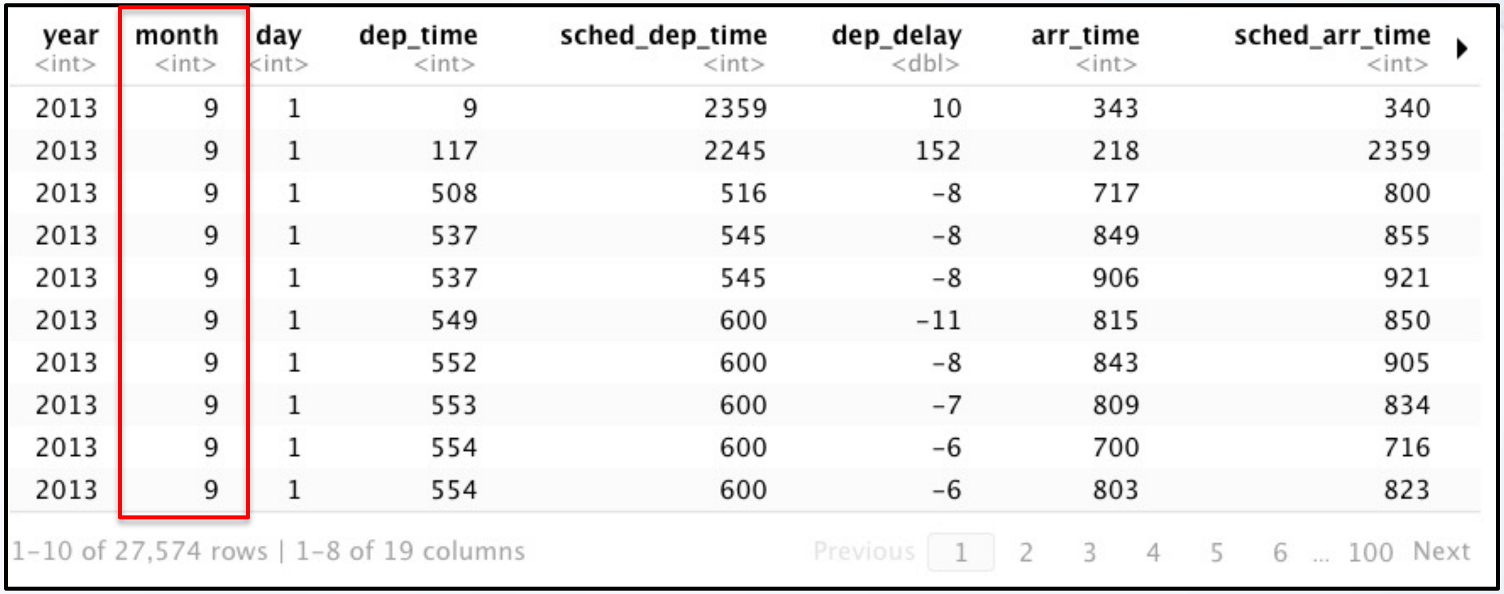
Basics of dplyr: 5 Key Functions

- 5 Key Functions
  - filter() = Chooses Observations Based on Values
  - arrange() = Sorts Observations
  - select() = Chooses Variables
  - mutate() = Creates New Variables
  - summarise() = Generates Statistics From Data

Basics of dplyr

- Function Usage
  - First, Specify the Dataset
  - Next, Specify What to Do with the Data
  - Result is a New Dataset

```
> filter(flights, month==9)
```

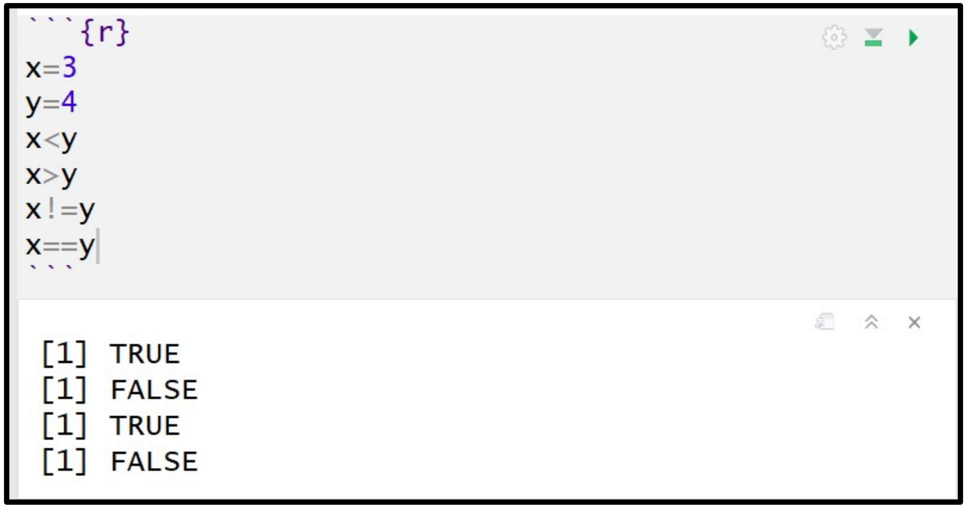


year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>
2013	9	1	9	2359	10	343	340
2013	9	1	117	2245	152	218	2359
2013	9	1	508	516	-8	717	800
2013	9	1	537	545	-8	849	855
2013	9	1	537	545	-8	906	921
2013	9	1	549	600	-11	815	850
2013	9	1	552	600	-8	843	905
2013	9	1	553	600	-7	809	834
2013	9	1	554	600	-6	700	716
2013	9	1	554	600	-6	803	823

1–10 of 27,574 rows | 1–8 of 19 columns

Comparisons

- Important Operators
  - Less Than (<)
  - Greater Than (>)
  - Not Equal (!=)
  - Equal (==)
- Returns TRUE or FALSE

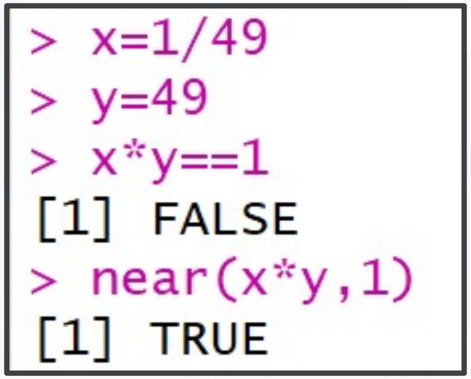


~~~
```{r}
x=3
y=4
x<y
x>y
x!=y
x==y
```
~~~

```
[1] TRUE
[1] FALSE
[1] TRUE
[1] FALSE
```

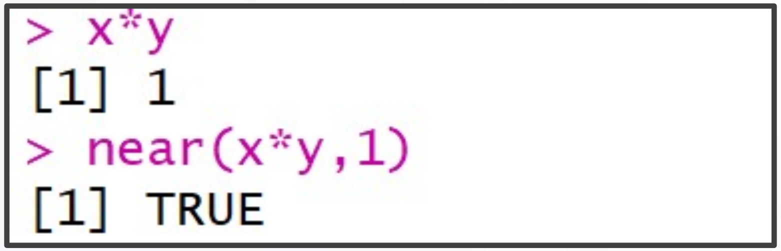
Numerical Precision

- Problem



```
> x=1/49
> y=49
> x*y==1
[1] FALSE
> near(x*y,1)
[1] TRUE
```

- Solution



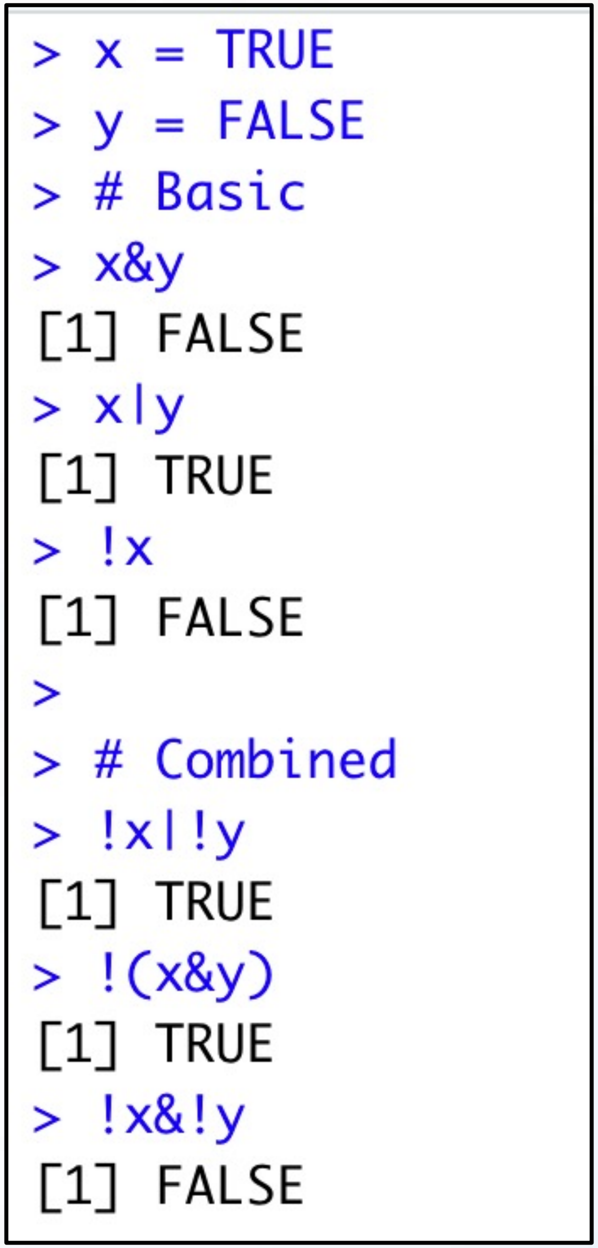
```
> x*y
[1] 1
> near(x*y,1)
[1] TRUE
```

Logical Operators

- Boolean Logic
  - And (&)
  - Or (|)
  - Not (!)

- Example →

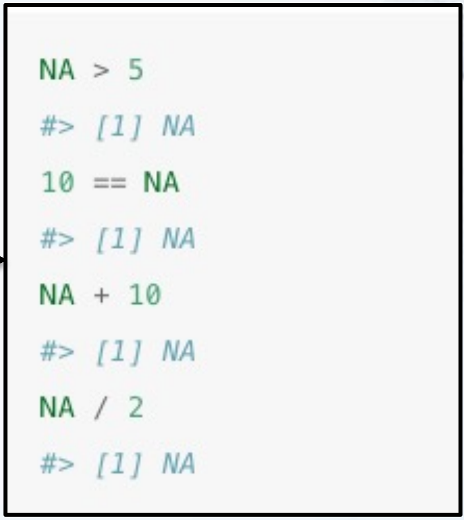
```
> filter(flights, month==9&day==1)
```



```
> x = TRUE
> y = FALSE
> # Basic
> x&y
[1] FALSE
> x|y
[1] TRUE
> !x
[1] FALSE
>
> # Combined
> !x|!y
[1] TRUE
> !(x&y)
[1] TRUE
> !x&!y
[1] FALSE
```

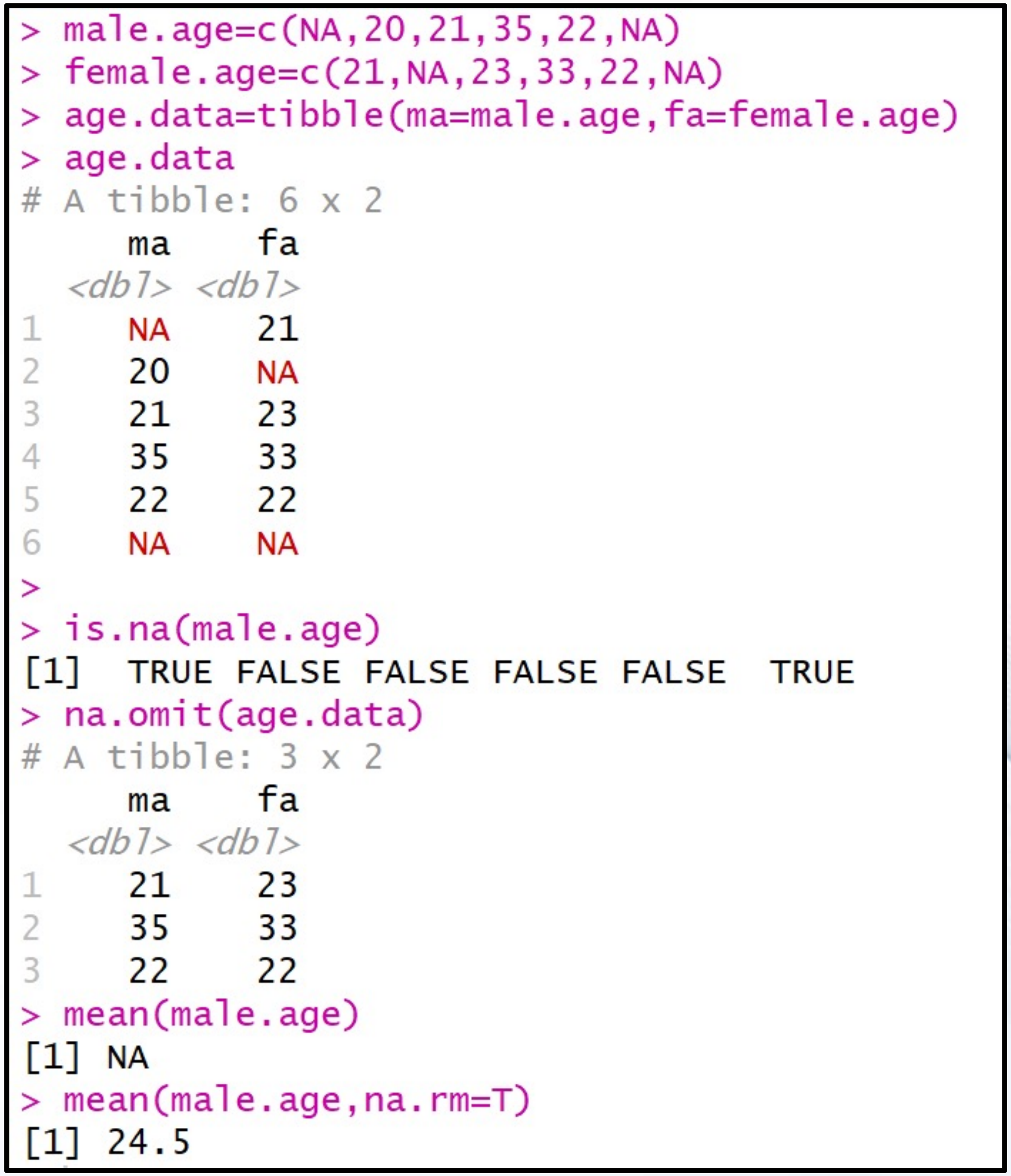
Missing Values

- Represented by NA
  - Enduring Questions
    - To Impute or Not Impute
    - To Ignore or Not Ignore
  - Handling Should Be Explained
  - Be Careful When Performing Operations on Missing Data



```
NA > 5
#> [1] NA
10 == NA
#> [1] NA
NA + 10
#> [1] NA
NA / 2
#> [1] NA
```

Missing Values



```
> male.age=c(NA,20,21,35,22,NA)
> female.age=c(21,NA,23,33,22,NA)
> age.data=tibble(ma=male.age,fa=female.age)
> age.data
# A tibble: 6 x 2
     ma    fa
  <dbl> <dbl>
1    NA    21
2    20    NA
3    21    23
4    35    33
5    22    22
6    NA    NA
>
> is.na(male.age)
[1]  TRUE FALSE FALSE FALSE FALSE  TRUE
> na.omit(age.data)
# A tibble: 3 x 2
     ma    fa
  <dbl> <dbl>
1    21    23
2    35    33
3    22    22
> mean(male.age)
[1] NA
> mean(male.age,na.rm=T)
[1] 24.5
```

filter()

- Used to Subset Observations Based on Their Values

  - Selects Row if TRUE
  - Removes Row if FALSE

- Examples:
  - All Flights from 9/13/2018 Out of LaGuardia Airport

```
>  filter(flights, month==9,day==13,origin ==“LGA”)
```

  - All Dec. or Nov. Flights

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

```
>  filter(flights, month %in% c(11,12))
```

filter()

- Examples:
  - Don’t Want Flights with Unusual Delays (> 120 min.)

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

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

!(x & y) = !x | !y

  - Want Flights with No Delays

```
>  filter(flights, dep_delay==0, arr_delay==0)
```

```
>  filter(flights, dep_delay==0 & arr_delay==0)
```

filter()

- Examples:
  - Want Flights Missing Air Time

```
>  filter(flights, is.na(air_time) )
```

  - Do not Want Flights Missing Air Time

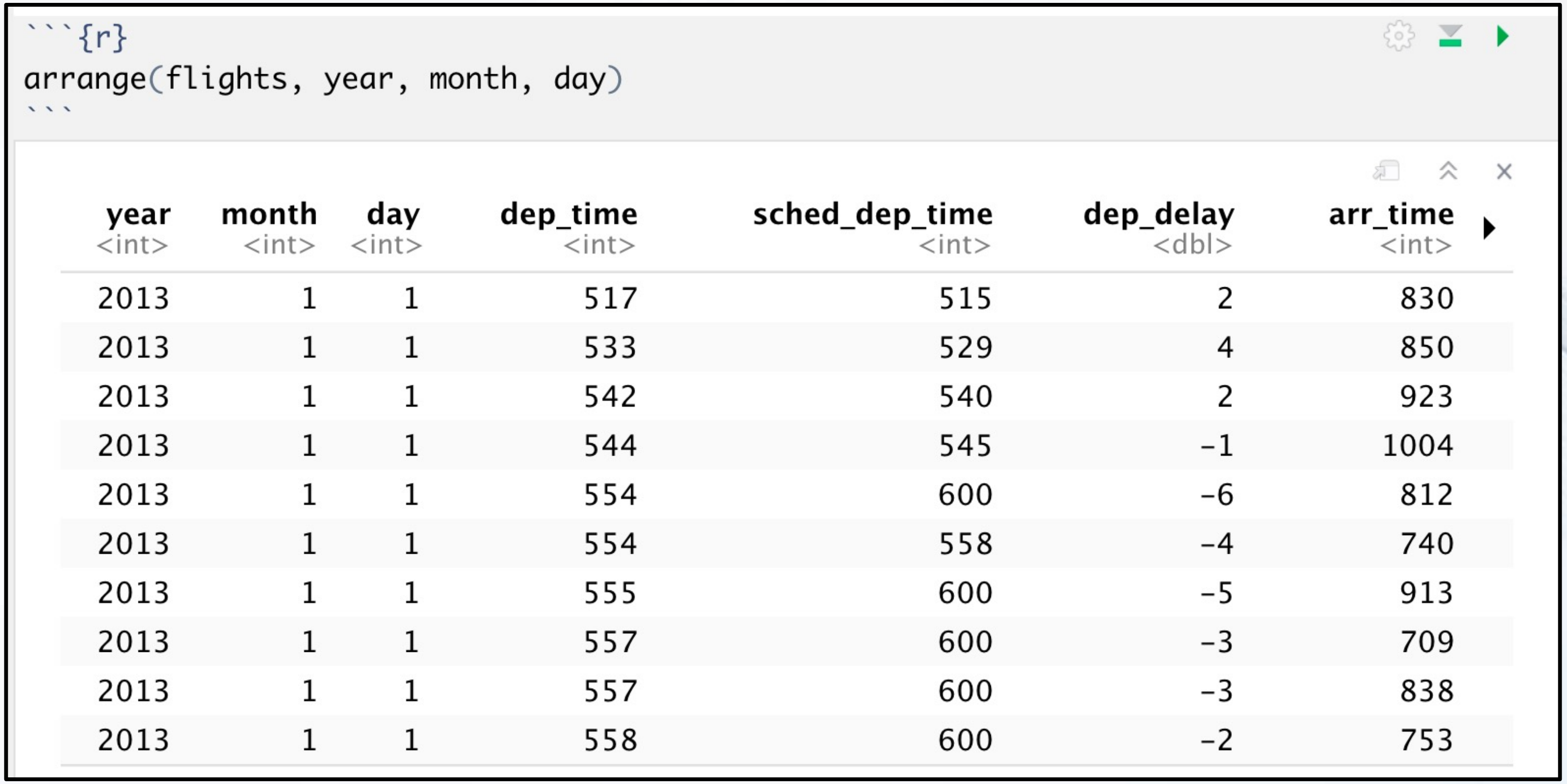
```
>  filter(flights, !is.na(air_time) )
```

  - Remove All Cases with Missing Values

```
>  na.omit(flights)
```

arrange()

- Used to Sort Observations
- Sort flights by date

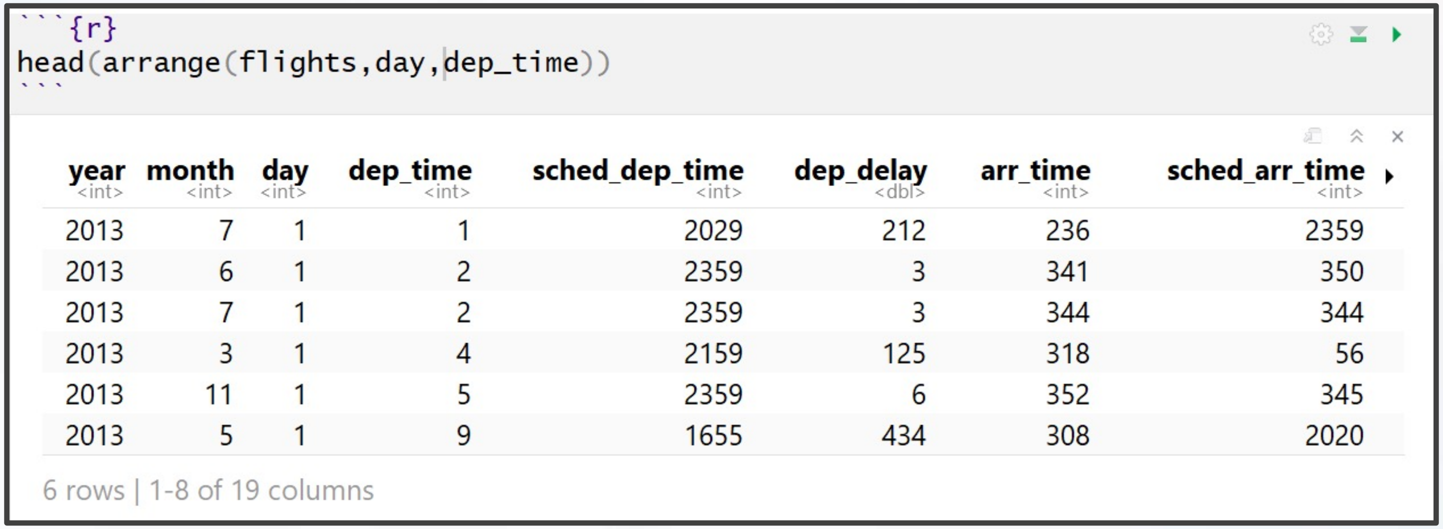


```{r}
arrange(flights, year, month, day)
```

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>
2013	1	1	517	515	2	830
2013	1	1	533	529	4	850
2013	1	1	542	540	2	923
2013	1	1	544	545	-1	1004
2013	1	1	554	600	-6	812
2013	1	1	554	558	-4	740
2013	1	1	555	600	-5	913
2013	1	1	557	600	-3	709
2013	1	1	557	600	-3	838
2013	1	1	558	600	-2	753

arrange()

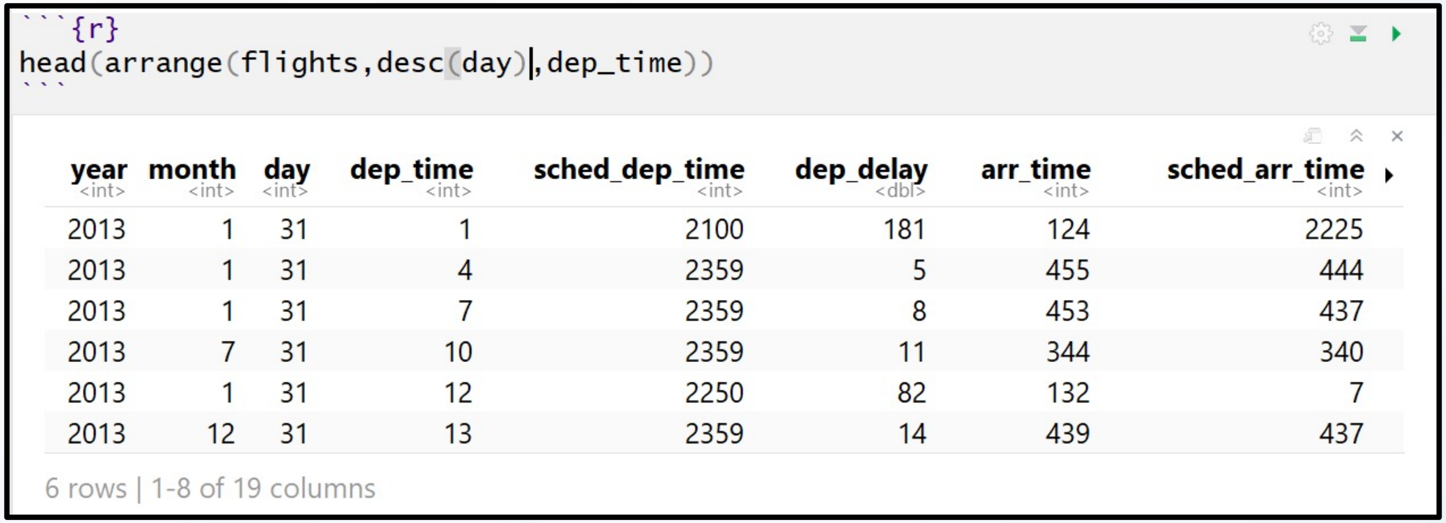
- Sorting Experiment



~~~
```{r}
head(arrange(flights,day,dep_time))
```
~~~

year <int>	month <int>	day <int>	dep_time <int>	sched_dep_time <int>	dep_delay <dbl>	arr_time <int>	sched_arr_time <int>
2013	7	1	1	2029	212	236	2359
2013	6	1	2	2359	3	341	350
2013	7	1	2	2359	3	344	344
2013	3	1	4	2159	125	318	56
2013	11	1	5	2359	6	352	345
2013	5	1	9	1655	434	308	2020

6 rows | 1-8 of 19 columns



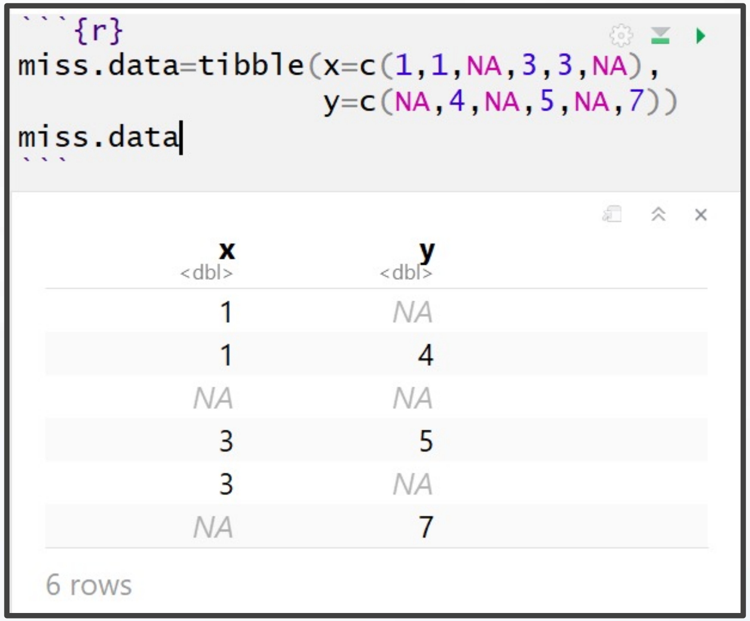
~~~
```{r}
head(arrange(flights,desc(day),dep_time))
```
~~~

year <int>	month <int>	day <int>	dep_time <int>	sched_dep_time <int>	dep_delay <dbl>	arr_time <int>	sched_arr_time <int>
2013	1	31	1	2100	181	124	2225
2013	1	31	4	2359	5	455	444
2013	1	31	7	2359	8	453	437
2013	7	31	10	2359	11	344	340
2013	1	31	12	2250	82	132	7
2013	12	31	13	2359	14	439	437

6 rows | 1-8 of 19 columns

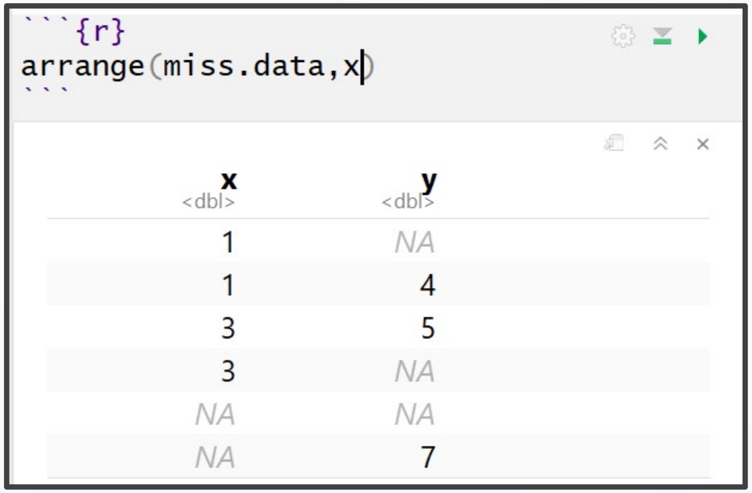
arrange()

- Handling NA



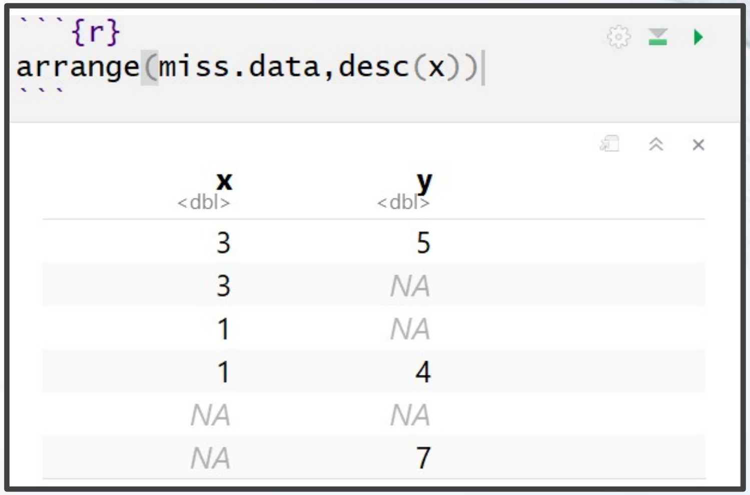
~~~
```{r}
miss.data=tibble(x=c(1,1,NA,3,3,NA),
                 y=c(NA,4,NA,5,NA,7))
miss.data
```
~~~

```
    x      y
<dbl>  <dbl>
    1     NA
    1      4
   NA     NA
    3      5
    3     NA
   NA      7
6 rows
```



~~~
```{r}
arrange(miss.data,x)
```
~~~

```
    x      y
<dbl>  <dbl>
    1     NA
    1      4
    3      5
    3     NA
   NA     NA
   NA      7
```

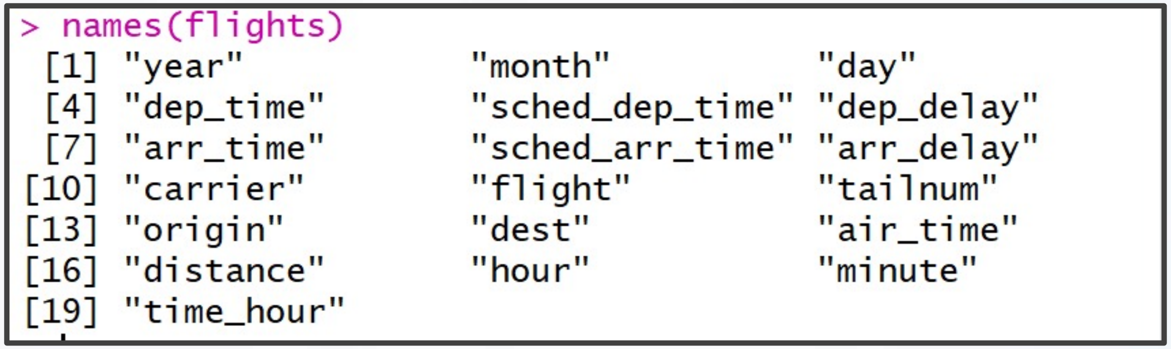


~~~
```{r}
arrange(miss.data,desc(x))
```
~~~

```
    x      y
<dbl>  <dbl>
    3      5
    3     NA
    1     NA
    1      4
   NA     NA
   NA      7
```

select()

- Used to Select Variables
- Why? Not All Variables are Created Equal
- Need to Know Variable Names

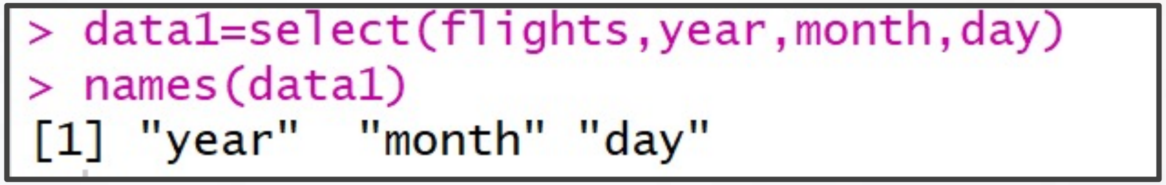


```
> names(flights)
 [1] "year"           "month"          "day"           
 [4] "dep_time"       "sched_dep_time" "dep_delay"     
 [7] "arr_time"       "sched_arr_time" "arr_delay"     
[10] "carrier"        "flight"         "tailnum"       
[13] "origin"         "dest"           "air_time"      
[16] "distance"       "hour"           "minute"        
[19] "time_hour"
```

select()

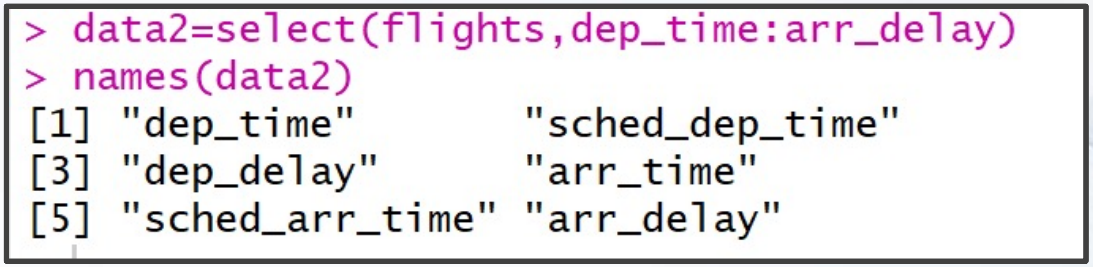
- Basic Examples

  - Select Only Year, Month, Day



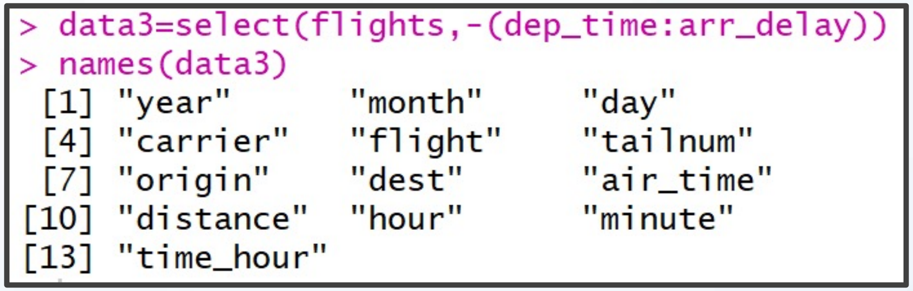
```
> data1=select(flights,year,month,day)
> names(data1)
[1] "year"  "month" "day"
```

  - Select All Variables Between dep_time to arr_delay



```
> data2=select(flights,dep_time:arr_delay)
> names(data2)
[1] "dep_time"       "sched_dep_time"
[3] "dep_delay"      "arr_time"
[5] "sched_arr_time" "arr_delay"
```

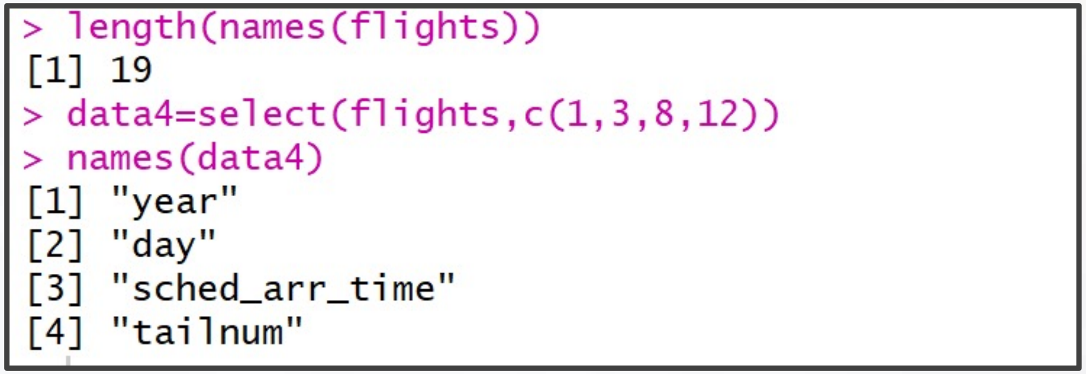
  - Deselect All Variables Between dep_time to arr_delay



```
> data3=select(flights,-(dep_time:arr_delay))
> names(data3)
 [1] "year"      "month"     "day"
 [4] "carrier"   "flight"    "tailnum"
 [7] "origin"    "dest"      "air_time"
[10] "distance"  "hour"      "minute"
[13] "time_hour"
```

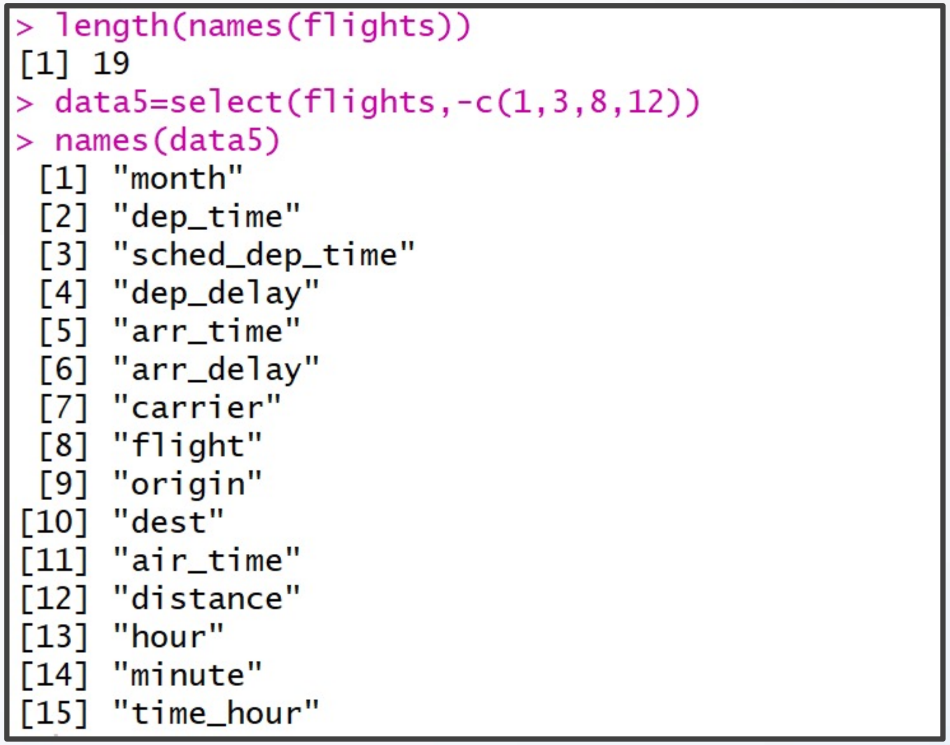
select()

- Select Based on Column Index



```
> length(names(flights))
[1] 19
> data4=select(flights,c(1,3,8,12))
> names(data4)
[1] "year"
[2] "day"
[3] "sched_arr_time"
[4] "tailnum"
```

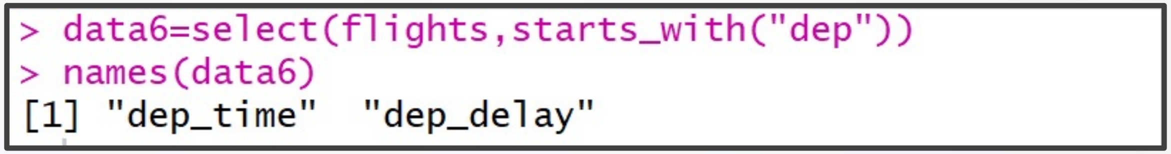
- Deselect Based on Column Index



```
> length(names(flights))
[1] 19
> data5=select(flights,-c(1,3,8,12))
> names(data5)
 [1] "month"
 [2] "dep_time"
 [3] "sched_dep_time"
 [4] "dep_delay"
 [5] "arr_time"
 [6] "arr_delay"
 [7] "carrier"
 [8] "flight"
 [9] "origin"
[10] "dest"
[11] "air_time"
[12] "distance"
[13] "hour"
[14] "minute"
[15] "time_hour"
```

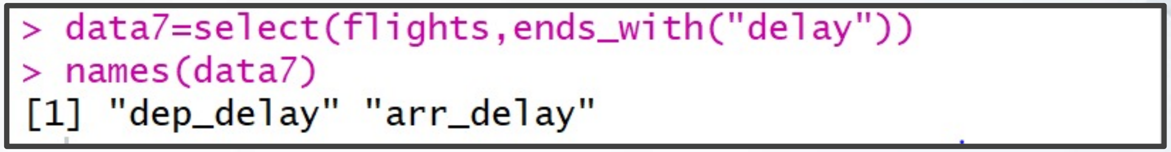
select()

- Select Based on Text
  - starts_with("TEXT")



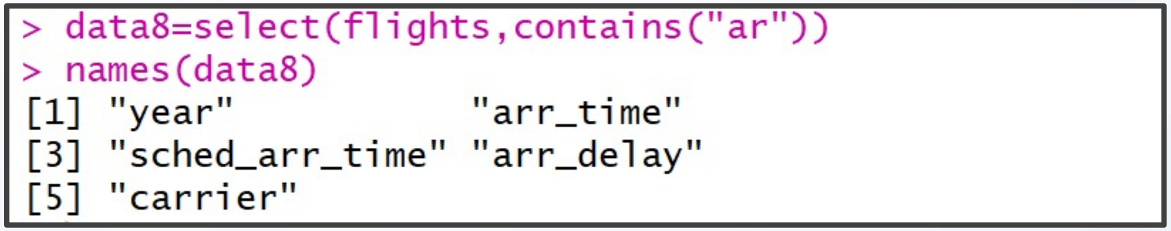
```
> data6=select(flights,starts_with("dep"))
> names(data6)
[1] "dep_time"  "dep_delay"
```

  - ends_with("TEXT")



```
> data7=select(flights,ends_with("delay"))
> names(data7)
[1] "dep_delay" "arr_delay"
```

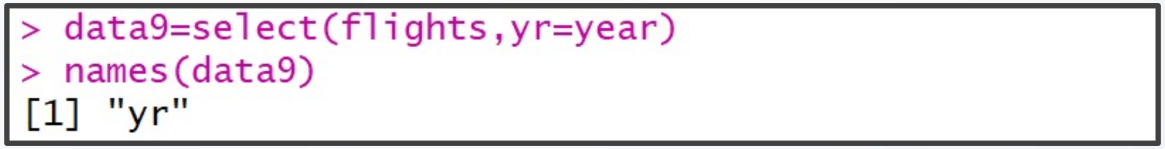
  - contains("TEXT")



```
> data8=select(flights,contains("ar"))
> names(data8)
[1] "year"           "arr_time"
[3] "sched_arr_time" "arr_delay"
[5] "carrier"
```

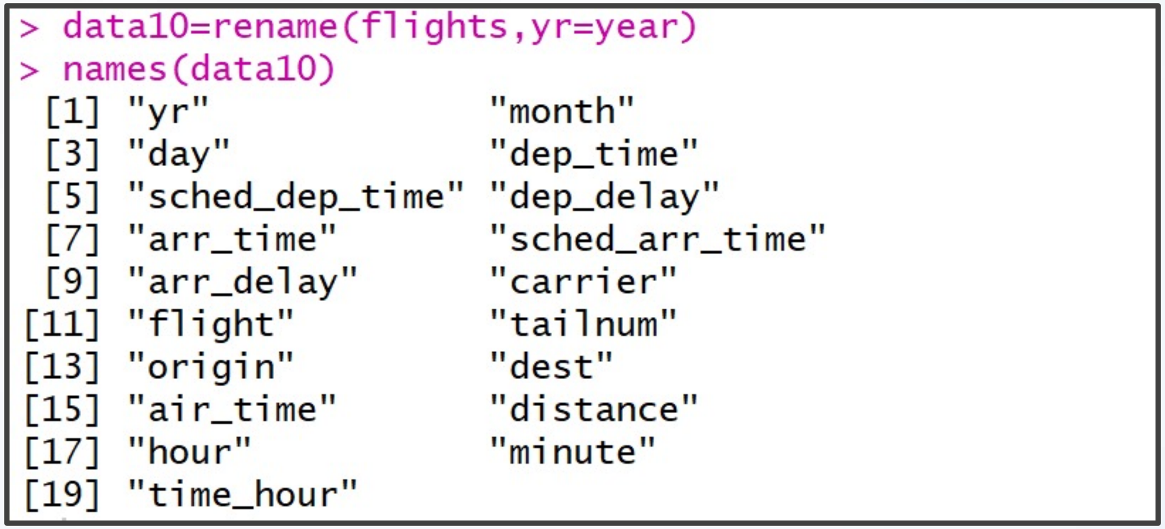
select()

- Renaming Variables
  - Can Use select()



```
> data9=select(flights,yr=year)
> names(data9)
[1] "yr"
```

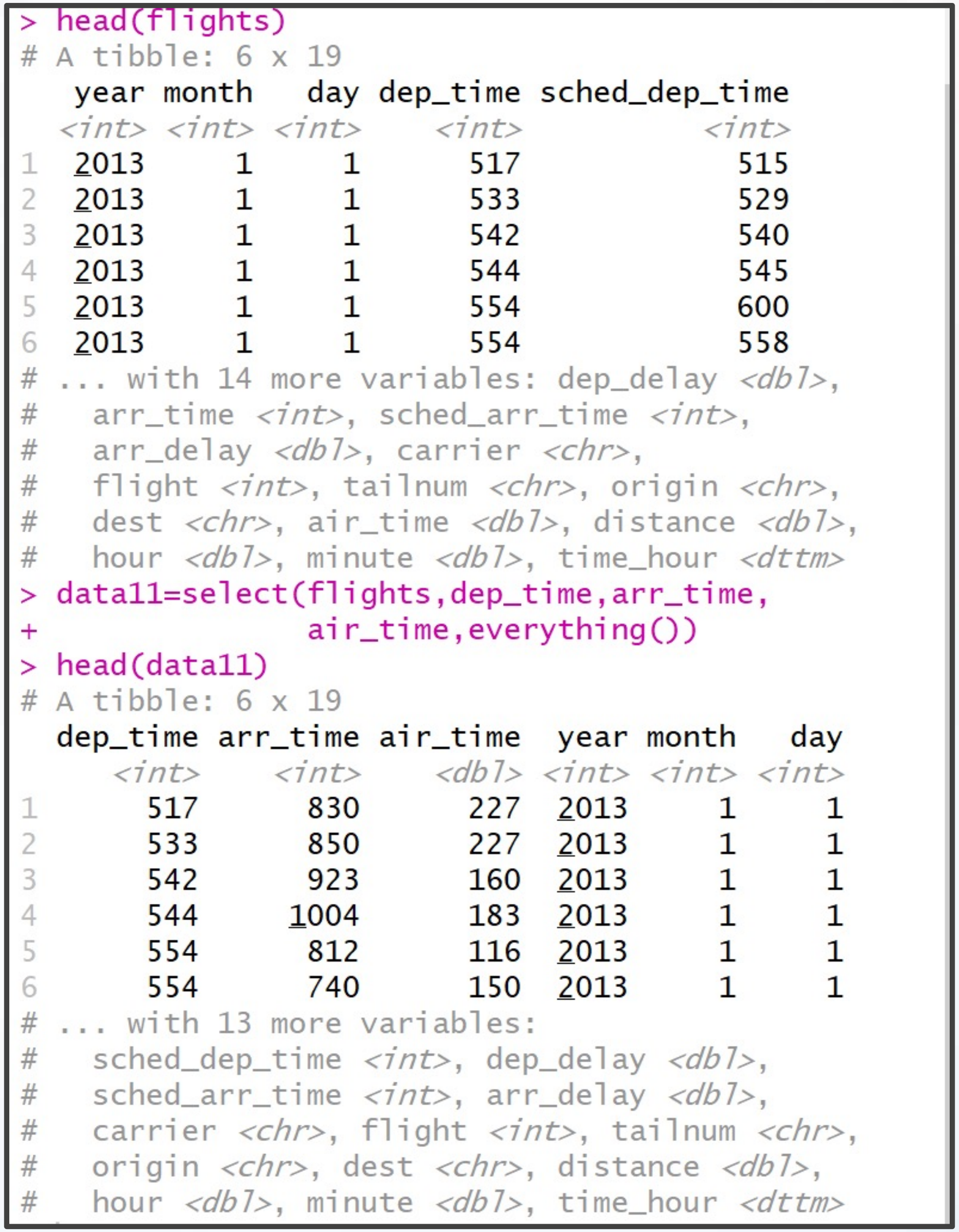
  - But Use rename()



```
> data10=rename(flights,yr=year)
> names(data10)
 [1] "yr"             "month"
 [3] "day"            "dep_time"
 [5] "sched_dep_time" "dep_delay"
 [7] "arr_time"       "sched_arr_time"
 [9] "arr_delay"      "carrier"
[11] "flight"         "tailnum"
[13] "origin"         "dest"
[15] "air_time"       "distance"
[17] "hour"           "minute"
[19] "time_hour"
```

select()

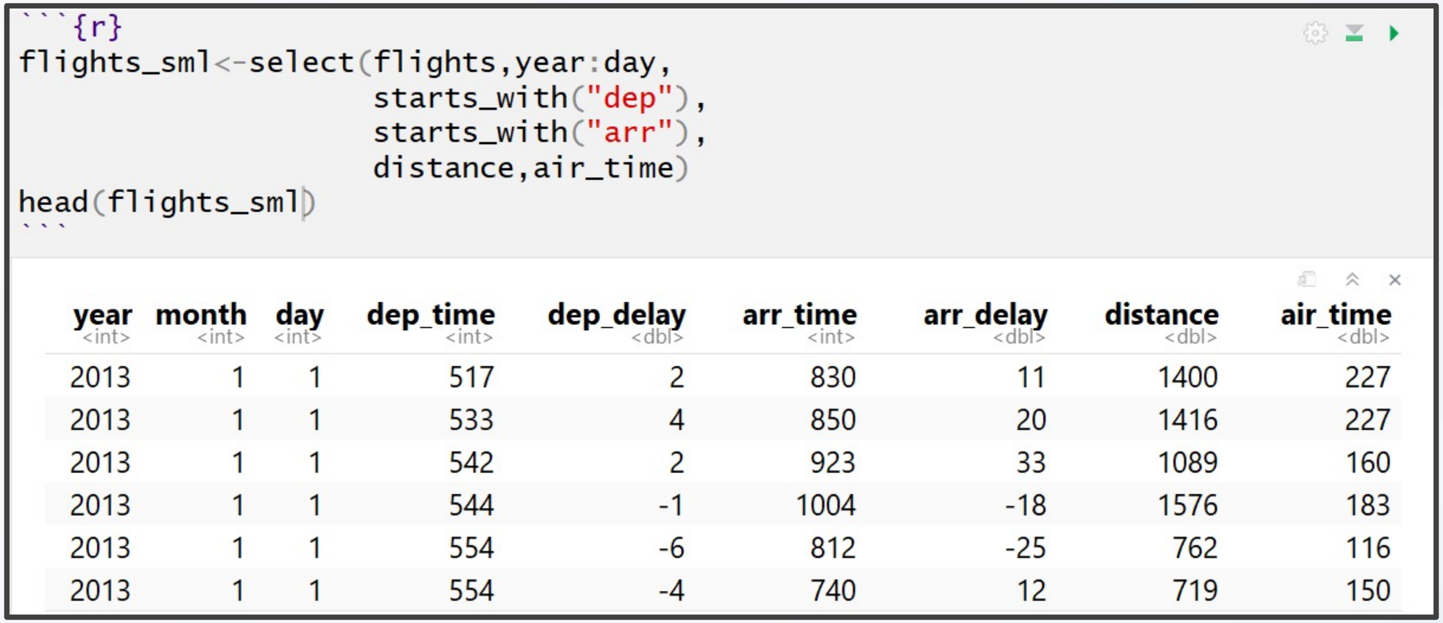
- Reordering Variables



```
> head(flights)
# A tibble: 6 x 19
   year month   day dep_time sched_dep_time
  <int> <int> <int>    <int>          <int>
1  2013     1     1      517            515
2  2013     1     1      533            529
3  2013     1     1      542            540
4  2013     1     1      544            545
5  2013     1     1      554            600
6  2013     1     1      554            558
# ... with 14 more variables: 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>
> data11=select(flights,dep_time,arr_time,
+               air_time,everything())
> head(data11)
# A tibble: 6 x 19
  dep_time arr_time air_time  year month   day
     <int>    <int>    <dbl> <int> <int> <int>
1      517      830      227  2013     1     1
2      533      850      227  2013     1     1
3      542      923      160  2013     1     1
4      544     1004      183  2013     1     1
5      554      812      116  2013     1     1
6      554      740      150  2013     1     1
# ... with 13 more variables:
#   sched_dep_time <int>, dep_delay <dbl>,
#   sched_arr_time <int>, arr_delay <dbl>,
#   carrier <chr>, flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

mutate()

- Used to Create New Variables
  - Creative New Metrics
  - Modify Units
  - Transform Variables
  - Unique Identifiers
  - Numeric to Categorical
  - Categorical to Numeric
- Reduced Dataset

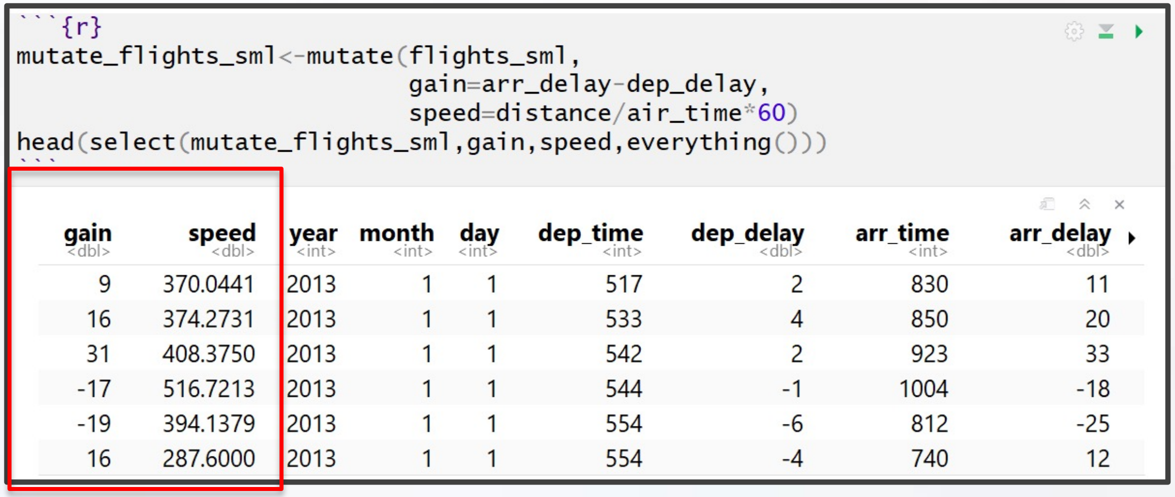


~~~
```{r}
flights_sml<-select(flights,year:day,
                    starts_with("dep"),
                    starts_with("arr"),
                    distance,air_time)
head(flights_sml)
```
~~~

year	month	day	dep_time	dep_delay	arr_time	arr_delay	distance	air_time
<int>	<int>	<int>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>
2013	1	1	517	2	830	11	1400	227
2013	1	1	533	4	850	20	1416	227
2013	1	1	542	2	923	33	1089	160
2013	1	1	544	-1	1004	-18	1576	183
2013	1	1	554	-6	812	-25	762	116
2013	1	1	554	-4	740	12	719	150

mutate()

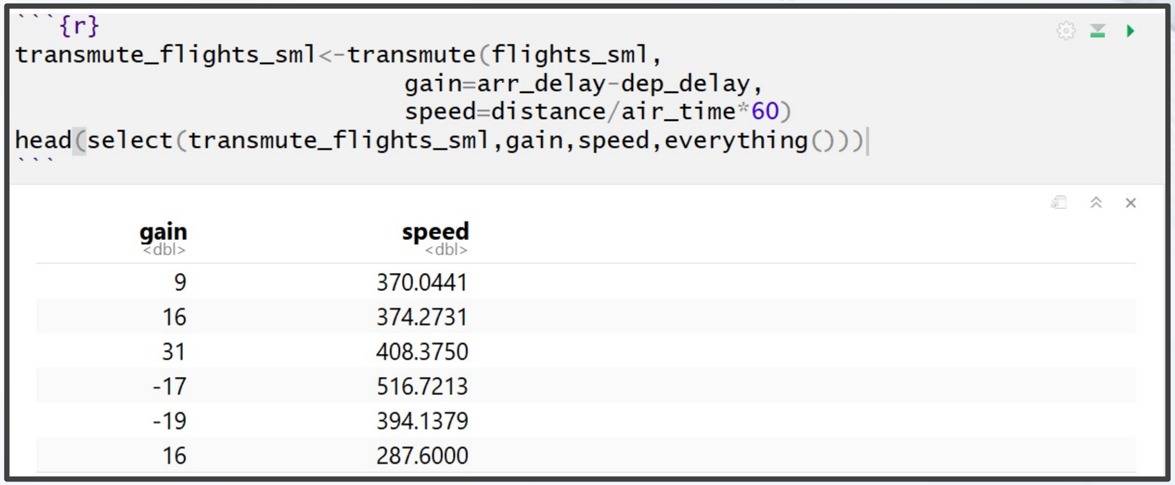
- Example of mutate()



~~~~
```{r}
mutate_flights_sml<-mutate(flights_sml,
                           gain=arr_delay-dep_delay,
                           speed=distance/air_time*60)
head(select(mutate_flights_sml,gain,speed,everything()))
```
~~~~

gain	speed	year	month	day	dep_time	dep_delay	arr_time	arr_delay
<dbl>	<dbl>	<int>	<int>	<int>	<int>	<dbl>	<int>	<dbl>
9	370.0441	2013	1	1	517	2	830	11
16	374.2731	2013	1	1	533	4	850	20
31	408.3750	2013	1	1	542	2	923	33
-17	516.7213	2013	1	1	544	-1	1004	-18
-19	394.1379	2013	1	1	554	-6	812	-25
16	287.6000	2013	1	1	554	-4	740	12

- Example of transmute()



~~~~
```{r}
transmute_flights_sml<-transmute(flights_sml,
                                 gain=arr_delay-dep_delay,
                                 speed=distance/air_time*60)
head(select(transmute_flights_sml,gain,speed,everything()))
```
~~~~

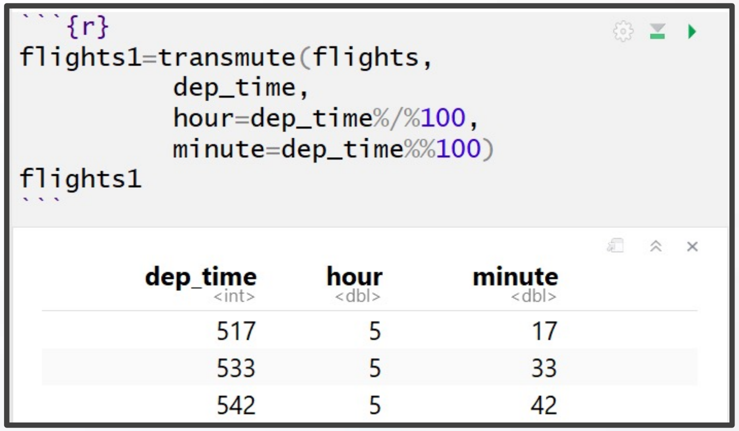
gain	speed
<dbl>	<dbl>
9	370.0441
16	374.2731
31	408.3750
-17	516.7213
-19	394.1379
16	287.6000

mutate()

- Plethora of Examples
  - Basic and Modular Arithmetic

dep_time, arr_time

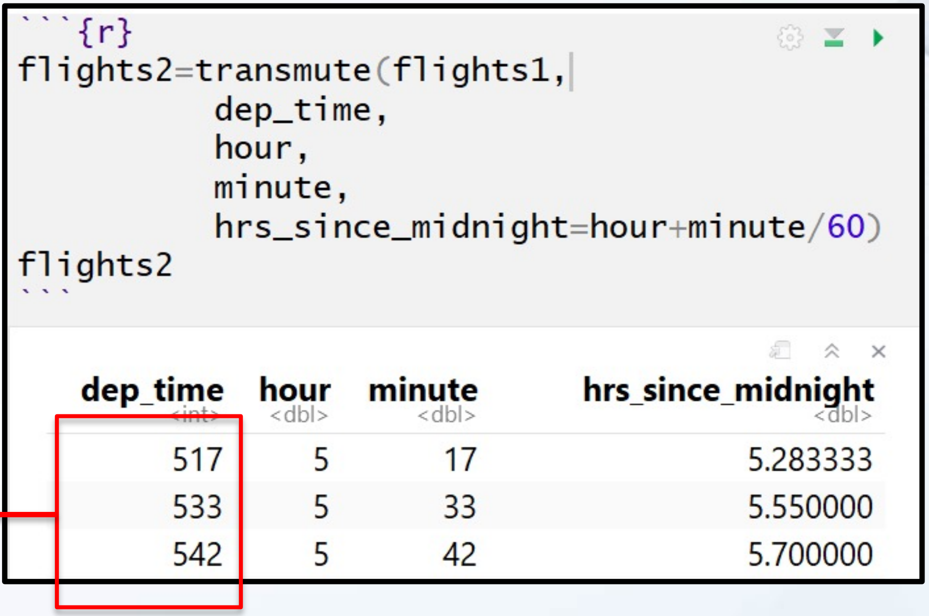
Actual departure and arrival times (format HHMM or HMM), local tz.



~~~
```{r}
flights1=transmute(flights,
          dep_time,
          hour=dep_time%/%100,
          minute=dep_time%%100)
flights1
```
~~~

```
 dep_time     hour     minute
    <int>    <dbl>      <dbl>
      517        5         17
      533        5         33
      542        5         42
```

$$\begin{aligned} 517 &= 100 * 5 + 17 \\ &= 100 * (517 \ \%/\% \ 100) + (517 \ \%\% \ 100) \end{aligned}$$



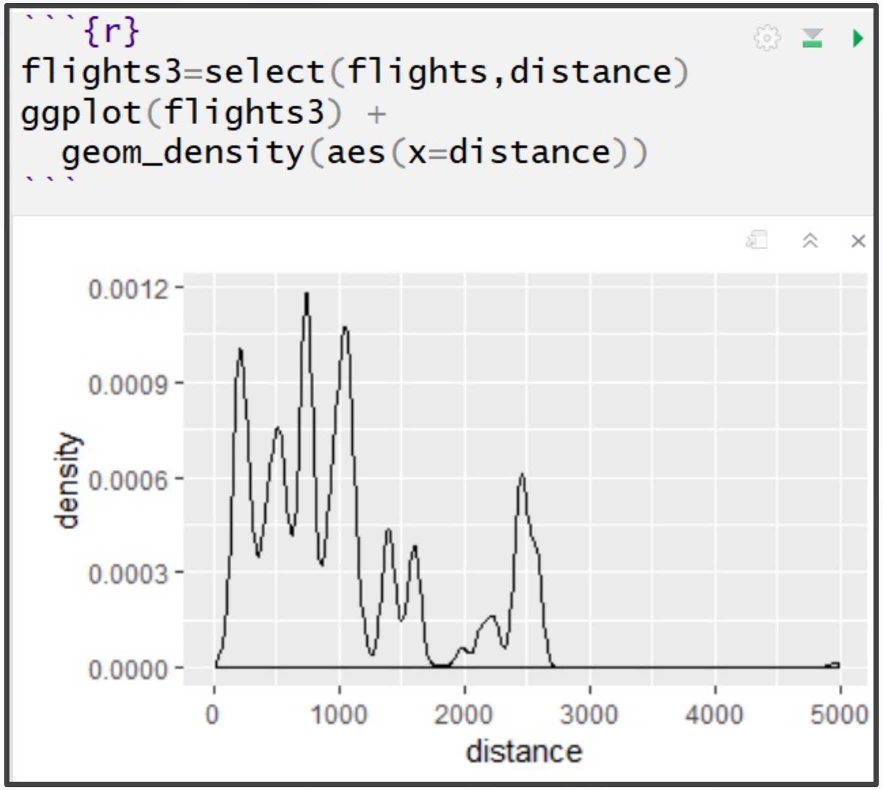
~~~
```{r}
flights2=transmute(flights1,
          dep_time,
          hour,
          minute,
          hrs_since_midnight=hour+minute/60)
flights2
```
~~~

```
 dep_time   hour   minute   hrs_since_midnight
    <int>  <dbl>    <dbl>                <dbl>
      517      5       17             5.283333
      533      5       33             5.550000
      542      5       42             5.700000
```

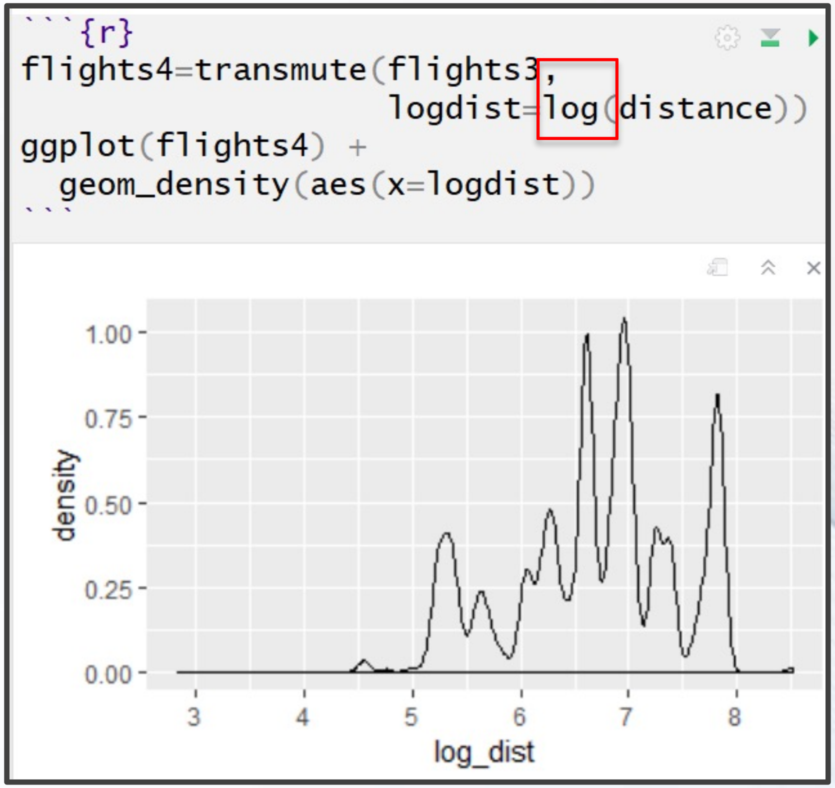
5:17AM
5:33AM
5:42AM

mutate()

- Plethora of Examples
  - Nonlinear Transformation



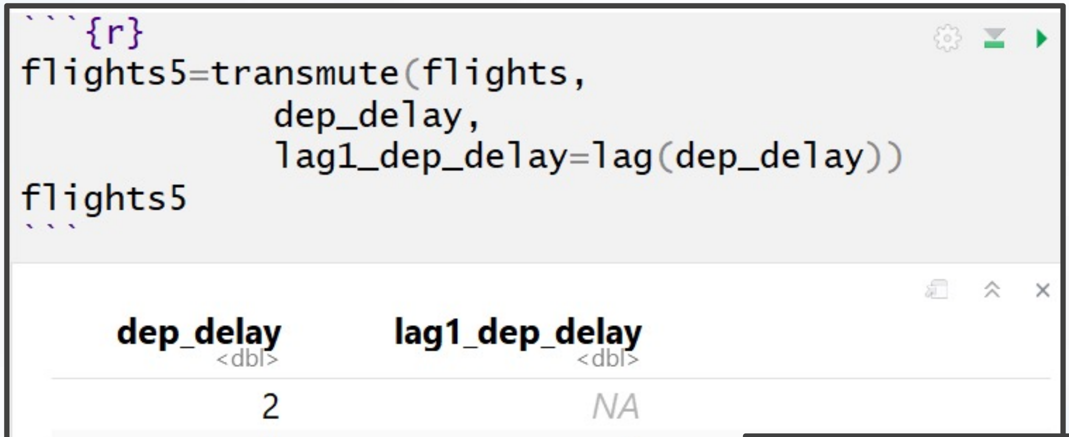
~~~
```{r}
flights3=select(flights,distance)
ggplot(flights3) +
  geom_density(aes(x=distance))
```
~~~



~~~
```{r}
flights4=transmute(flights3,
                   logdist=log(distance))
ggplot(flights4) +
  geom_density(aes(x=logdist))
```
~~~

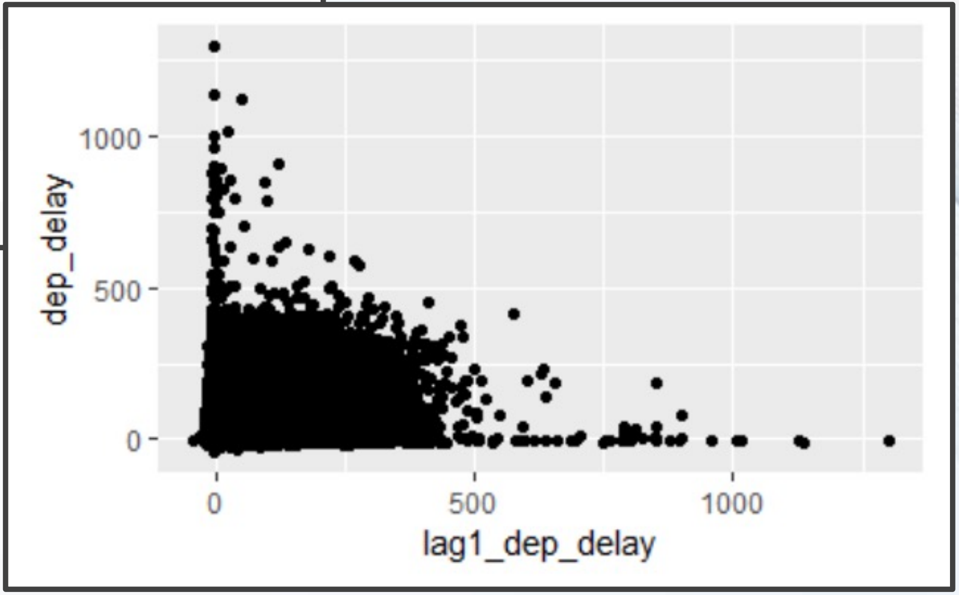
mutate()

- Plethora of Examples
  - Offsets: lead() and lag()



~~~
```{r}
flights5=transmute(flights,
              dep_delay,
              lag1_dep_delay=lag(dep_delay))
flights5
```
~~~

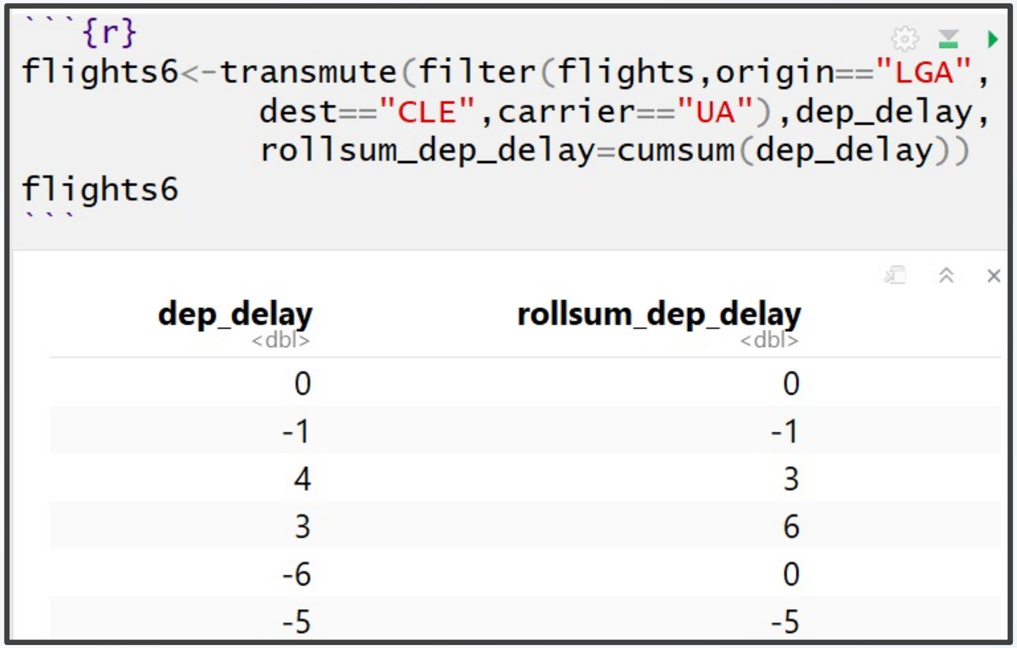
```
dep_delay    lag1_dep_delay
    <dbl>             <dbl>
        2                NA
        4                 2
        2                 4
       -1                 2
       -6                -1
       -4                -6
```



mutate()

- Plethora of Examples
  - Cumulative and Rolling Aggregates

- cumsum()
- cumprod()
- cummin()
- cummax()
- cummean()

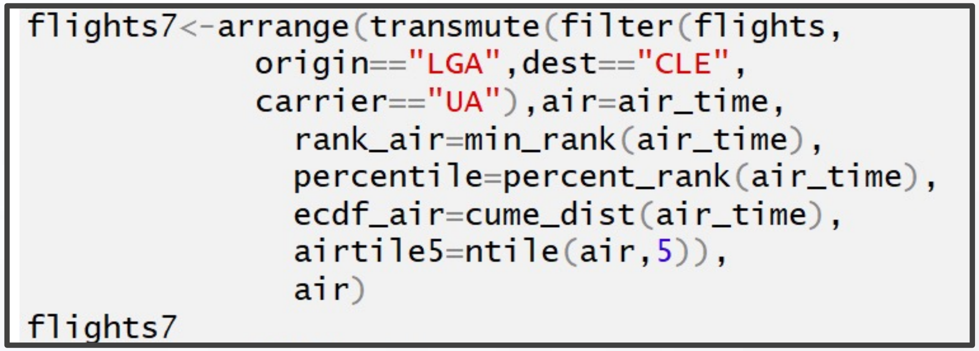


~~~~
```{r}
flights6<-transmute(filter(flights,origin=="LGA",
               dest=="CLE",carrier=="UA"),dep_delay,
               rollsum_dep_delay=cumsum(dep_delay))
flights6
```
~~~~

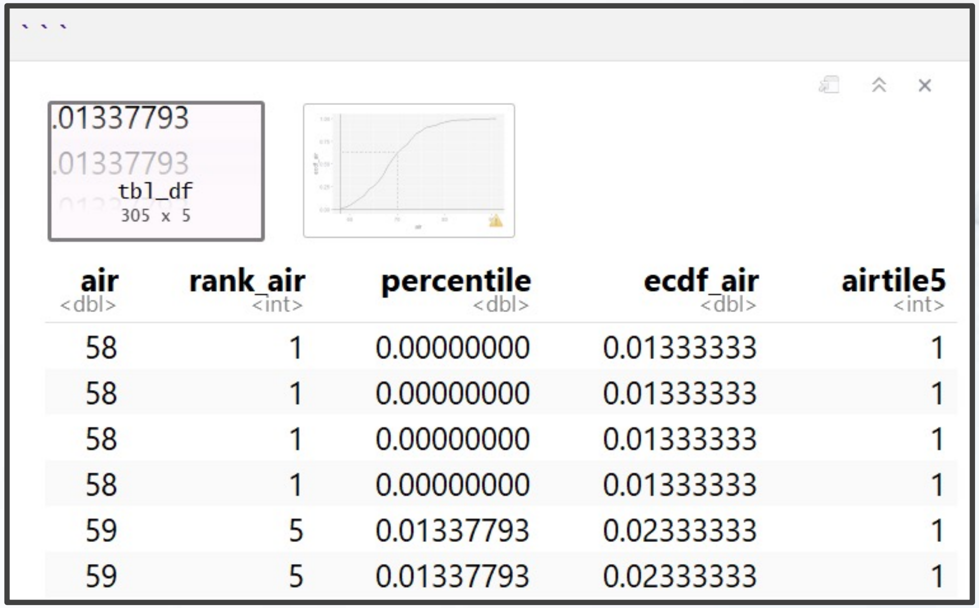
dep_delay	rollsum_dep_delay
<dbl>	<dbl>
0	0
-1	-1
4	3
3	6
-6	0
-5	-5

mutate()

- Plethora of Examples
  - Ranking
    - min_rank()
    - percent_rank()
    - cume_dist()
    - ntile()



```
flights7<-arrange(transmute(filter(flights,
           origin=="LGA",dest=="CLE",
           carrier=="UA"),air=air_time,
             rank_air=min_rank(air_time),
             percentile=percent_rank(air_time),
             ecdf_air=cume_dist(air_time),
             airtile5=ntile(air,5)),
             air)
flights7
```



```
```
```

```
.01337793
.01337793
   tbl_df
  305 x 5
```

```
 air   rank_air   percentile    ecdf_air   airtile5
<dbl>     <int>        <dbl>       <dbl>      <int>
  58          1   0.00000000  0.01333333          1
  58          1   0.00000000  0.01333333          1
  58          1   0.00000000  0.01333333          1
  58          1   0.00000000  0.01333333          1
  59          5   0.01337793  0.02333333          1
  59          5   0.01337793  0.02333333          1
```
```

Information

- Tutorial 3
  - Practice
    - filter()
    - arrange()
    - select()
    - mutate()
  - Introduced
    - Piping %>%
    - group_by()
    - summarize()