



Data Transformation I

Introduction



- Read Chapter 3
- 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 Meta Data



- Preview Data: > flights

```
> 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 <dtm>
```

- Four Different Types of Variables
 - int = integer
 - dbl = double
 - chr = character
 - dtm = date and times
- Other Types of Variables
 - lgl = logical (TRUE or FALSE)
 - fctr = factor
 - date = dates

Basics of dplyr



- 5 Key Functions
 - `filter()` = Chooses Observations Based on Values
 - `arrange()` = Sorts Observations
 - `select()` = Chooses Variables
 - `mutate()` = Creates New Variables
 - `summarize()` = 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
- Powerful When Used With `group_by()` Function

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
```

Comparisons



- 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

```
> x=T
> y=F
> #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

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 9/13/2013 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)
```

- 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
- Raw Flight Data

```
{r}  
head(flights)
```

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>
2013	1	1	517	515	2	830	819
2013	1	1	533	529	4	850	830
2013	1	1	542	540	2	923	850
2013	1	1	544	545	-1	1004	1022
2013	1	1	554	600	-6	812	837
2013	1	1	554	558	-4	740	728

6 rows | 1-8 of 19 columns

- Sorted by year, month, and day,
in that order

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

```
> order(miss.data$x)
[1] 1 2 4 5 3 6
> order(desc(miss.data$x))
[1] 4 5 1 2 3 6
> is.na(miss.data$x)
[1] FALSE FALSE TRUE FALSE FALSE
[6] TRUE
> order(is.na(miss.data$x))
[1] 1 2 4 5 3 6
> order(is.na(miss.data$x),
+       decreasing=T)
[1] 3 6 1 2 4 5
```

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 Year, Month, and Day

```
> 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"
```

- Etc. AKA Others Exist

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 <dtm>
> 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 <dtm>
```

Closing



Disperse
and Make
Reasonable
Decisions