Exploring the abacus dataset (Barner, et al. 2018)

Modern Research Methods

Practice Exercise 1:

```
pet_df
```

```
## # A tibble: 5 x 3
##   family pet_type   age
##   <chr>  <chr>    <dbl>
## 1 Ung    cat         10
## 2 Ung    dog          4
## 3 Ung    cat          2
## 4 Oliver dog          8
## 5 Oliver dog         12
```

Sketch the resulting data frame:

```
pet_df %>%
  group_by(pet_type) %>%
  summarize(num = n(),
            mean = mean(age))
```

Practice Exercise 2:

```
pet_df
```

```
## # A tibble: 5 x 3
##   family pet_type   age
##   <chr>  <chr>    <dbl>
## 1 Ung    cat         10
## 2 Ung    dog          4
## 3 Ung    cat          2
## 4 Oliver dog          8
## 5 Oliver dog         12
```

Sketch the resulting data frame:

```
pet_df %>%
  group_by(pet_type, family) %>%
  summarize(num = n(),
            mean = mean(age))
```

Practice Exercise 3:

```
country_df
```

```
## # A tibble: 6 x 4
##   country   continent population square_miles
##   <chr>     <chr>          <dbl>        <dbl>
## 1 Anglia    X                 50           10
## 2 Xeno      X                 40          100
## 3 Spatarium X                 20           30
## 4 Pluti     Y                 10           40
## 5 Cincroane Z                 70           20
## 6 Franktum  Z                 40           60
```

Sketch the resulting data frame:

```
country_df %>%
  group_by(continent) %>%
  summarize(sum_population = sum(population),
            sum_square_miles = sum(square_miles))
```

Practice Exercise 4:

```
country_df
```

```
## # A tibble: 6 x 4
##   country   continent population square_miles
##   <chr>     <chr>          <dbl>        <dbl>
## 1 Anglia    X                 50           10
## 2 Xeno      X                 40          100
## 3 Spatarium X                 20           30
## 4 Pluti     Y                 10           40
## 5 Cincroane Z                 70           20
## 6 Franktum  Z                 40           60
```

Sketch the resulting data frame:

```
country_df %>%
  group_by(country) %>%
  summarize(num = n())
```

Exercise 1

```
pet_df
```

```
## # A tibble: 5 x 3
##   family pet_type   age
##   <chr>  <chr>    <dbl>
## 1 Ung    cat         10
## 2 Ung    dog          4
## 3 Ung    cat          2
## 4 Oliver dog          8
## 5 Oliver dog         12
```

```
pet_df %>%
  group_by(pet_type) %>%
  summarize(num = n(),
            mean = mean(age))
```

```
## # A tibble: 2 x 3
##   pet_type   num  mean
##   <chr>    <int> <dbl>
## 1 cat          2     6
## 2 dog          3     8
```

Exercise 2

```
pet_df
```

```
## # A tibble: 5 x 3
##   family pet_type   age
##   <chr>  <chr>    <dbl>
## 1 Ung    cat         10
## 2 Ung    dog          4
## 3 Ung    cat          2
## 4 Oliver dog          8
## 5 Oliver dog         12
```

```
pet_df %>%
  group_by(pet_type, family) %>%
  summarize(num = n(),
            mean = mean(age))
```

```
## # A tibble: 3 x 4
## # Groups:   pet_type [2]
##   pet_type family   num  mean
##   <chr>    <chr>  <int> <dbl>
## 1 cat      Ung        2     6
## 2 dog      Oliver     2    10
## 3 dog      Ung        1     4
```

Exercise 3

```
country_df
```

```
## # A tibble: 6 x 4
##   country   continent population square_miles
##   <chr>     <chr>          <dbl>        <dbl>
## 1 Anglia    X                 50           10
## 2 Xeno      X                 40          100
## 3 Spatarium X                 20           30
## 4 Pluti     Y                 10           40
## 5 Cincroane Z                 70           20
## 6 Franktum  Z                 40           60
```

```
country_df %>%
  group_by(continent) %>%
  summarize(sum_population = sum(population),
            sum_square_miles = sum(square_miles))
```

```
## # A tibble: 3 x 3
##   continent sum_population sum_square_miles
##   <chr>              <dbl>            <dbl>
## 1 X                    110              140
## 2 Y                     10               40
## 3 Z                    110               80
```

Exercise 4

```
country_df
```

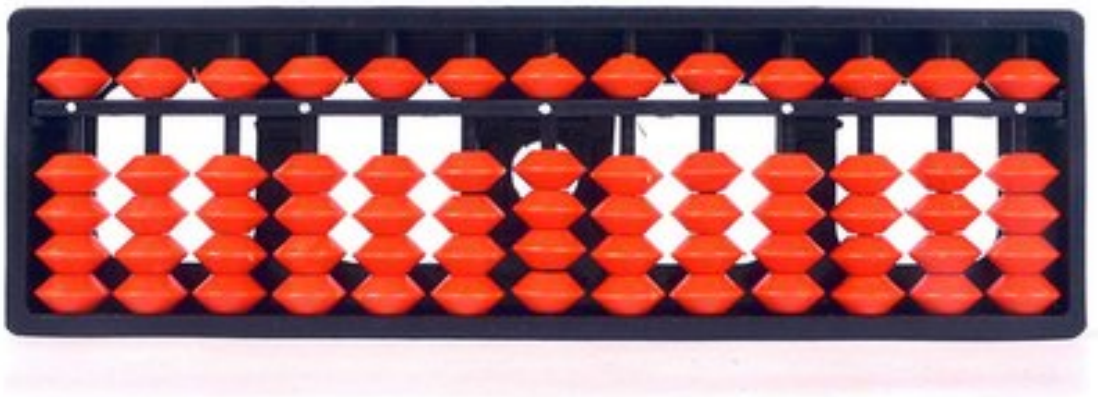
```
## # A tibble: 6 x 4
##   country   continent population square_miles
##   <chr>     <chr>          <dbl>        <dbl>
## 1 Anglia    X                 50           10
## 2 Xeno      X                 40          100
## 3 Spatarium X                 20           30
## 4 Pluti     Y                 10           40
## 5 Cincroane Z                 70           20
## 6 Franktum  Z                 40           60
```

```
country_df %>%
  group_by(country) %>%
  summarize(num = n())
```

```
## # A tibble: 6 x 2
##   country     num
##   <chr>     <int>
## 1 Anglia        1
## 2 Cincroane     1
## 3 Franktum      1
## 4 Pluti         1
## 5 Spatarium     1
## 6 Xeno          1
```

abacus dataset

- Does training kids to use an abacus help with their math skills?



Let's read in the dataset

```
abacus_data <- read_csv("data/tidy_majic_data.csv")
```

```
## Parsed with column specification:
## cols(
##   subid = col_character(),
##   class_num = col_character(),
##   grade = col_character(),
##   group = col_character(),
##   year = col_double(),
##   time = col_character(),
##   task = col_character(),
##   score = col_double()
## )
```

Here's what it looks like

```
abacus_data %>%
  head() %>%
  kable(format = "html")
```

subid	class_num	grade	group	year	time	task	score
S1-02-02	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2016	Post	Place Value	0.36
S1-02-08	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-08	S1_02	First Grade	Control	2016	Post	Place Value	0.36

A new tidyverse function: `slice()`

- `slice()` is like `filter()` in that it subsets your dataframe by rows.
- Filter subsets rows through Booleans
- Slice subsets rows through indices
- For example, the following code selects the 2nd and 3rd row from the data frame.

```
abacus_data %>%
  slice(c(2,3)) %>%
  kable(format = "html")
```

subid	class_num	grade	group	year	time	task	score
S1-02-03	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2016	Post	Place Value	0.36

- We can also specify a range of indices. The following code uses the ":" symbol to subset to rows 2, 3, and 4.

```
abacus_data %>%
slice(2:4) %>%
 kable(format = "html")
```

subid	class_num	grade	group	year	time	task	score
S1-02-03	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2016	Post	Place Value	0.36
S1-02-08	S1_02	First Grade	Control	2015	Pre	Place Value	0.00

- How would you produce the same output as `head()` (default)?

```
abacus_data %>%
slice(1:6) %>%
 kable(format = "html")
```

subid	class_num	grade	group	year	time	task	score
S1-02-02	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2016	Post	Place Value	0.36
S1-02-08	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-08	S1_02	First Grade	Control	2016	Post	Place Value	0.36
S1-02-15	S1_02	First Grade	Control	2016	Post	Place Value	0.64

A new tidyverse function: `distinct()`

- `distinct()` returns a subset of rows in your data frame (similiar to `filter()`)
- Specifically, `distinct` returns ONE row in your data frame for each value of a variable you pass it.
- The `abacus_data` data frame contains 2,094 rows - one row for each subject-task-time combination.

- For example, the following code returns a data fram with ONE row for each subject id.

```
abacus_data %>%
 distinct(subid)
```

```
## # A tibble: 188 x 1
##    subid
##    <chr>
##  1 S1-02-02
##  2 S1-02-03
##  3 S1-02-08
##  4 S1-02-15
##  5 S1-02-17
##  6 S1-03-04
##  7 S1-03-05
##  8 S1-03-06
##  9 S1-03-09
## 10 S1-03-14
## # … with 178 more rows
```

- The following code returns a data frame with ONE row for each subject id and time.

```
abacus_data %>%
 distinct(subid, time)
```

```
## # A tibble: 349 x 2
##    subid    time
##    <chr>    <chr>
##  1 S1-02-02 Pre
##  2 S1-02-03 Pre
##  3 S1-02-03 Post
##  4 S1-02-08 Pre
##  5 S1-02-08 Post
##  6 S1-02-15 Post
##  7 S1-02-17 Pre
##  8 S1-02-17 Post
##  9 S1-03-04 Post
## 10 S1-03-05 Post
## # … with 339 more rows
```

- You can keep the other variables in the data frame by adding the argument `.keep_all = T` to `distinct()`.

```
abacus_data %>%
distinct(subid, time, .keep_all = T)
```

```
## # A tibble: 349 x 8
##    subid    class_num grade       group          year time  task        score
##    <chr>    <chr>     <chr>       <chr>         <dbl> <chr> <chr>       <dbl>
##  1 S1-02-02 S1_02     First Grade Control        2015 Pre   Place Value  0
##  2 S1-02-03 S1_02     First Grade Control        2015 Pre   Place Value  0
##  3 S1-02-03 S1_02     First Grade Control        2016 Post  Place Value  0.36
##  4 S1-02-08 S1_02     First Grade Control        2015 Pre   Place Value  0
##  5 S1-02-08 S1_02     First Grade Control        2016 Post  Place Value  0.36
##  6 S1-02-15 S1_02     First Grade Control        2016 Post  Place Value  0.64
##  7 S1-02-17 S1_02     First Grade Control        2015 Pre   Place Value  0.09
##  8 S1-02-17 S1_02     First Grade Control        2016 Post  Place Value NA
##  9 S1-03-04 S1_03     First Grade Mental Abacus  2016 Post  Place Value  0.55
## 10 S1-03-05 S1_03     First Grade Mental Abacus  2016 Post  Place Value  0.91
## # … with 339 more rows
```

Let's get started with exploring these data.

- Let's look at the scores in this datasets.
- What kind of variable is scores?
- What kind of geoms could we use to look at scores?
- Let's try visualizing the scores with a histogram
- The geom for histograms is `geom_histogram()`

How do we start? What aesthetics does `geom_histogram()` take?

subid	class_num	grade	group	year	time	task	score
S1-02-02	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2016	Post	Place Value	0.36
S1-02-08	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-08	S1_02	First Grade	Control	2016	Post	Place Value	0.36
S1-02-15	S1_02	First Grade	Control	2016	Post	Place Value	0.64

A basic ggplot

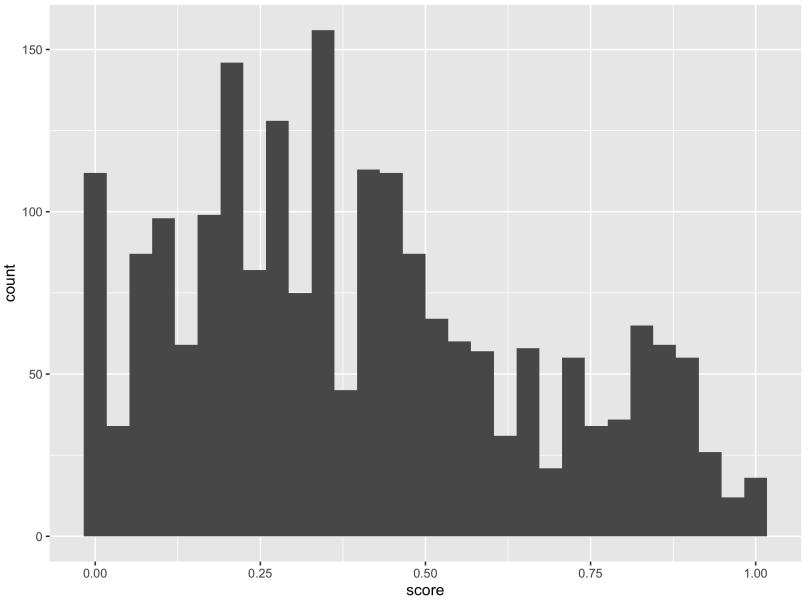
```
ggplot(data = abacus_data, aes(x = score)) +
  geom_histogram()
```

```
ggplot(abacus_data, aes(x = score)) +
  geom_histogram()
```

Note that you don't have to name the arguments, though it's usually a good idea to.

This is great, except we can't see the different tasks.

What can we do about this?

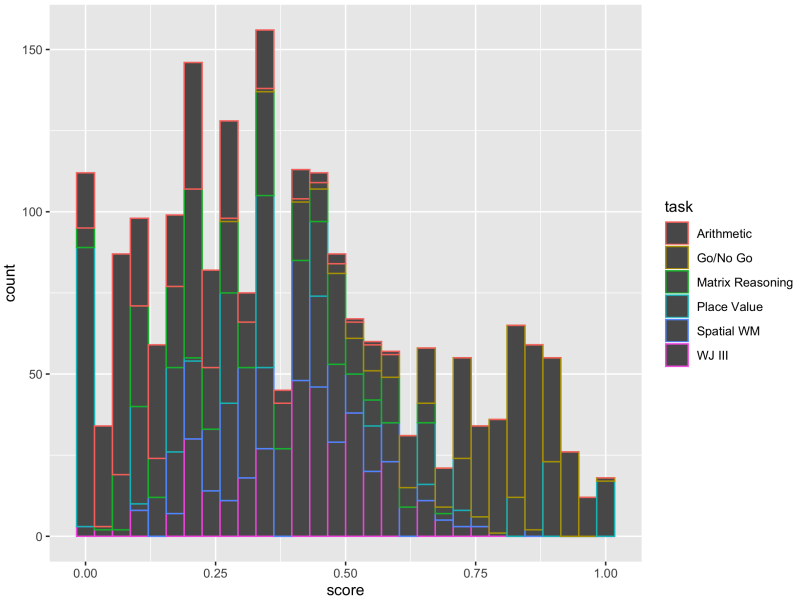


We could try coloring the different task differently.

```
ggplot(abacus_data, aes(x = score, color = task)) +
  geom_histogram()
```

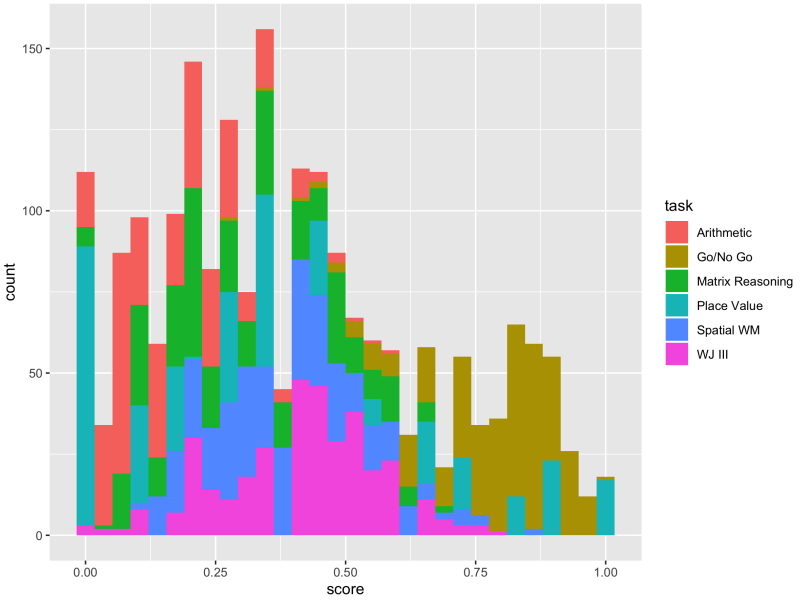
That's not quite want we wanted. We wanted the **fill** aesthetic.

The **color** aesthetic determines the hue of the border; the **fill** aesthetic determines the hue of the inside of the bar.



```
ggplot(abacus_data, aes(x = score, fill = task)) +
  geom_histogram()
```

This is looking better. But it's still a lot of information.



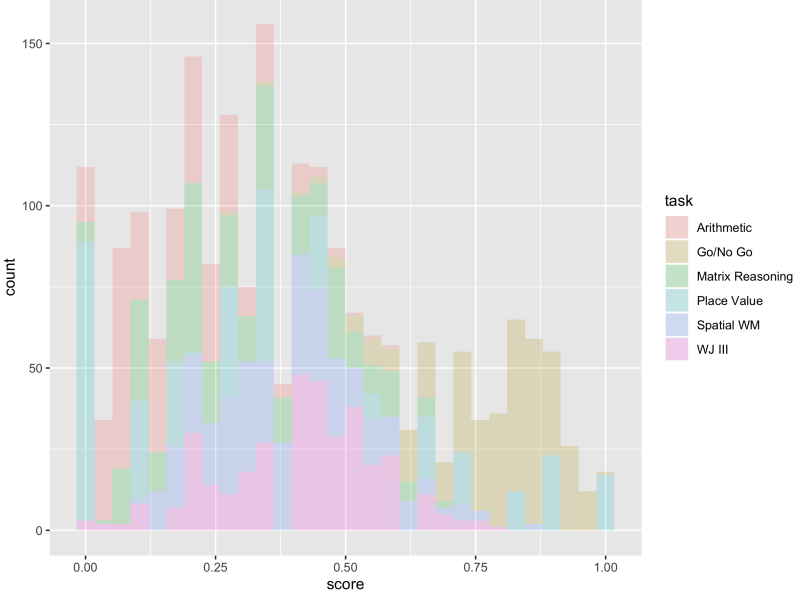
A new channel: alpha

Alpha controls how "transparent" the geom is.

It ranges from zero to one. One is most transparent.

```
ggplot(abacus_data, aes(x = score, fill = task)) +
  geom_histogram(alpha = .2)
```

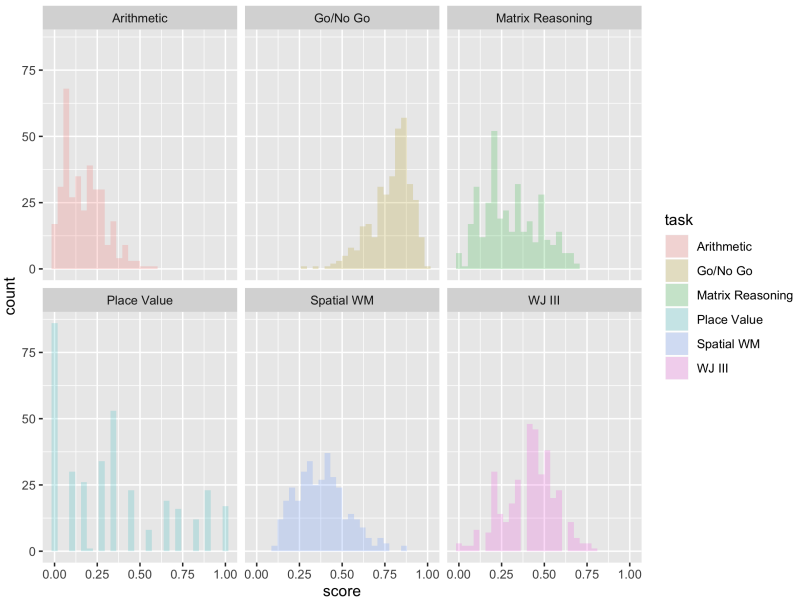
This is still a lot of information. What could we do?



Facet!

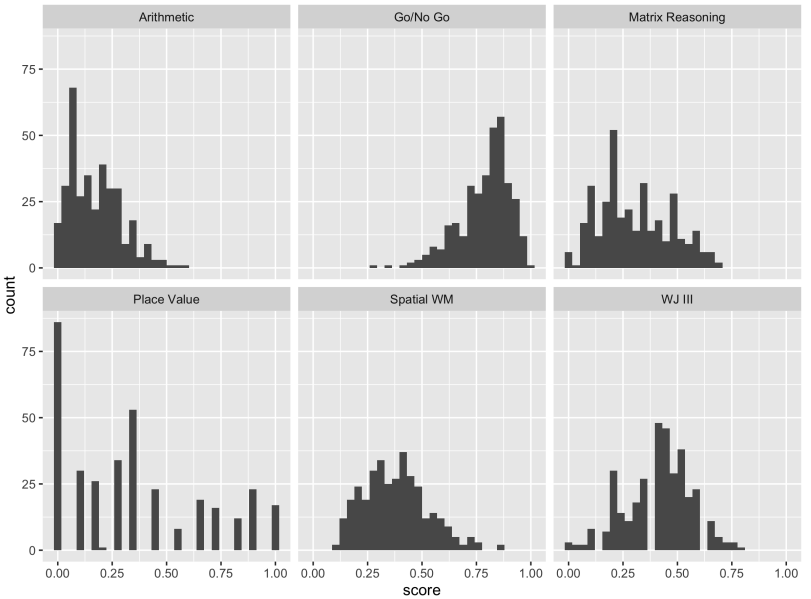
Let's use `facet_wrap()` to facet by task.

```
ggplot(abacus_data, aes(x = score, fill = task)) +
  geom_histogram(alpha = .2) +
  facet_wrap(~task)
```



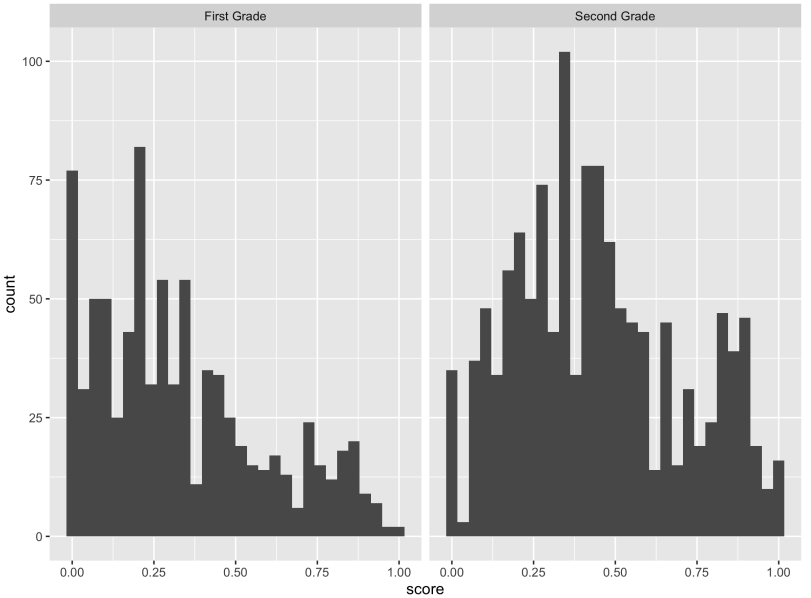
The fill and alpha aesthetics aren't really doing anything for us here. Let's get rid of them

```
ggplot(abacus_data, aes(x = score)) +
  geom_histogram() +
  facet_wrap(~task)
```



What if we wanted to facet by grade rather than task?

```
ggplot(abacus_data, aes(x = score)) +
  geom_histogram() +
  facet_wrap(~grade)
```



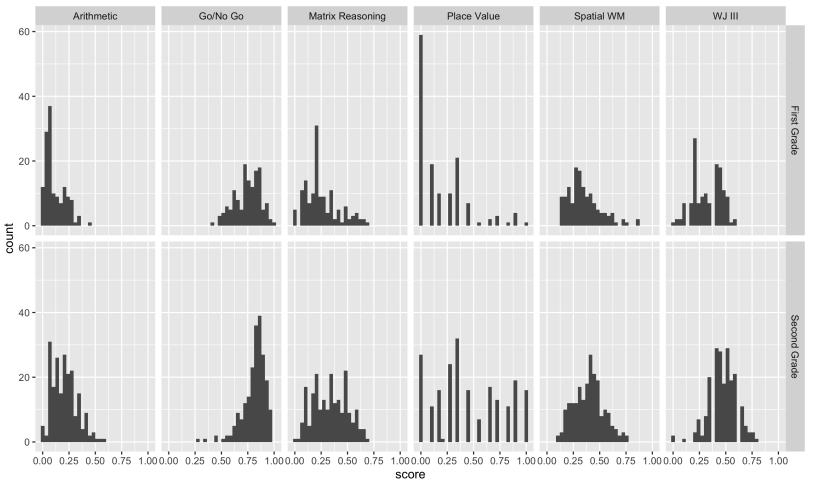
How would we facet by BOTH grade and task?

`facet_grid()`

How would we put the different task facets on horizontally, and the grade facets vertically?

The general syntax is "vertical variable" ~ "horizontal variable"

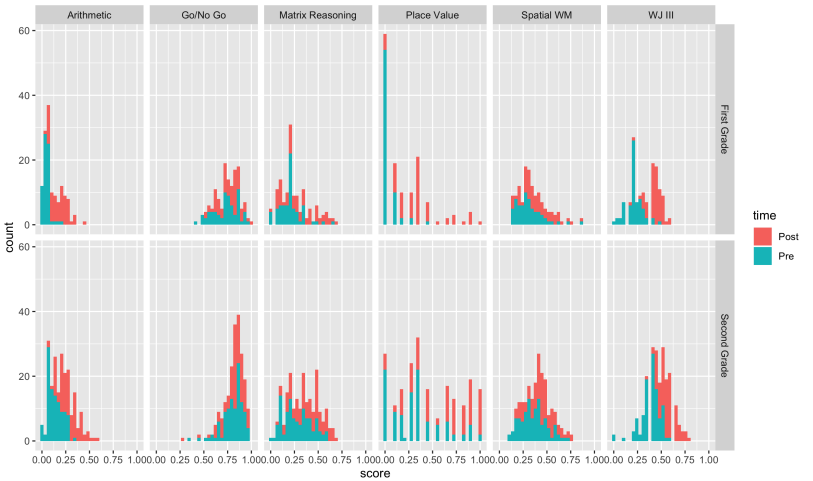
```
ggplot(abacus_data, aes(x = score)) +
  geom_histogram() +
  facet_grid(grade~task)
```



What we're really interested in here, though, is whether or not students got better at these tasks after over time.

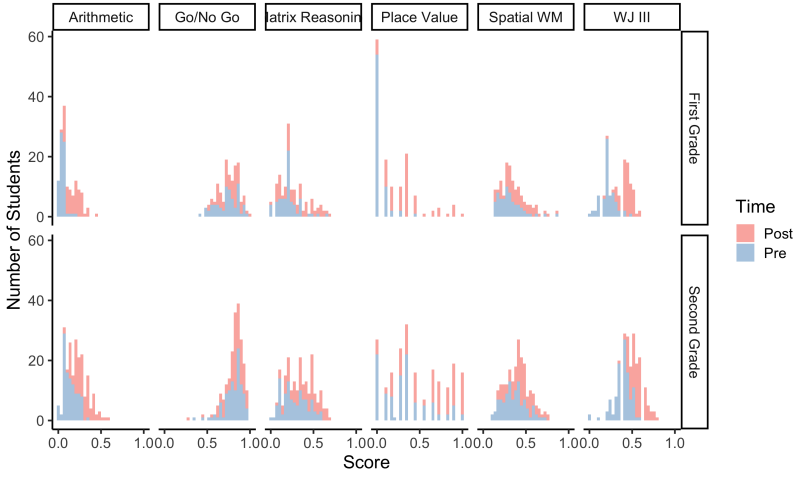
Let's use fill to show the pre and post distributions.

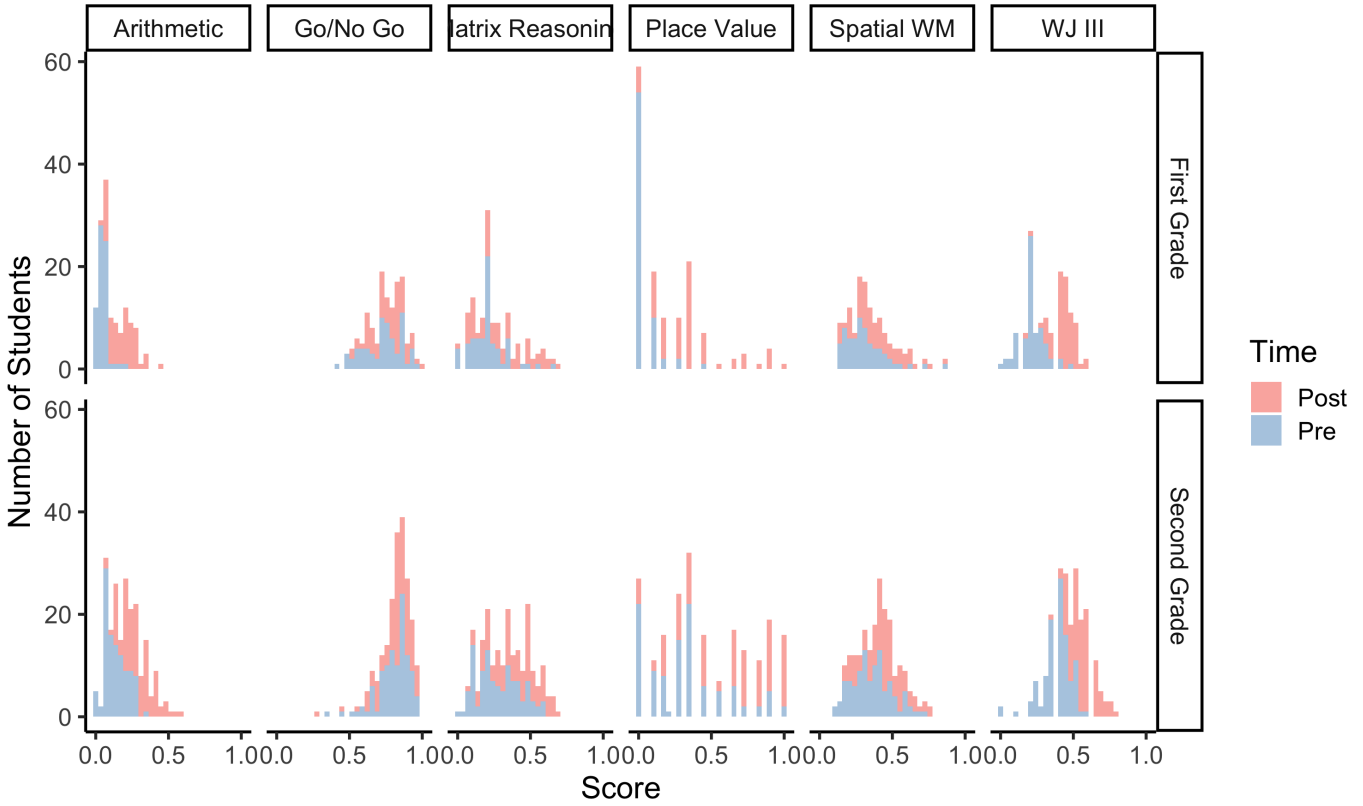
```
ggplot(abacus_data, aes(x = score, fill = time)) +
  geom_histogram() +
  facet_grid(grade~task)
```



Let's do a few more things to make our plot more attractive.

```
ggplot(abacus_data, aes(x = score, fill = time)) +
  geom_histogram(bindwidth = .1) +
  facet_grid(grade~task)  +
  scale_fill_brewer(name = "Time", palette = "Pastel1") +
  ylab("Number of Students") +
  xlab("Score") +
  scale_x_continuous(breaks = c(0, .5, 1)) +
  theme_classic(base_size = 16)
```




Arithmetic
Go/No Go
latrix Reasonin
Place Value
Spatial WM
WJ III
First Grade
Second Grade
Number of Students
Score
Time
Post
Pre

Do scores get better over time?

Let's add a vertical line to each distribution that shows the mean.

How do we calculate the mean of each distribution?

We want the mean by task, grade, and time.

We neet to **group_by** task, grade, and time, and then **summarize.**

```
distribution_means <- abacus_data %>%
  group_by(task, time, grade) %>%
  summarize(dist_mean = mean(score, na.rm=TRUE))
```

```
## # A tibble: 24 x 4
## # Groups:   task, time [12]
##    task             time  grade        dist_mean
##    <chr>            <chr> <chr>            <dbl>
##  1 Arithmetic       Post  First Grade     0.179
##  2 Arithmetic       Post  Second Grade    0.279
##  3 Arithmetic       Pre   First Grade     0.0448
##  4 Arithmetic       Pre   Second Grade    0.137
##  5 Go/No Go         Post  First Grade     0.762
##  6 Go/No Go         Post  Second Grade    0.820
##  7 Go/No Go         Pre   First Grade     0.730
##  8 Go/No Go         Pre   Second Grade    0.806
##  9 Matrix Reasoning Post  First Grade     0.305
## 10 Matrix Reasoning Post  Second Grade    0.392
## # … with 14 more rows
```

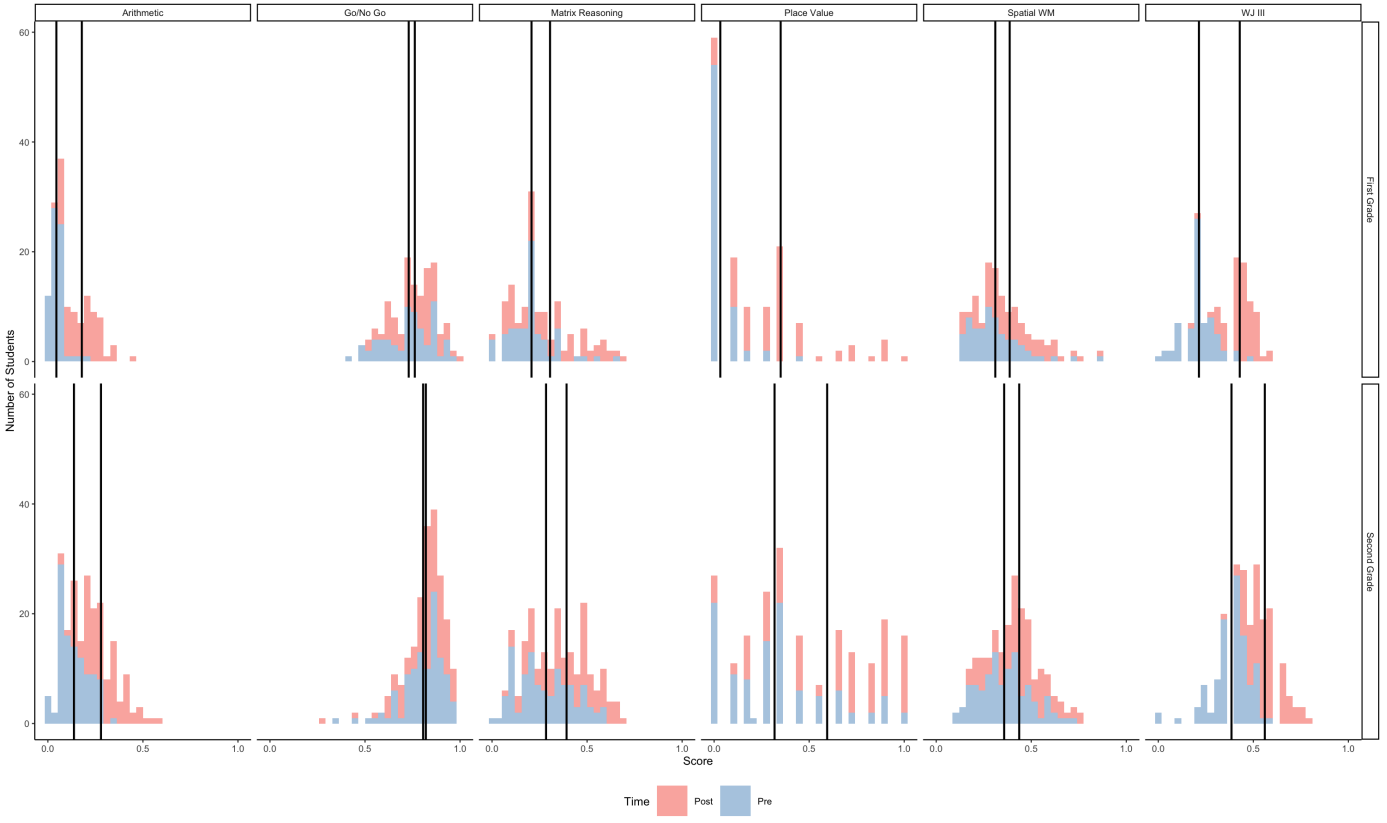
Now we have a new data frame with the means of each distribution.

How can we add them to our plot?

The geom for adding vertical lines is called `geom_vline()`

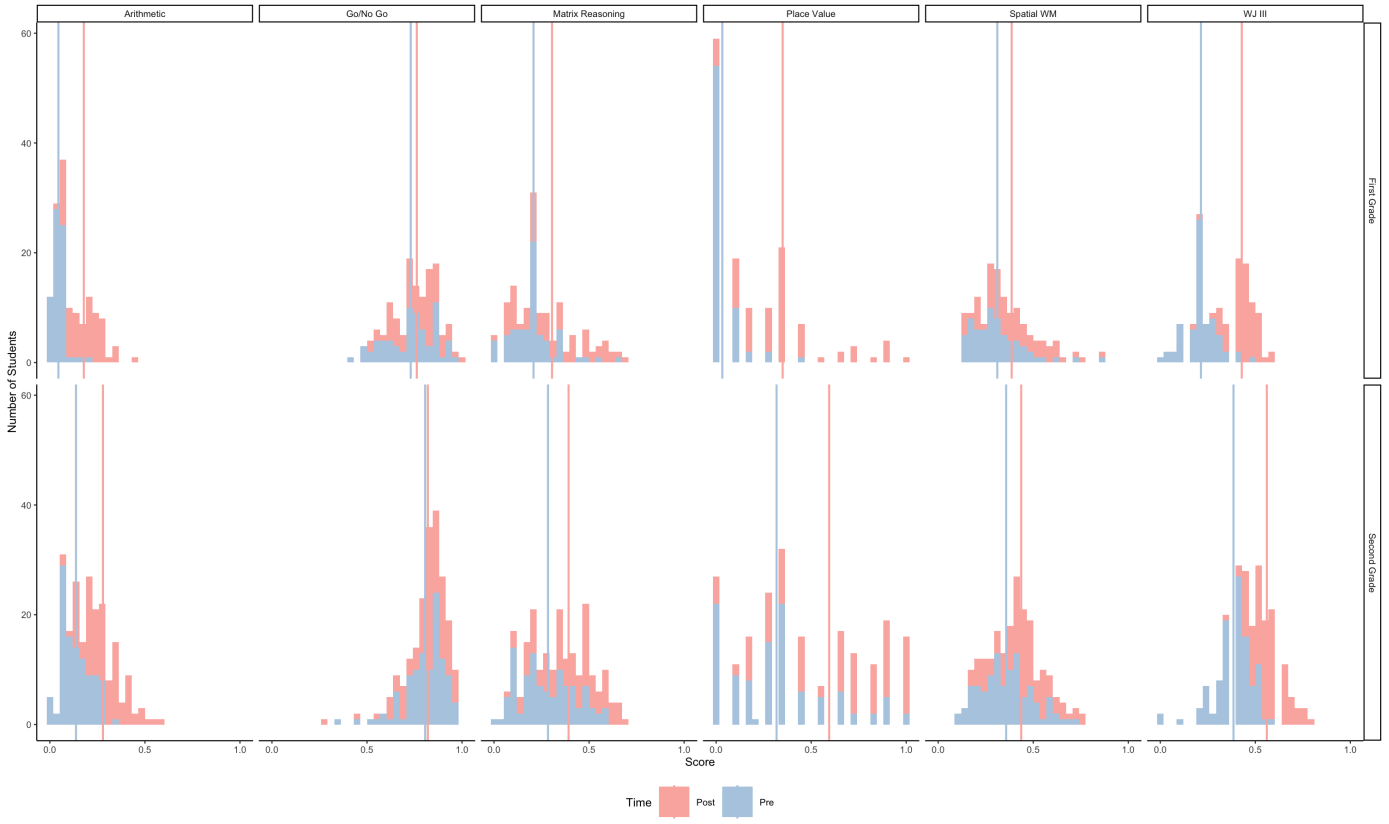
`geom_vline()` requires an aesthetic that tells it where to intersect the x-axis called **xintercept**.

```
ggplot(abacus_data, aes(x = score, fill = time)) +
  geom_histogram(bindwidth = .1) +
  geom_vline(data = distribution_means, aes(xintercept = dist_mean)) +
  facet_grid(grade~task)  +
  scale_fill_brewer(name = "Time", palette = "Pastel1") +
  ylab("Number of Students") +
  xlab("Score") +
  scale_x_continuous(breaks = c(0, .5, 1)) +
  theme_classic(base_size = 6)
```


Arithmetic
Go/No Go
Matrix Reasoning
Place Value
Spatial WM
WJ III
First Grade
Second Grade
Number of Students
Score
Time
Post
Pre

Let's color the lines to correspond to the distribution.

```
ggplot(abacus_data, aes(x = score, fill = time)) +
  geom_histogram(bindwidth = .1) +
  geom_vline(data = distribution_means, aes(xintercept = dist_mean, color = tim
  facet_grid(grade~task)  +
  scale_color_brewer(name = "Time", palette = "Pastel1", guide = F) +
  scale_fill_brewer(name = "Time", palette = "Pastel1") +
  ylab("Number of Students") +
  xlab("Score") +
  scale_x_continuous(breaks = c(0, .5, 1)) +
    theme_classic(base_size = 6) +
  theme(legend.position = "bottom")
```


Arithmetic
Go/No Go
Matrix Reasoning
Place Value
Spatial WM
WJ III
First Grade
Second Grade
Number of Students
Score
Time
Post
Pre

Okay, but the key question is: Did kids get better more when they got the abacus training?

Let's look at this just for the first grade group, and only the arithmetic task and spatial working memory tasks.

How could subset to these rows?

```
abacus_data
```

```
## # A tibble: 2,094 x 8
##    subid    class_num grade       group          year time  task        score
##    <chr>    <chr>     <chr>       <chr>         <dbl> <chr> <chr>       <dbl>
##  1 S1-02-02 S1_02     First Grade Control        2015 Pre   Place Value  0
##  2 S1-02-03 S1_02     First Grade Control        2015 Pre   Place Value  0
##  3 S1-02-03 S1_02     First Grade Control        2016 Post  Place Value  0.36
##  4 S1-02-08 S1_02     First Grade Control        2015 Pre   Place Value  0
##  5 S1-02-08 S1_02     First Grade Control        2016 Post  Place Value  0.36
##  6 S1-02-15 S1_02     First Grade Control        2016 Post  Place Value  0.64
##  7 S1-02-17 S1_02     First Grade Control        2015 Pre   Place Value  0.09
##  8 S1-02-17 S1_02     First Grade Control        2016 Post  Place Value NA
##  9 S1-03-04 S1_03     First Grade Mental Abacus  2016 Post  Place Value  0.55
## 10 S1-03-05 S1_03     First Grade Mental Abacus  2016 Post  Place Value  0.91
## # … with 2,084 more rows
```

```
first_grade_abacus_data <- abacus_data %>%
            filter(grade == "First Grade",
                   task %in% c("Spatial WM", "Arithmetic"))
```

```
first_grade_abacus_data
```

```
## # A tibble: 278 x 8
##    subid    class_num grade       group          year time  task       score
##    <chr>    <chr>     <chr>       <chr>         <dbl> <chr> <chr>      <dbl>
##  1 S1-02-02 S1_02     First Grade Control        2015 Pre   Arithmetic  0
##  2 S1-02-03 S1_02     First Grade Control        2015 Pre   Arithmetic  0
##  3 S1-02-03 S1_02     First Grade Control        2016 Post  Arithmetic  0.25
##  4 S1-02-08 S1_02     First Grade Control        2015 Pre   Arithmetic  0
##  5 S1-02-08 S1_02     First Grade Control        2016 Post  Arithmetic  0.08
##  6 S1-02-15 S1_02     First Grade Control        2016 Post  Arithmetic  0.1
##  7 S1-02-17 S1_02     First Grade Control        2015 Pre   Arithmetic  0.17
##  8 S1-02-17 S1_02     First Grade Control        2016 Post  Arithmetic NA
##  9 S1-03-04 S1_03     First Grade Mental Abacus  2016 Post  Arithmetic  0.17
## 10 S1-03-05 S1_03     First Grade Mental Abacus  2016 Post  Arithmetic  0.33
## # … with 268 more rows
```

```r
ms <- first_grade_abacus_data %>%
  group_by(group, task, time) %>%
  summarize(mean_score = mean(score, na.rm = T))

ggplot(first_grade_abacus_data, aes(x = score, fill = time)) +
  geom_histogram() +
  geom_vline(data = ms, aes(xintercept = mean_score, color = time), linetype =
  facet_grid(group~task) +
  scale_color_brewer(guide = "none", palette = "Pastel1") +

  scale_fill_brewer(name = "Time", palette = "Pastel1") +
  ylab("Number of Students") +
  xlab("Score") +
  scale_x_continuous(breaks = c(0, .5, 1)) +
  theme_classic(base_size = 12)
```

Does abacus training help?