



STOR 320 Data Transformation II

Lecture 5

Yao Li

Department of Statistics and Operations Research

UNC Chapel Hill



Data Transformation II Info

- Finish Reading Chapter 5 and Practice the Code in R4DS
- Covers
 - The Pipe
 - Statistical Summaries
 - Grouped Summaries
 - Helpful Functions
- Builds Off Last Tutorial



The Pipe

- Useful for Combining Multiple Steps of Operations
- Represented by `%>%`
- Reads as “Then”
- Works Like a Composite Function From Algebra

$$f(x) = 3x + 4$$

$$g(x) = 2x$$

$$h = 1$$

$$f(g(h)) = 3(2(1)) + 4 = 10$$



OUT = h %>%

g() %>%

f()

OUT = 10

The Pipe

- Chaining with the Pipe



```
```{r,eval=F}
f2e.pipedream =
Acknowledge the Original Data
flights %>%
Input Original Data and Perform Mutations
transmute(dep_hr=dep_time%%100+(dep_time%%100)/60,
 sched_dep_hr=sched_dep_time%%100+(sched_dep_time%%100)/60,
 arr_hr=arr_time%%100+(arr_time%%100)/60,
 sched_arr_hr=sched_arr_time%%100+(sched_arr_time%%100)/60) %>%

mutate(dep_delay_hr=dep_hr-sched_dep_hr,
 arr_delay_hr=arr_hr-sched_arr_hr) %>%

mutate(percent_dep_delay_hr=percent_rank(dep_delay_hr)) %>%
Input Modified Data and Filter the observations
filter(percent_dep_delay_hr<0.1|percent_dep_delay_hr>0.9) %>%

Input Modified Data and Sort according to percent_dep_delay_hr
arrange(desc(percent_dep_delay_hr))
```

dep_hr <dbl>	sched_dep_hr <dbl>	arr_hr <dbl>	sched_arr_hr <dbl>	dep_delay_hr <dbl>	arr_delay_hr <dbl>
23.35000	8.166667	1.583333	10.33333	15.18333	-8.750000
22.95000	7.983333	1.350000	10.43333	14.96667	-9.083333
22.71667	8.500000	1.000000	11.10000	14.21667	-10.100000
23.40000	10.266667	1.233333	12.45000	13.13333	-11.216667
19.35000	6.250000	21.583333	8.70000	13.10000	12.883333

5 rows | 1-6 of 7 columns



# The Pipe

## Why use

- Avoid nested functions
- Minimize number of local variables
- Easier to add steps in the sequence

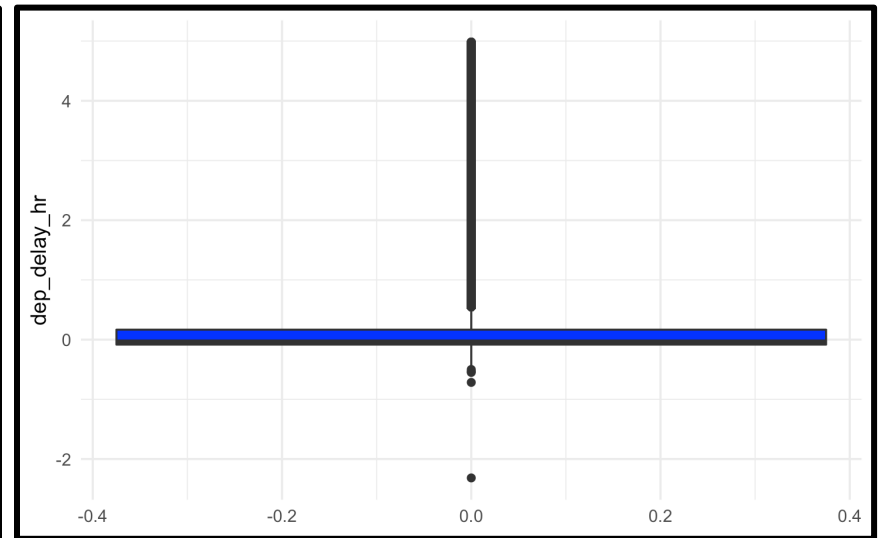
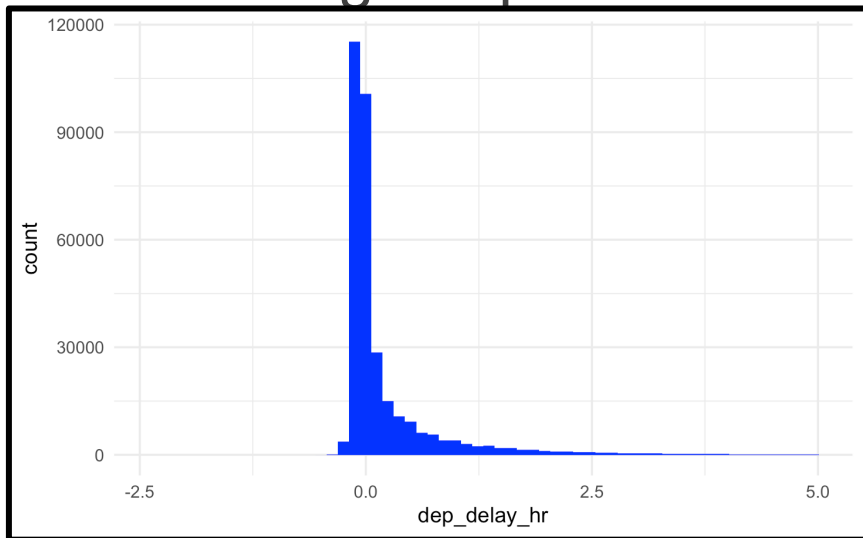
## Why not to use

- Debug
- Can't handle multiple inputs
- Can't handle complex code structure



# summarize()

- Summarizing All Data
- Using Graphics



Both the histogram and the boxplot are made from summary statistics.

**(Statistical Transformations in Ch. 3)**



# summarize()

- Summarizing All Data

```
```{r}
dep_delay_hr.summary1 = summarize(f3e.pipedream,
                                  n=n(),
                                  mean=mean(dep_delay_hr,na.rm=T),
                                  var=var(dep_delay_hr,na.rm=T),
                                  sd=sd(dep_delay_hr,na.rm=T))

dep_delay_hr.summary2 =
  f3e.pipedream %>%
  summarize(n=n(),
            min=min(dep_delay_hr),
            Q1=quantile(dep_delay_hr,0.25),
            Q2=quantile(dep_delay_hr,0.5),
            Q3=quantile(dep_delay_hr,0.75),
            max=max(dep_delay_hr),
            IQR=Q3-Q1)
```
```

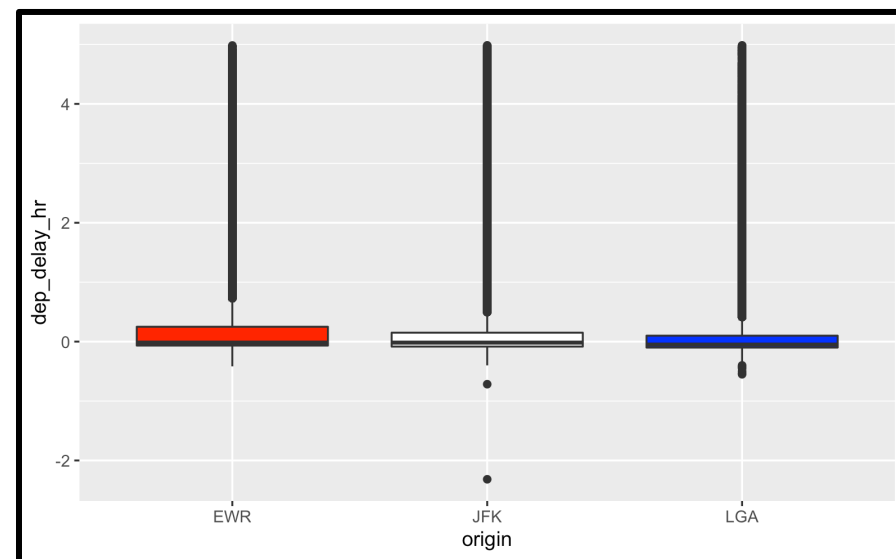
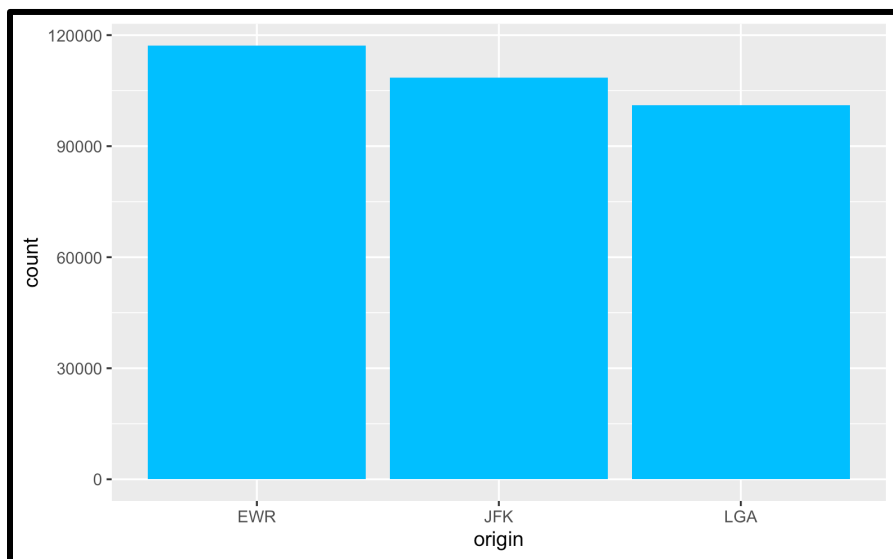
| n      | mean     | var       | sd        |
|--------|----------|-----------|-----------|
| 326848 | 0.192752 | 0.3506166 | 0.5921289 |

| n      | min       | Q1         | Q2         | Q3        | max      | IQR  |
|--------|-----------|------------|------------|-----------|----------|------|
| 326848 | -2.316667 | -0.0833333 | -0.0333333 | 0.1666667 | 4.983333 | 0.25 |



# summarize() with group\_by()

- Summarizing Data by Groups
  - Using Graphics







# summarize() with group\_by()

- Summarizing Data by Groups
  - Using Tables

```
group.summary1 = f3e.pipedream %>%
 group_by(origin) %>%
 summarize(n=n())

group.summary2 =
 f3e.pipedream %>%
 group_by(origin) %>%
 summarize(n=n(),
 min=min(dep_delay_hr),
 Q1=quantile(dep_delay_hr,0.25),
 Q2=quantile(dep_delay_hr,0.5),
 Q3=quantile(dep_delay_hr,0.75),
 max=max(dep_delay_hr),
 IQR=Q3-Q1,
 nLow=sum(dep_delay_hr<Q1-1.5*IQR),
 propHigh=mean(dep_delay_hr>Q3+1.5*IQR))
```

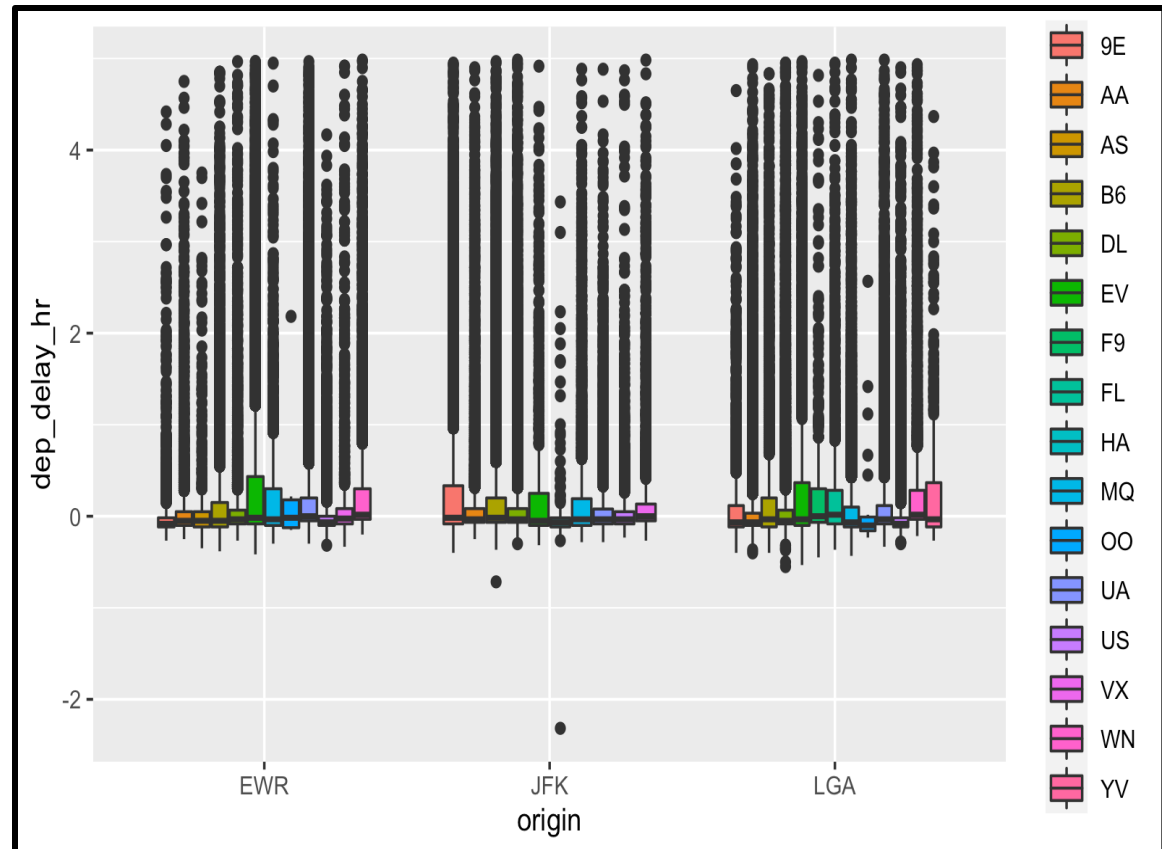
| origin | n      |
|--------|--------|
| EWR    | 117209 |
| JFK    | 108486 |
| LGA    | 101153 |

| origin | n      | min        | Q1          | Q2          | Q3    | max      | IQR       | nL... | propHigh  |
|--------|--------|------------|-------------|-------------|-------|----------|-----------|-------|-----------|
| <chr>  | <int>  | <dbl>      | <dbl>       | <dbl>       | <dbl> | <dbl>    | <dbl>     | <int> | <dbl>     |
| EWR    | 117209 | -0.4166667 | -0.06666667 | -0.01666667 | 0.25  | 4.983333 | 0.3166667 | 0     | 0.1259204 |
| JFK    | 108486 | -2.3166667 | -0.08333333 | -0.01666667 | 0.15  | 4.983333 | 0.2333333 | 2     | 0.1372988 |
| LGA    | 101153 | -0.5500000 | -0.10000000 | -0.05000000 | 0.10  | 4.983333 | 0.2000000 | 7     | 0.1466491 |



# summarize() with group\_by()

- Multiple Groups
  - Using Graphics





# summarize() with group\_by()

- Multiple Groups
  - Using Tables

| origin | carrier | n     | min        | Q1          | Q2          | Q3          | max      |
|--------|---------|-------|------------|-------------|-------------|-------------|----------|
| EWR    | 9E      | 1199  | -0.2666667 | -0.1166667  | -0.08333333 | -0.01666667 | 4.416667 |
| EWR    | AA      | 3376  | -0.2500000 | -0.1000000  | -0.0500000  | 0.0500000   | 4.750000 |
| EWR    | AS      | 712   | -0.3500000 | -0.1166667  | -0.0500000  | 0.0500000   | 3.750000 |
| EWR    | B6      | 6446  | -0.3833333 | -0.1166667  | -0.0500000  | 0.1500000   | 4.850000 |
| EWR    | DL      | 4281  | -0.2666667 | -0.08333333 | -0.03333333 | 0.0666667   | 4.966667 |
| EWR    | EV      | 41592 | -0.4166667 | -0.08333333 | -0.01666667 | 0.43333333  | 4.966667 |
| EWR    | MQ      | 2095  | -0.3000000 | -0.1000000  | -0.03333333 | 0.3000000   | 4.950000 |
| EWR    | OO      | 6     | -0.1500000 | -0.1250000  | -0.01666667 | 0.1791667   | 2.183333 |
| EWR    | UA      | 45561 | -0.3000000 | -0.0500000  | 0.0000000   | 0.2000000   | 4.966667 |
| EWR    | US      | 4326  | -0.3166667 | -0.1000000  | -0.06666667 | 0.0000000   | 4.166667 |
| EWR    | VX      | 1554  | -0.3333333 | -0.08333333 | -0.0250000  | 0.08333333  | 4.916667 |
| EWR    | WN      | 6061  | -0.2000000 | -0.03333333 | 0.01666667  | 0.3000000   | 4.983333 |
| JFK    | 9E      | 13801 | -0.4000000 | -0.08333333 | -0.01666667 | 0.33333333  | 4.950000 |
| JFK    | AA      | 13617 | -0.2500000 | -0.06666667 | -0.03333333 | 0.08333333  | 4.900000 |
| JFK    | B6      | 41005 | -0.7166667 | -0.06666667 | -0.01666667 | 0.2000000   | 4.966667 |
| JFK    | DL      | 20551 | -0.3000000 | -0.06666667 | -0.03333333 | 0.08333333  | 4.983333 |
| JFK    | EV      | 1315  | -0.3166667 | -0.1000000  | -0.0500000  | 0.2500000   | 4.916667 |

```
{r}
group.summary3 =
 f3e.ninedream %>%
 group_by(origin, carrier) %>%
 summarize(n=n(),
 min=min(dep_delay_hr),
 Q1=quantile(dep_delay_hr,0.25),
 Q2=quantile(dep_delay_hr,0.5),
 Q3=quantile(dep_delay_hr,0.75),
 max=max(dep_delay_hr),
 IQR=Q3-Q1,
 nLow=sum(dep_delay_hr<Q1-1.5*IQR),
 propHigh=mean(dep_delay_hr>Q3+1.5*IQR))
```



# Useful Summary Functions

- Measures of Center
  - `mean()`
  - `median()`
  - `mode()`
- Measures of Spread
  - `var()`
  - `sd()`
  - `IQR()`
  - `mad()`
- Measures of Rank
  - `min()`
  - `max()`
  - `quantile()`



# Useful Summary Functions

- Measures of Position
  - Order Matters
  - `first() = x[1]`
  - `last() = x[length(x)]`
  - `nth(,k) = x[k]`
- Counts
  - `n()`
  - `n_distinct()`
- Counts/Proportions for Logical
  - `sum()`
  - `mean()`
  - Example
    - `sum(x>10)`
    - `mean(x>10)`



# Case Study

- Flight Accuracy
  - Accurate Flight Means
    - Departure Delay = 0
    - Arrival Delay = 0

- Bad Metric

$$Accuracy = delay_{dep} + delay_{arr}$$

$$Accuracy = (delay_{dep} + delay_{arr})/2$$

- Good Metrics

$$Accuracy = |delay_{dep}| + |delay_{arr}|$$

$$Accuracy = \sqrt{delay_{dep}^2 + delay_{arr}^2}$$



# Case Study

- Summary Table
  - Step 1: Accuracy Variable
  - Step 2: Grouping
  - Step 3: Summarize Info
    - Mean
    - Standard Error
    - Lower Bound (95% CI)
    - Upper Bound (95% CI)

```
```{r}
accuracy<-
  f.pipedream3 %>%
  transmute(carrier,origin,
            accuracy=abs(dep_delay_hr)+abs(arr_delay_hr)) %>%
  group_by(carrier,origin) %>%
  summarize(n=n(),
            avg=mean(accuracy,na.rm=T),
            se=sd(accuracy,na.rm=T)/sqrt(n),
            low=avg-2*se,
            high=avg+2*se
  )
```
```



# Case Study

- Sorted by Average Accuracy
  - Best Carriers/Origin

```
> head(arrange(accuracy, avg), 5)
A tibble: 5 x 7
Groups: carrier [3]
 carrier origin n avg se low high
 <chr> <chr> <int> <dbl> <dbl> <dbl> <dbl>
1 US EWR 4322 0.505 0.0123 0.481 0.530
2 US JFK 2960 0.509 0.0152 0.479 0.539
3 US LGA 12517 0.544 0.0121 0.520 0.569
4 HA JFK 342 0.556 0.0362 0.483 0.628
5 UA JFK 4367 0.591 0.0173 0.556 0.625
```

- Worst Carriers/Origin

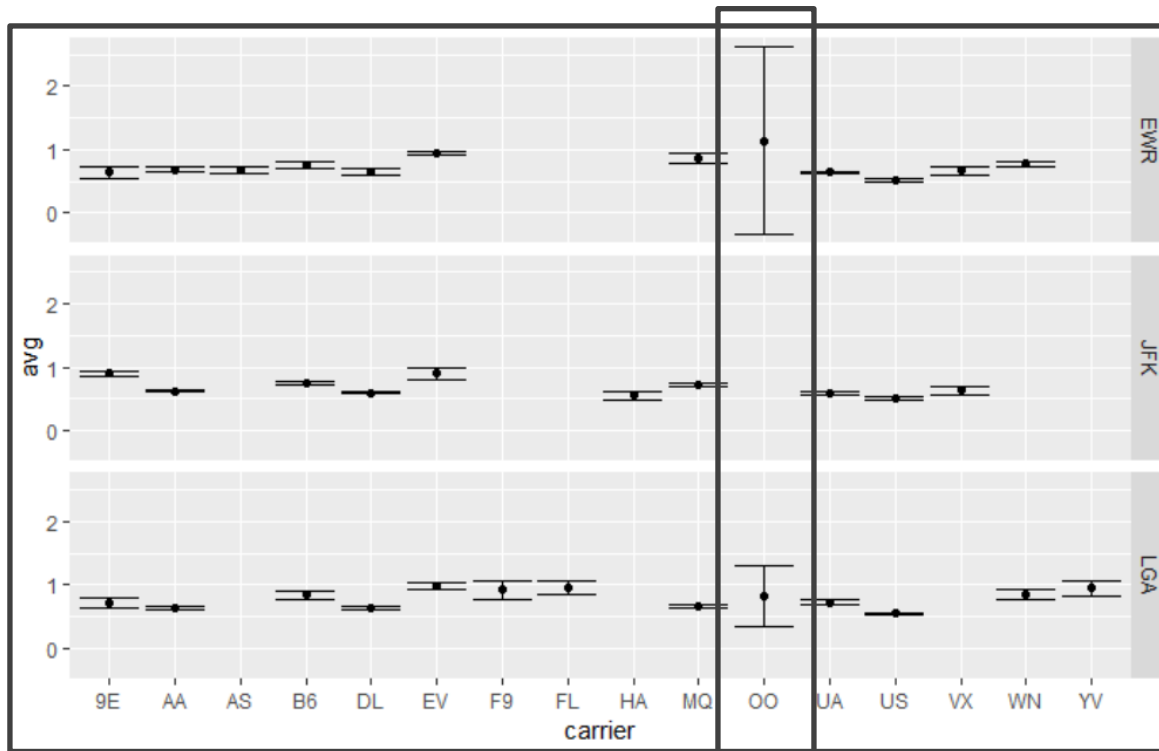
```
> head(arrange(accuracy, desc(avg)), 5)
A tibble: 5 x 7
Groups: carrier [4]
 carrier origin n avg se low high
 <chr> <chr> <int> <dbl> <dbl> <dbl> <dbl>
1 OO EWR 6 1.14 0.737 -0.334 2.61
2 EV LGA 8086 0.986 0.0265 0.933 1.04
3 YV LGA 542 0.954 0.0597 0.835 1.07
4 FL LGA 3136 0.952 0.0545 0.843 1.06
5 EV EWR 40571 0.952 0.0125 0.927 0.977
```





# Case Study

- 95% Confidence Intervals



Carrier "OO" Creates a Visual Problem Due to Small Sample Size



# Case Study

```
```{r}
ggplot(filter(accuracy, carrier!="OO")) +
  geom_point(aes(x=carrier, y=avg)) +
  geom_errorbar(aes(x=carrier, ymin=low, ymax=high)) +
  facet_grid(origin~.)
```
```

- 95% Confidence Intervals

