DATA 100,Week 3 (B)

relocate()

Move columns around, with further details using parameters .before and .after.

• The default behaviour is to move the columns to the front (leftmost) of the table

```
flights |>
relocate( year:day, distance, air_time,
# .after = carrier,
.before = dep_time )
```

```
#> # A tibble: 336,776 x 19
#>    year month   day distance air_time dep_time sched_dep_time dep_delay
#>   <int> <int> <int>    <dbl>    <dbl>    <int>          <int>     <dbl>
#> 1  2013     1     1     1400      227      517            515         2
#> 2  2013     1     1     1416      227      533            529         4
#> 3  2013     1     1     1089      160      542            540         2
#> 4  2013     1     1     1576      183      544            545        -1
#> 5  2013     1     1      762      116      554            600        -6
#> 6  2013     1     1      719      150      554            558        -4
#> # i 336,770 more rows
#> # i 11 more variables: arr_time <int>, sched_arr_time <int>, ...
```

DATA 100,Week 3 (B)

```
flights |>
relocate( year:day, distance, air_time,
.after = carrier,
# .before = dep_time
 )
```

```
# A tibble: 336,776 × 19
   dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier  year month   day distance air_time
      <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>   <int> <int> <int>    <dbl>    <dbl>
 1      517            515         2      830            819        11 UA       2013     1     1     1400      227
 2      533            529         4      850            830        20 UA       2013     1     1     1416      227
 3      542            540         2      923            850        33 AA       2013     1     1     1089      160
 4      544            545        -1     1004           1022       -18 B6       2013     1     1     1576      183
 5      554            600        -6      812            837       -25 DL       2013     1     1      762      116
 6      554            558        -4      740            728        12 UA       2013     1     1      719      150
 7      555            600        -5      913            854        19 B6       2013     1     1     1065      158
 8      557            600        -3      709            723       -14 EV       2013     1     1      229       53
 9      557            600        -3      838            846        -8 B6       2013     1     1      944      140
10      558            600        -2      753            745         8 AA       2013     1     1      733      138
# ℹ 336,766 more rows
# ℹ 7 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>, hour <dbl>, minute <dbl>, time_hour <dttm>
# ℹ Use `print(n = ...)` to see more rows
```

DATA 100,Week 3 (B)

Comments on |> v.s %>%

- |>: pipe introduced into **base** R in 2021, and is recommended most of the time

- %>%: pipe designed for tidyverse in magrittr package in 2014.

DATA 100,Week 3 (B)

group_by()

Groups the rows according to the values in specified columns to facilitates group statistics.

• Rows in the same group have the same value for all the specified columns.

flights_by_day contains **groups** of flights, where each group is formed by all flights in the same day

• Compare across different days – volume of flight, average delay, etc

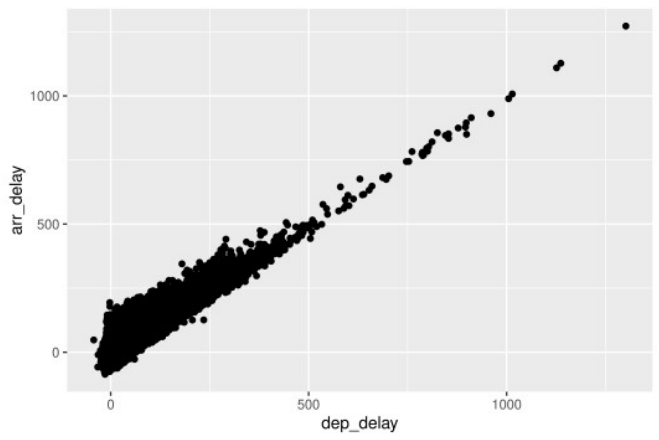
• Grouped dataframe is **different** from the original one as an object, even though contains the same data

• ungroup() will remove the group information

DATA 100,Week 3 (B)

```
flights_by_day <- flights |>
group_by(year, month, day)

flights_by_day |>
ggplot(mapping = aes(x = dep_delay, y = arr_delay)) + geom_jitter()
```

Warning message:Removed 9430 rows containing missing values or values outs ide the scale range (`geom_point()`).



DATA 100,Week 3 (B)

slice_() **takes some rows in each group**

There are 5 of them

- slice_head(): the first n = number of rows in each group

- slice_tail(): the last n = number of rows in each group

- slice_max(): the rows (could be more than one), in each group, that have the highest n = values in a column (as the first parameter)

- slice_min(): the rows (could be more than one), in each group, that have the lowest n = values in a column (as the first parameter)

- slice_sample(): randomly select n = number of rows in each group
n = can be replaced by prop = to take the corresponding percentage

DATA 100,Week 3 (B)

- Use prop = 0.25 to take 25% of each group

If the dataframe is **not** grouped, then the slice_ functions select from the whole dataframe.

If the dataframe is grouped, then the resulting dataframe is also grouped in the same way, and can be ungrouped.

If only need one row in slice_max and slice_min, add with_ties = FALSE

DATA 100,Week 3 (B)

```r
least_delay <- flights |>
group_by(dest) |>
slice_min(arr_delay, n = 1) |> # add `with_ties = FALSE` to see the difference
ungroup()
```

We can see the number of distinct dest is included in the result using n_distinct() after pull the dest column from the dataframe.

```r
least_delay |>
pull(dest) |> # this step outputs the column `dest` from `least_delay`
n_distinct()
#> [1] 105
```

DATA 100,Week 3 (B)

summarize() **get useful statistics**

- Recall summary() function

```
sample <- c(29, 35, 33, 32, 34, 30, 28, 33, 34, 35, 10, 14, 14, 12, 12, 12, 10, 14,
16, 10)
```

```
summary(sample)
```

```
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   10.00   12.00   22.00   22.35   33.00   35.00
```

DATA 100,Week 3 (B)

summarize() is similar, and more powerful with group_by()

Use ?summarize() to see a list of useful functions

- Center: mean(), median()
- Spread: sd(), IQR(), mad(), quantile()
- Range: min(), max()
- Position: first(), last(), nth(),
- Count: n(), n_distinct()
- Logical: any(), all()

DATA 100,Week 3 (B)

- Remove NA by `na.rm = TRUE`

  since computation involving **a single** missing value would give NA

- Look up with ? to see the option on NA for the functions

For example

Really need to find more about **why** missing value is in the dataset, which needs **domain knowledge** and / or **common sense**:

- e.g. in flights, the NA in delay times means canceled flight

- in fact, whenever dep_delay is NA, arr_delay is NA too

DATA 100,Week 3 (B)

```
dep_delayed <- flights |>
filter(is.na(dep_delay)) |>
count()

both_delayed <- flights |>
filter(is.na(dep_delay), is.na(arr_delay)) |>
count()

dep_delayed - both_delayed

#> n
#> 1 0
```

DATA 100,Week 3 (B)

Back to summarize(). The following counts the number of flights for year 2013, output a single row.

```
flights |>
summarize(
n = n(),
mean_delay = mean(arr_delay, na.rm = TRUE),
)

#>     # A tibble: 1 x 2
#>     n          mean_delay
#>      <int>          <dbl>
#> 1    336776          6.90
```

DATA 100,Week 3 (B)

With grouping by day, the following counts number of flights for each day in 2013, output **365 rows**.

• Look it up using ?summarize() for discussions of the parameter .groups

– basically, it specifies how the resulting dataframe deals with the grouping information

– drop means to keep no grouping information in the result

– which should suffice for most of our purposes

DATA 100,Week 3 (B)

- `.groups = "drop_last"` drops the last grouping level (i.e. the default behaviour sans message).
- `.groups = "drop"` drops all grouping levels and returns a tibble.
- `.groups = "keep"` preserves the grouping of the input.

DATA 100,Week 3 (B)

```
# Note: counts include flights that were canceled

day_count <- flights |>
group_by(year, month, day) |>
summarize(
count = n(),
mean_delay = mean(arr_delay, na.rm = TRUE),
.groups = 'drop'
)
```

DATA 100,Week 3 (B)

```
day_count
#> # A tibble: 365 x 5
#>    year month   day count mean_delay
#>   <int> <int> <int> <int>      <dbl>
#> 1  2013     1     1   842      12.7
#> 2  2013     1     2   943      12.7
#> 3  2013     1     3   914       5.73
#> 4  2013     1     4   915      -1.93
#> 5  2013     1     5   720      -1.53
#> 6  2013     1     6   832       4.24
#> # i 359 more rows
```

.by in summarize() and by in slice_ functions

```
day_count <- flights |>
summarize(
.by = c(year, month, day), n = n(),
mean_delay = mean(arr_delay, na.rm = TRUE), )

day_count
```

DATA 100,Week 3 (B)

```
#> # A tibble: 365 x 5
#>    year month   day     n mean_delay
#>   <int> <int> <int> <int>      <dbl>
#> 1  2013     1     1   842      12.7
#> 2  2013     1     2   943      12.7
#> 3  2013     1     3   914       5.73
#> 4  2013     1     4   915      -1.93
#> 5  2013     1     5   720      -1.53
#> 6  2013     1     6   832       4.24
#> # i 359 more rows
```

DATA 100,Week 3 (B)

```
least_delay <- flights |>
slice_min(
by = dest, n = 1,
arr_delay, with_ties = FALSE, )

#View(least_delay)
```

DATA 100,Week 3 (B)

Combination of tools

plot using the flights **data**

Take a look at the variables distance, arr_delay and dep_delay.

```
flights |>
select(distance, arr_delay, dep_delay) |>
summary()
```

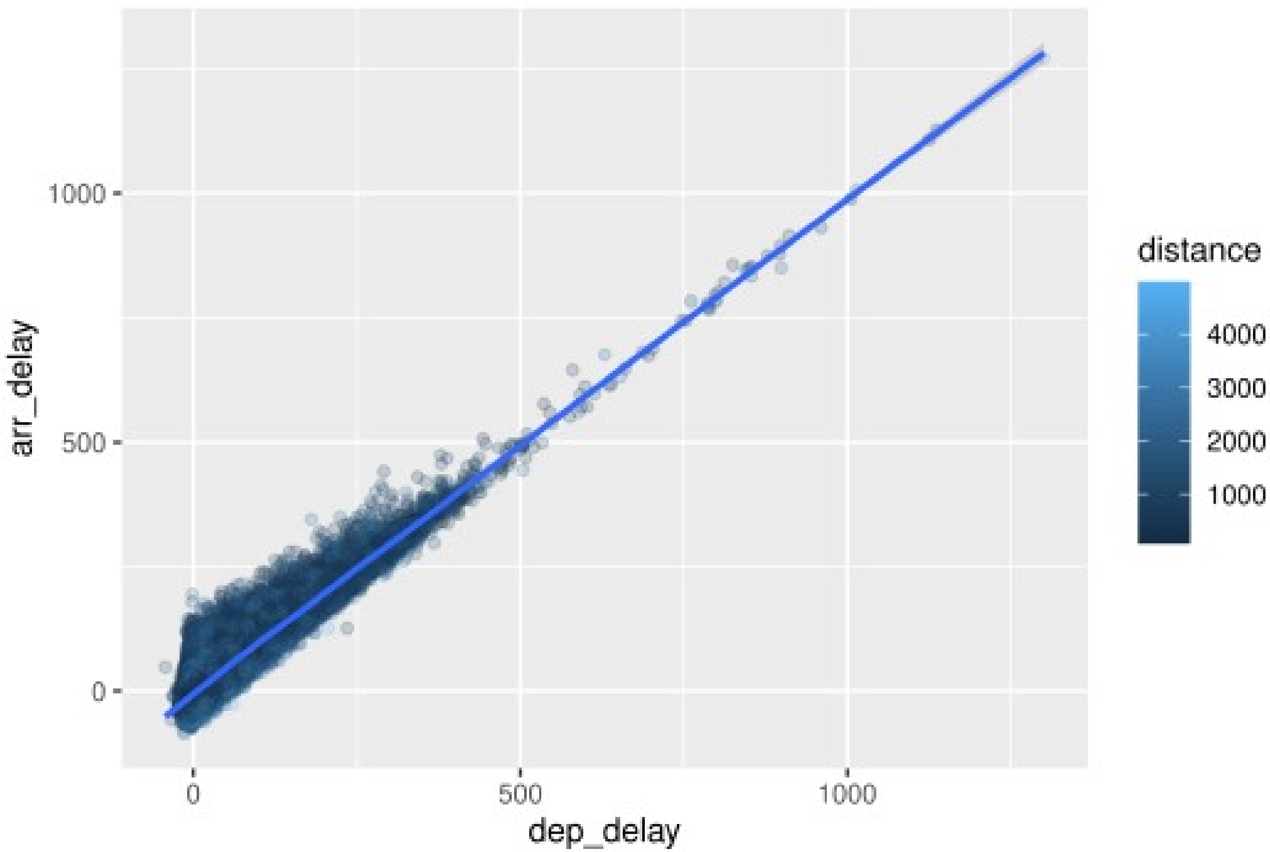
```
#>     distance       arr_delay           dep_delay
#>  Min.   :  17   Min.   : -86.000   Min.   : -43.00
#>  1st Qu.: 502   1st Qu.: -17.000   1st Qu.:  -5.00
#>  Median : 872   Median :  -5.000   Median :  -2.00
#>  Mean   :1040   Mean   :   6.895   Mean   :  12.64
#>  3rd Qu.:1389   3rd Qu.:  14.000   3rd Qu.:  11.00
#>  Max.   :4983   Max.   :1272.000   Max.   :1301.00
#>                 NA's   :9430       NA's   :8255
```

DATA 100,Week 3 (B)

Try to plot relations between arr_delay and dep_delay, expecting them to be related

```
flights |>
filter(!is.na(dep_delay), !is.na(arr_delay)) |> # valid flights
ggplot(mapping = aes(x = dep_delay, y = arr_delay)) + geom_point(mapping =
aes(color=distance), alpha = 0.2) + geom_smooth(se = TRUE)

#> `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
#geom_smooth(se = FALSE)
```

DATA 100,Week 3 (B)



DATA 100,Week 3 (B)

Selecting threshold to cut the data. For instance, use the outlier computation to get some ideas

Outliers are observations that are unusual, in other words, data points that don't seem to fit the pattern.

- ✓ Sometimes outliers are data entry errors, sometimes they are simply values at the extremes that happened to be observed in this data collection,
- ✓ and other times they suggest important new discoveries.

DATA 100,Week 3 (B)

```
dep_whisker <- IQR(flights$dep_delay, na.rm=TRUE) * 1.5

arr_whisker <- IQR(flights$arr_delay, na.rm=TRUE) * 1.5


not_out <- flights |>
filter(
dep_delay |> between(
quantile(dep_delay, prob = 0.25, na.rm = TRUE) - dep_whisker,
quantile(dep_delay, prob = 0.75, na.rm = TRUE) + dep_whisker
 ),
arr_delay |> between(
quantile(arr_delay, prob = 0.25, na.rm = TRUE) - arr_whisker,
quantile(arr_delay, prob = 0.75, na.rm = TRUE) + arr_whisker
) )
```

DATA 100,Week 3 (B)

```
total <- flights |> filter(!is.na(dep_delay))

count(not_out) / count(total)
```

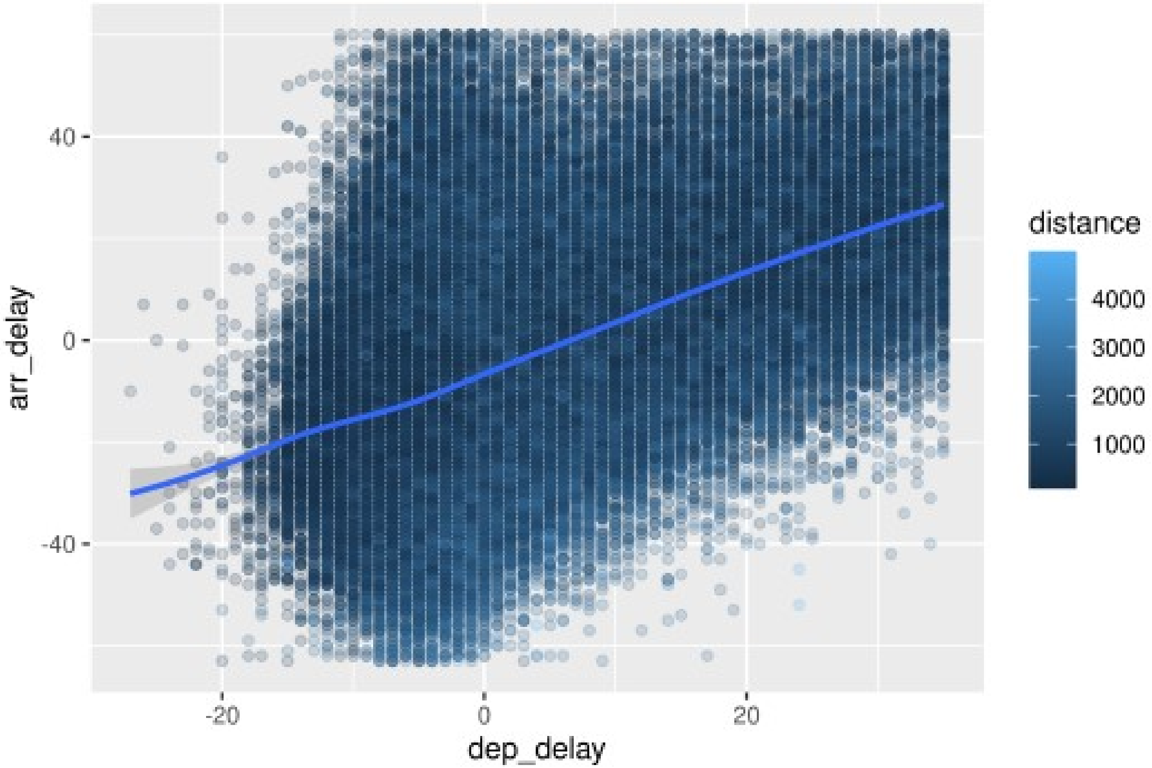
```
#>           n
#> 1 0.8586544
```

DATA 100,Week 3 (B)

Then plot only the non-outliers:

```
not_out |>
ggplot(mapping = aes(x = dep_delay, y = arr_delay)) + geom_point(mapping =
aes(color=distance), alpha = 0.2) + geom_smooth(se = TRUE)

#> `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
#geom_smooth(se = FALSE)
```

DATA 100,Week 3 (B)



Getting rid of the outliers changes the scale of the plot

DATA 100,Week 3 (B)

anti_join(x, y) is a useful tool here because it selects only the rows in x that don't have a match in y.

For example

```
#create data frames
df1 <- data.frame(team=c('A', 'B', 'C', 'D', 'E'),
                  points=c(12, 14, 19, 24, 36))

df2 <- data.frame(team=c('A', 'B', 'C', 'F', 'G'),
                  points=c(12, 14, 19, 33, 17))

library(dplyr)
```

DATA 100,Week 3 (B)

#perform anti join using 'team' column

```
anti_join(df1, df2, by='team')

  team points
1    D     24
2    E     36
```

DATA 100,Week 3 (B)

Example

#create data frames

```
df_a <- data.frame(team=c('A', 'A', 'A', 'B', 'B', 'B'),
                   position=c('G', 'G', 'F', 'G', 'F', 'C'),
                   points=c(12, 14, 19, 24, 36, 41))
```

df_a

```
  team position points
1    A        G     12
2    A        G     14
3    A        F     19
4    B        G     24
5    B        F     36
6    B        C     41
```

DATA 100,Week 3 (B)

```
df_b <- data.frame(team=c('A', 'A', 'A', 'B', 'B', 'B'),
            position=c('G', 'G', 'C', 'G', 'F', 'F'),
            points=c(12, 14, 19, 33, 17, 22))

df_b
```

```
  team position points
1    A        G     12
2    A        G     14
3    A        C     19
4    B        G     33
5    B        F     17
6    B        F     22
```

DATA 100,Week 3 (B)

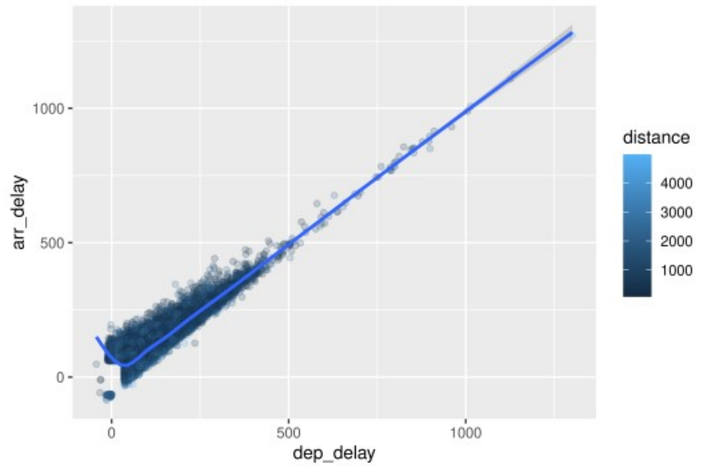
We can use the anti_join() function to return all rows in the first data frame that do not have a matching team and position in the second data frame:

```
library(dplyr)

#perform anti join using 'team' and 'position' columns

anti_join(df_a, df_b, by=c('team', 'position'))

  team position points
1    A        F     19
2    B        C     41
```

DATA 100,Week 3 (B)

Can also do the opposite, plotting only the outliers. The code below uses a function in dplyr that we will come back to: anti_join(), for the difference of dataframes total and not_out.

```
total |>
anti_join(not_out) |>
ggplot(mapping = aes(x = dep_delay, y = arr_delay)) + geom_point(mapping =
aes(color=distance), alpha = 0.2) + geom_smooth(se = TRUE)
```

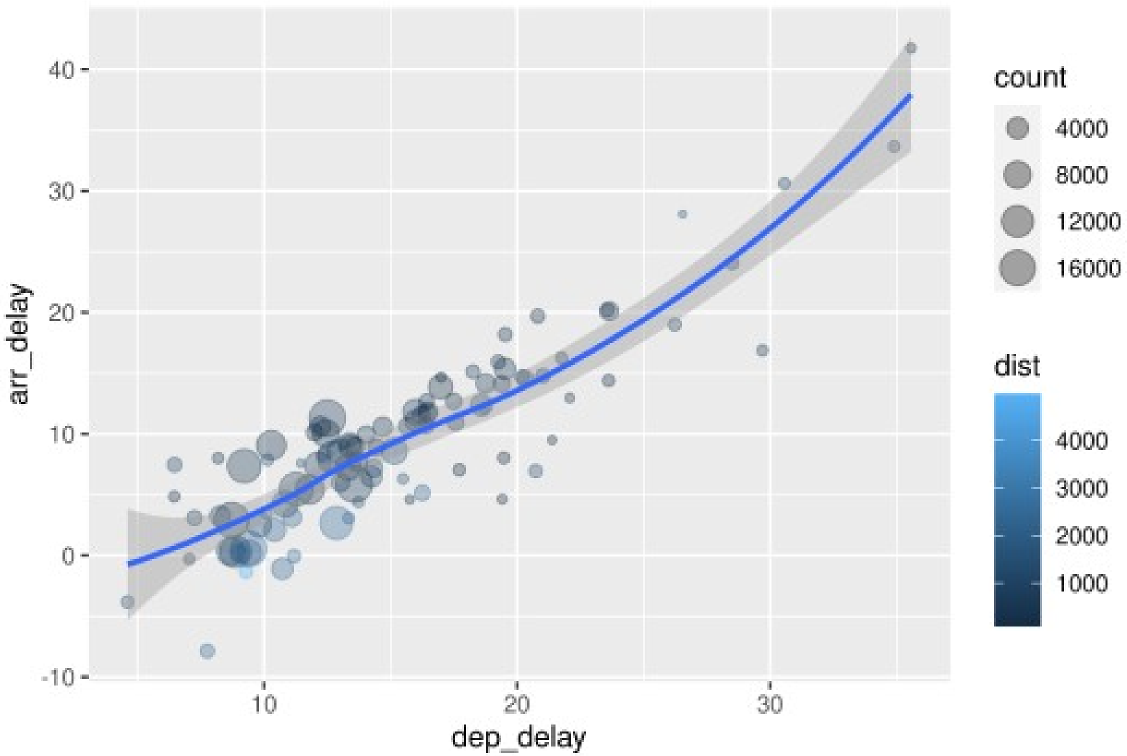


DATA 100,Week 3 (B)

Group then plot

```
flights |>
group_by(dest) |>
summarize( count = n(),
dist = mean(distance, na.rm = TRUE), # ? median
air_delay = mean(arr_delay - dep_delay, na.rm = TRUE), # ? median
arr_delay = mean(arr_delay, na.rm = TRUE), # ? median
dep_delay = mean(dep_delay, na.rm = TRUE) # ? median
 ) |>
filter(count > 20) |>
ggplot(mapping = aes(x = dep_delay, y = arr_delay)) + geom_point(aes(size =
count, color = dist), alpha = 1/3) + geom_smooth(se = TRUE)
```

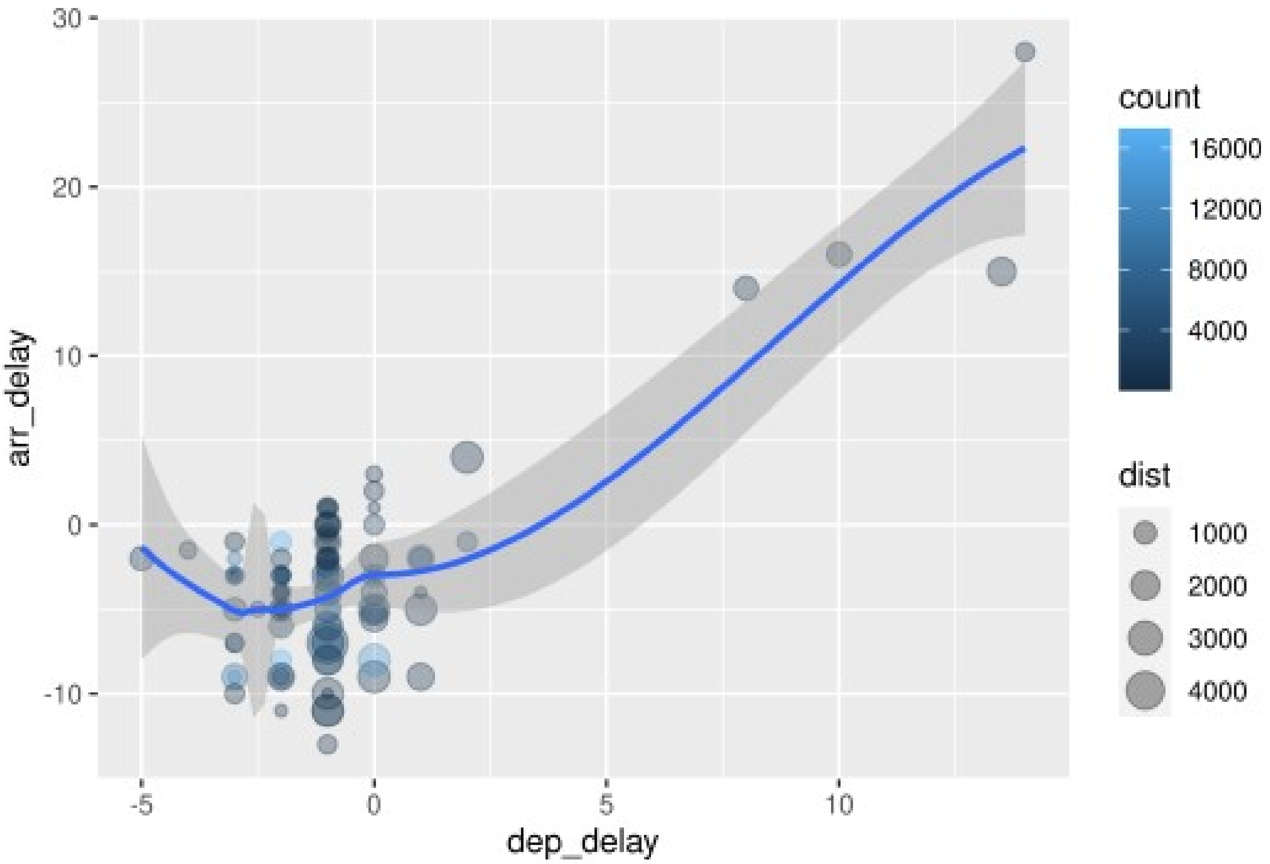
DATA 100,Week 3 (B)



DATA 100,Week 3 (B)

```
flights |>
group_by(dest) |>
summarize( count = n(),
dist = median(distance, na.rm = TRUE),
air_delay = median(arr_delay - dep_delay, na.rm = TRUE), arr_delay =
median(arr_delay, na.rm = TRUE),
dep_delay = median(dep_delay, na.rm = TRUE)
) |>
filter(count > 20) |>
ggplot(mapping = aes(x = dep_delay, y = arr_delay)) + geom_point(aes(size =
dist, color = count), alpha = 1/3) + geom_smooth(se = TRUE)
#> `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

DATA 100,Week 3 (B)



DATA 100,Week 3 (B)

Extreme delays

```
not_canceled <- flights |>
filter(!is.na(dep_delay), !is.na(arr_delay))

# How many planes are flying
n_distinct(not_canceled$tailnum)

# How many planes are flying
n_distinct(not_canceled$tailnum)
```

DATA 100,Week 3 (B)

```
delays <- not_canceled |>
summarize(
delay = mean(arr_delay), count = n(),
.by = tailnum )
```

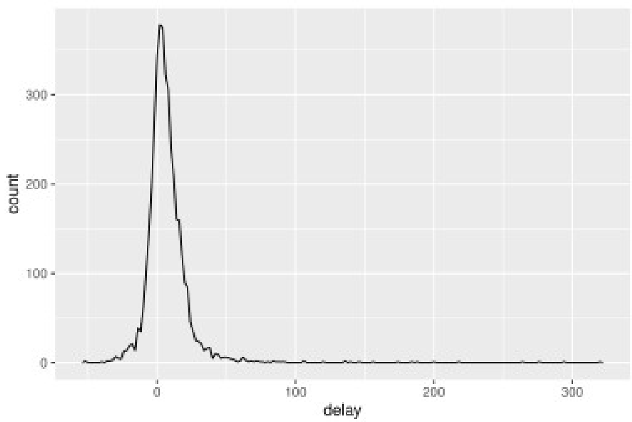
```
delays
```

```
#> # A tibble: 4,037 x 3
#>   tailnum delay count
#>   <chr>   <dbl> <int>
#> 1 N14228   3.71   111
#> 2 N24211   7.7    130
#> 3 N619AA   7.65    23
#> 4 N804JB  -1.86   215
#> 5 N668DN   2.62    48
#> 6 N39463   2.16   107
#> # i 4,031 more rows
```

DATA 100,Week 3 (B)

Filter out the planes that has low count

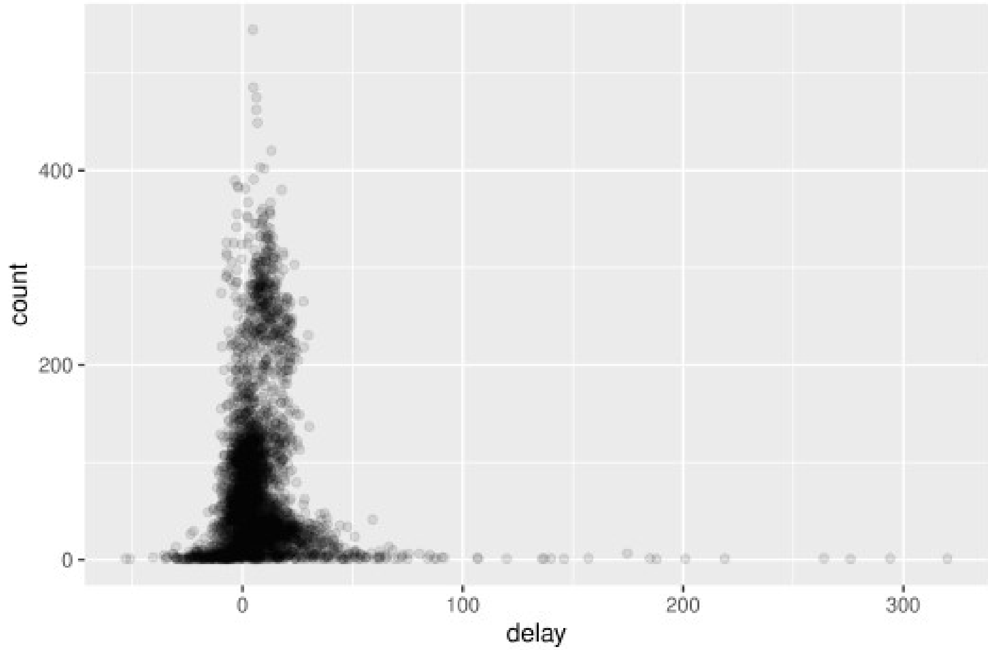
```
delays |>
# filter(count >= 20) |>
ggplot(mapping = aes(x = delay)) +
geom_freqpoly(binwidth = 2)
# use 2 when filtering, may have to change binwidth for details, with filter
```



The count in the plot above is the *computed* number of planes with the given average delays (Read ?geom_freqpoly)

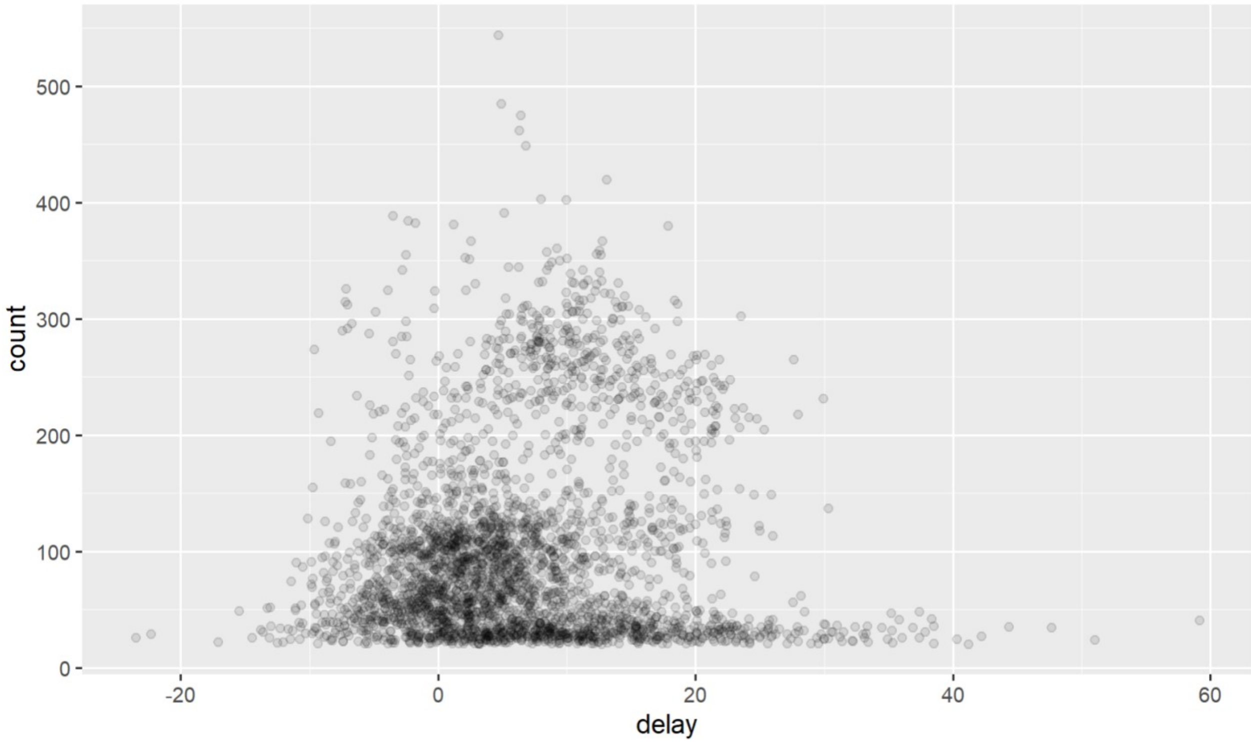
DATA 100,Week 3 (B)

```
delays |>
#filter(count > 20) |>
ggplot(mapping = aes(x = delay)) +
geom_point(mapping = aes(y = count), alpha = 1/10, position = "jitter")
```



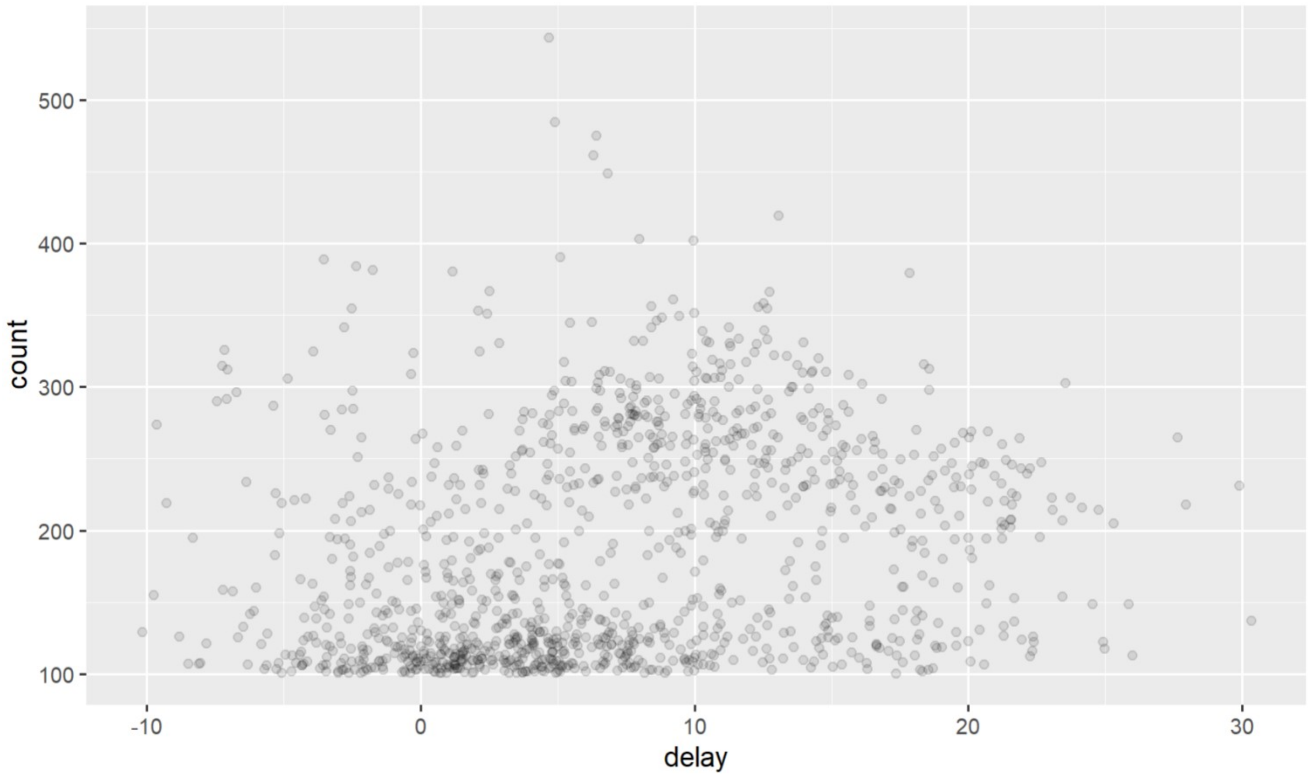
DATA 100,Week 3 (B)

```
delays |>
filter(count > 20) |>
ggplot(mapping = aes(x = delay)) +
geom_point(mapping = aes(y = count), alpha = 1/10, position = "jitter")
```



DATA 100,Week 3 (B)

```
delays |>
filter(count > 100) |>
ggplot(mapping = aes(x = delay)) +
geom_point(mapping = aes(y = count), alpha = 1/10, position = "jitter")
```



DATA 100,Week 3 (B)

```
delays |>
summarize(
median_count = median(count), max_count = max(count), mean_count =
mean(count), median_delay = median(delay), max_delay = max(delay),
mean_delay = mean(delay)
)
```

```
#> # A tibble: 1 x 6
#>   median_count max_count mean_count median_delay max_delay mean_delay
#>          <int>     <int>      <dbl>        <dbl>     <dbl>      <dbl>
#> 1           53       544       81.1         4.84       320       7.09
```

DATA 100,Week 3 (B)

```
quantile(delays$count,
probs = c(0,0.25,0.5,0.75,1))
```

```
#>     0%   25%   50%   75%  100%
#>      1    23    53   109   544
```

DATA 100,Week 3 (B)

Descriptive statistics, III

Discussed here are **frequency, relative frequency, mode** and **skewness**

These statistics concerns the distribution of the data

```
# Use the same collection of 20 integers
sample <- c(29, 35, 33, 32, 34, 30, 28, 33, 34, 35, 10, 14, 14, 12, 12, 12, 10, 14,
16, 10)
# together with a new collection of 30 doubles
sample2 <- c(38.47, 35.61, 34.45, 33.37, 34.92, 40.27, 33.07, 35.21, 36.84,
40.06, 32.12, 40.25, 36.68, 33.71, 29.79, 33.59, 38.95, 40.06, 35.24, 37.32,
22.19, 25.66, 21.31, 24.73, 19.42, 17.18, 21.12, 21.61, 17.83, 23.85)
```

DATA 100,Week 3 (B)

frequency

For a categorical variable, it describes the number of times the data taking a value - represented well using a bar plot, e.g. by geom_bar()

For a continuous variable, it describes the number of times the data taking values in an interval - represented well using a histogram, e.g. by geom_histogram() or geom_freqpoly()

sample can be regarded as categorical data, as it only takes finite values in integers

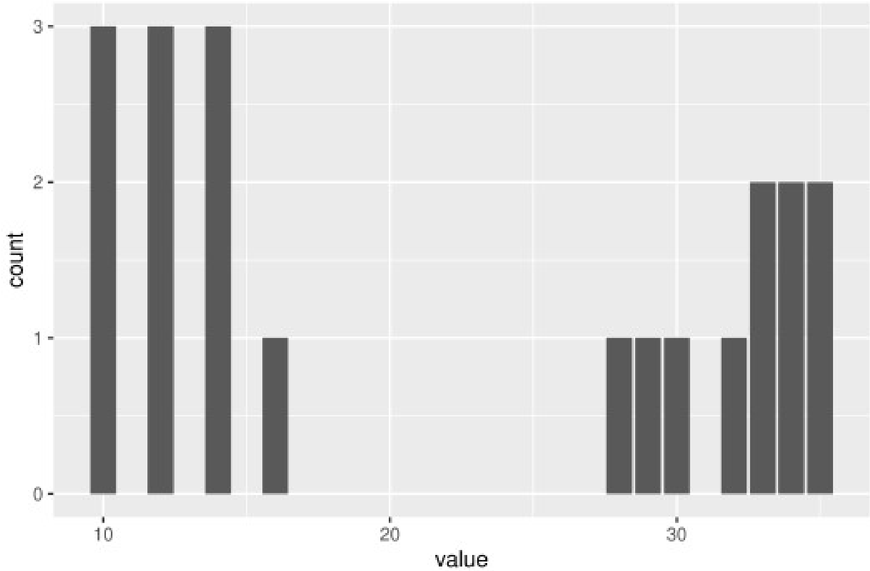
To manually see the distribution, it's better to have the data sorted

```
sort(sample)

#> [1] 10 10 10 12 12 12 14 14 14 16 28 29 30 32 33 33 34 34 35 35
```

DATA 100,Week 3 (B)

Then we compare it with the bar plot below

```
enframe(sample) |>
ggplot() +
geom_bar(mapping = aes(x = value))
```



It clearly shows that the data is clustered around two ends with nothing in the middle.

DATA 100,Week 3 (B)

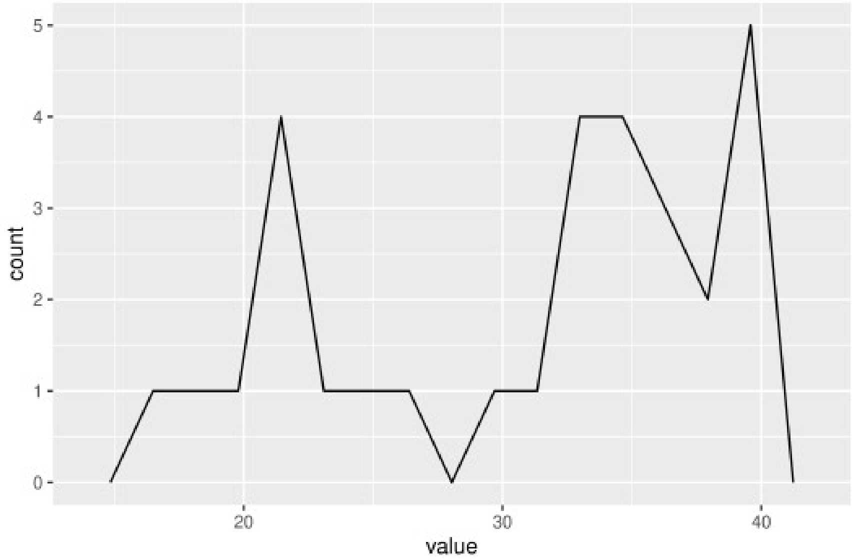
sample2 can be regarded as continuous data, as it takes values in double. Similarly, we can sort the sample2 data

```
sort(sample2)
```

```
#> [1] 17.18 17.83 19.42 21.12 21.31 21.61 22.19 23.85 24.73 25.66 29.79 32.12
#> [13] 33.07 33.37 33.59 33.71 34.45 34.92 35.21 35.24 35.61 36.68 36.84 37.32
#> [25] 38.47 38.95 40.06 40.06 40.25 40.27
```

```
enframe(sample2) |>
ggplot() +
geom_freqpoly(mapping = aes(x = value), bins = 15)
```

DATA 100,Week 3 (B)



relative frequency

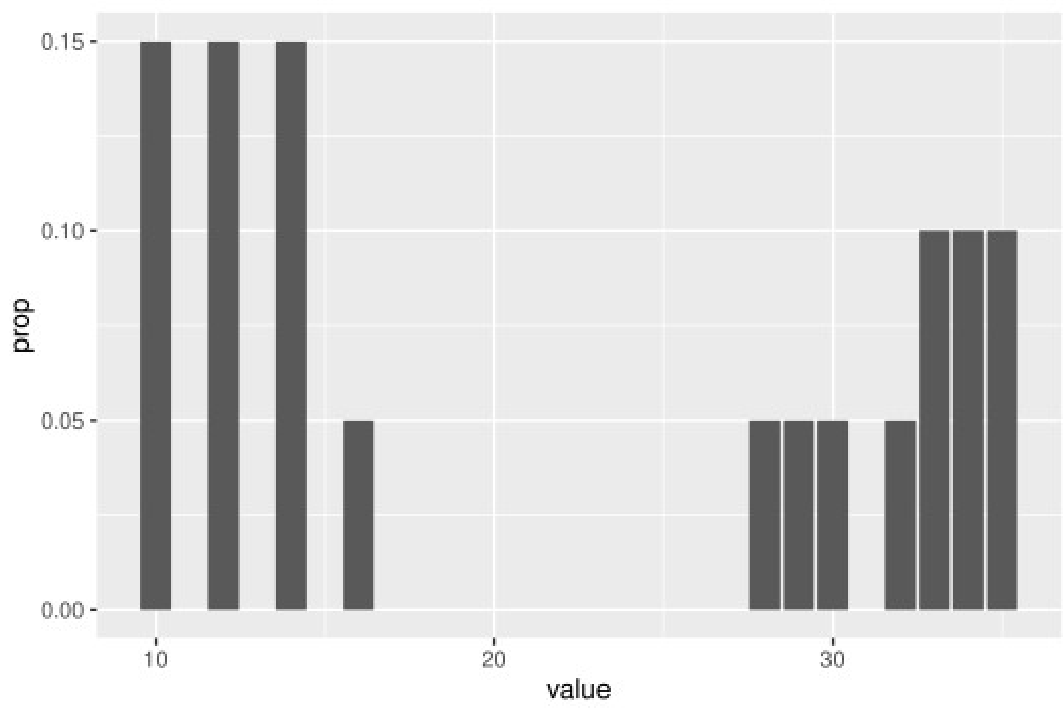
This is simply the frequency divided by the total number of data entries. It comes out as a percentage

• the plots will look similar, while with different values on the y-axis

• In geom_bar, the relative frequency is plotted using after_stat(prop)

DATA 100,Week 3 (B)

```
enframe(sample) |>
ggplot() +
geom_bar(mapping = aes(x = value, y = after_stat(prop), group = 1))
```



DATA 100,Week 3 (B)

after_stat(density) plots the relative frequency for geom_freqpoly or geom_histogram

```
ggplot(data = enframe(sample2)) + geom_freqpoly(mapping = aes(x = value, y = after_stat(density)), bins = 15)

#geom_histogram(mapping = aes(x = value, y = after_stat(density)), bins = 15)
```

DATA 100,Week 3 (B)

mode

• For discrete / categorical variable, mode are the values that have the highest frequency

• For continuous variable, mode are the local maximal of the probability density function

Generally, mode is the highest points of the distribution plot of the data

• For sample, the mode would be 10, 12, 14

• For sample2, the mode would be somewhere around 39.5

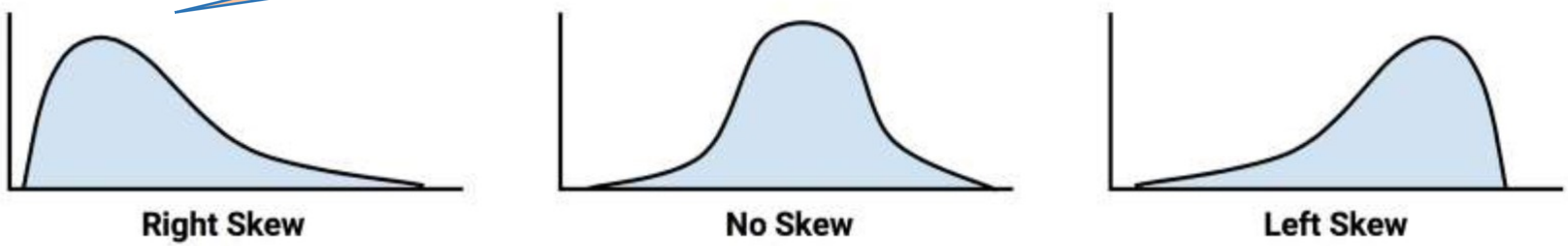
DATA 100,Week 3 (B)

```
sort(sample)
#> [1] 10 10 10 12 12 12 14 14 14 16 28 29 30 32 33 33 34 34 35 35
```

```
sort(sample2)
#> [1] 17.18 17.83 19.42 21.12 21.31 21.61 22.19 23.85 24.73 25.66 29.79 32.12
#> [13] 33.07 33.37 33.59 33.71 34.45 34.92 35.21 35.24 35.61 36.68 36.84 37.32
#> [25] 38.47 38.95 40.06 40.06 40.25 40.27
```

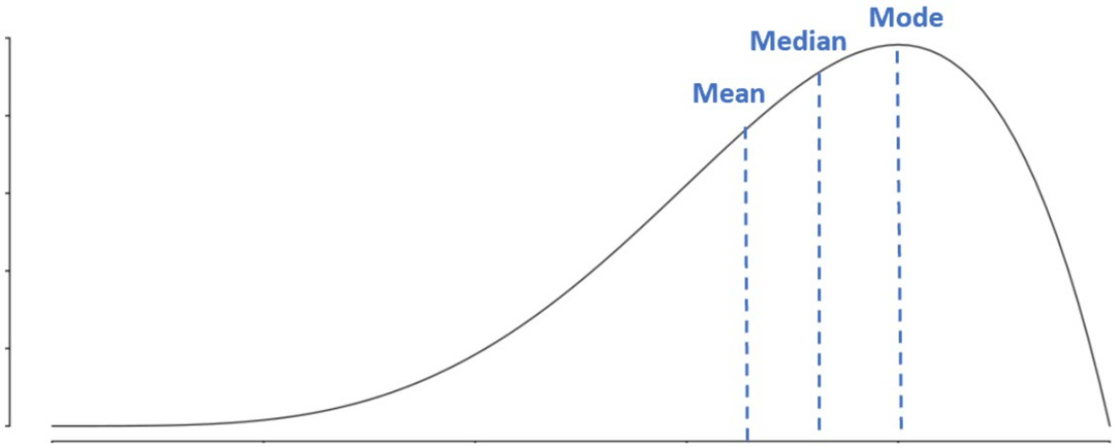
DATA 100,Week 3 (B)

skewed to the right, right skew, right-tailed, positive skew
– If the mean is greater than the median
– If the *right* tail of the distribution is more drawn out
– The distribution curve (bar/histogram plot) looks to be lumped to the *left*



skewed to the left, left skew, left-tailed, negative skew
– If the mean is less than the median
– If the *left* tail of the distribution is more drawn out
– The distribution curve (bar/histogram plot) looks to be lumped to the *right*

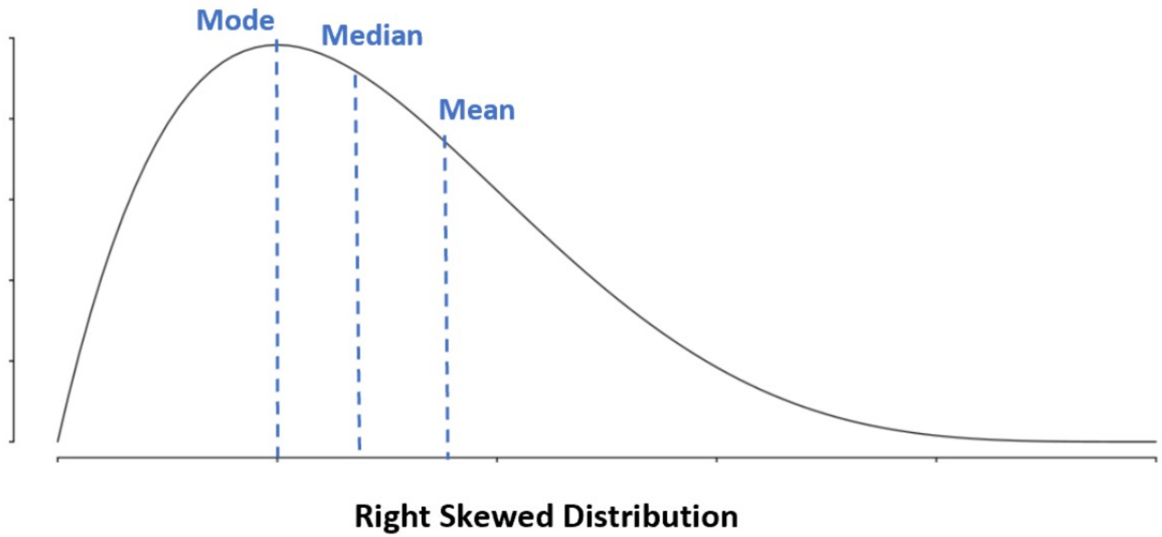
DATA 100,Week 3 (B)



Left Skewed Distribution

Left Skewed Distribution: Mean < Median < Mode

DATA 100,Week 3 (B)



Right Skewed Distribution: Mode < Median < Mean

In statistics, the mode is the value that appears most often in a set of data values.

DATA 100,Week 3 (B)

```
mean(sample2) - median(sample2)
```

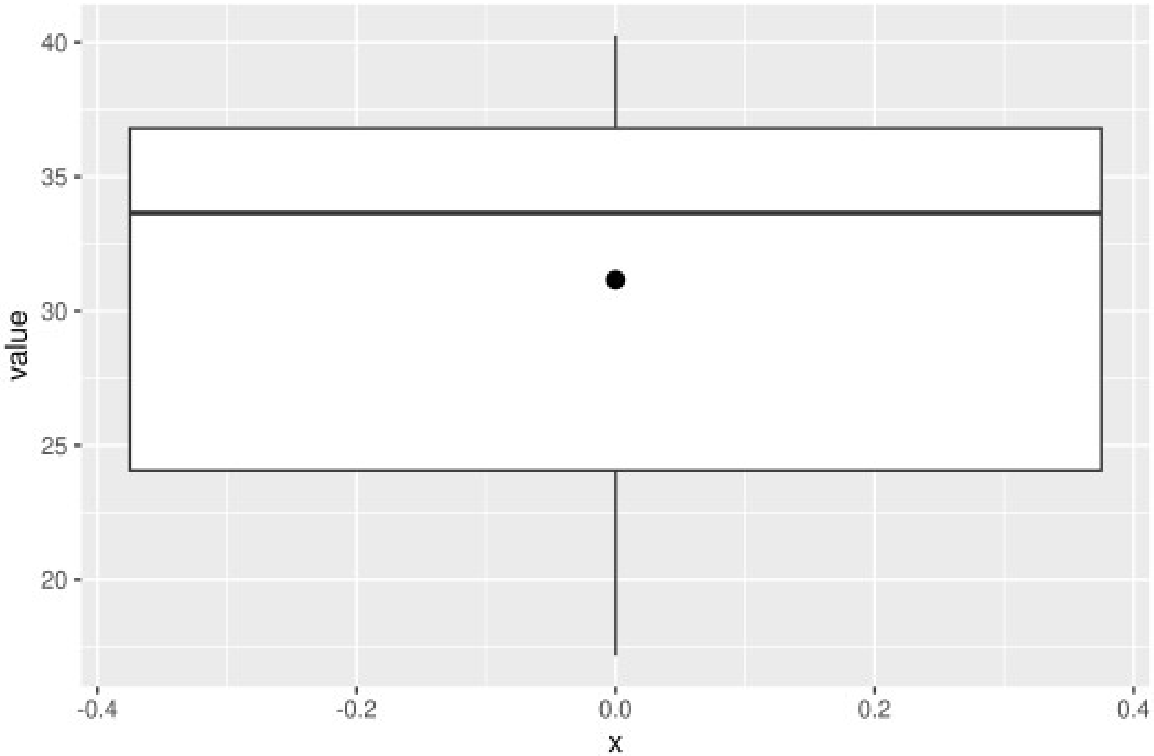
```
#> [1] -2.487333
```

Thus, the data in sample2 is slightly skewed to the left

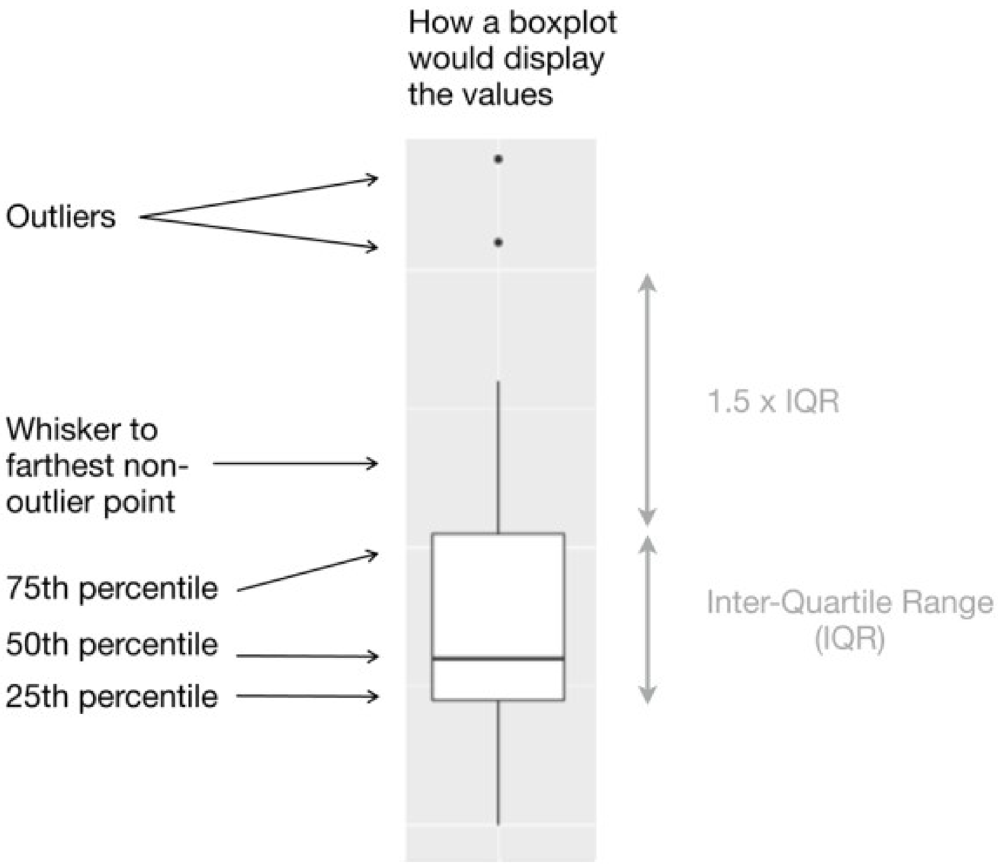
```
ggplot(data = enframe(sample2), mapping = aes(y = value)) + geom_boxplot() +
stat_summary(mapping = aes(x = 0), fun=mean)
```

```
#> Warning: Removed 1 rows containing missing values (`geom_segment()`).
```

DATA 100,Week 3 (B)



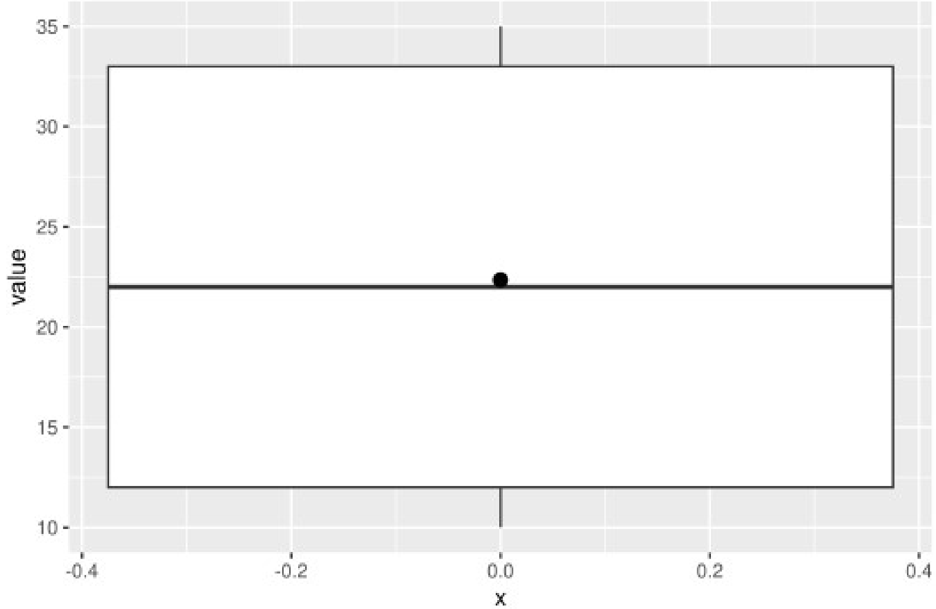
DATA 100,Week 3 (B)



DATA 100,Week 3 (B)

```
mean(sample) - median(sample)
#> [1] 0.35

ggplot(data = enframe(sample), mapping = aes(y = value)) + geom_boxplot() +
stat_summary(mapping = aes(x = 0), fun=mean)

#> Warning: Removed 1 rows containing missing values (`geom_segment()`).
```

DATA 100,Week 3 (B)



The data in sample is very slightly skewed to the right

- which really should not be regarded as skewed at all.