That's enough recapping. Let's get started on today's meet!

```
library(SwimmeR)
library(purrr)
library(dplyr)
library(stringr)
library(flextable)
library(ggplot2)
```

My `flextable` styling function is still working great since I made it three weeks ago. I'm not changing it. I'm not even really changing this paragraph. Reusable code/content for life!

```
flextable_style <- function(x) {
  x %>%
    flextable() %>%
    bold(part = "header") %>% # bold header
    bg(bg = "#D3D3D3", part = "header") %>% # puts gray background behind the
header row
    autofit()
}
```

Getting Results

After the first round where we used `SwimmeR::read_results` and `SwimmeR::swim_parse` to get and clean our data respectively I kept it all, and stored it all on github. Let's get this thing underway by grabbing those results. First step is just to list out links to each data set.

```
CA_Link <-
  "https://raw.githubusercontent.com/gpilgrim2670/Pilgrim_Data/master/CA_States_2019.csv"

TX_Link <-
  "https://raw.githubusercontent.com/gpilgrim2670/Pilgrim_Data/master/TX_States_2020.csv"

FL_Link <- "https://raw.githubusercontent.com/gpilgrim2670/Pilgrim_Data/master/FL_States_2020.csv"

NY_Link <- "https://raw.githubusercontent.com/gpilgrim2670/Pilgrim_Data/master/NY_States_2020.csv"

PA_Link <- "https://raw.githubusercontent.com/gpilgrim2670/Pilgrim_Data/master/PA_States_2020.csv"

IL_Link <- "https://raw.githubusercontent.com/gpilgrim2670/Pilgrim_Data/master/IL_States_2020.csv"

OH_Link <- "https://raw.githubusercontent.com/gpilgrim2670/Pilgrim_Data/master/OH_States_2020.csv"

Links <- c(CA_Link, TX_Link, FL_Link, NY_Link, PA_Link, IL_Link, OH_Link)
```

Georgia of course is a special case because their results reporting is so bad.

```
GA_Link <- "https://raw.githubusercontent.com/gpilgrim2670/Pilgrim_Data/master/GA_States_2020.csv"
GA_Results <- read.csv(url(GA_Link)) %>%
  rename("Grade" = Age, Finals_Time = "Time") %>%
  mutate(State = "GA",
         Grade = as.character(Grade),
         Exhibition = 0,
         DQ = 0,
         Finals_Time = str_remove(Finals_Time, "^0"))
```

Now that we have our links we'll read them all in together by `map`ping a reading function across the list of links.

```
Read_Links <- function(x) {
  df <- read.csv(url(x)) %>% # reads each CSV file
    mutate(Grade = as.character(Grade)) # some states use FR/SO etc. for grade,
others use 9, 10 - need column to be all the same type
  return(df)
}

Results <- map(Links, Read_Links) %>%
  bind_rows(GA_Results) %>% # add in GA results
  mutate(Gender = case_when( # create gender column to score boys and girls meet
    str_detect(Event, "Girls") == TRUE ~ "Girls",
    str_detect(Event, "Boys") == TRUE ~ "Boys"
  )) %>%
  mutate( # slightly different event naming for some states
    Event = str_replace(Event, "Boys 1m Diving", "Boys 1 mtr Diving"),
    Event = str_replace(Event, "Girls 1m Diving", "Girls 1 mtr Diving")
  ) %>%
  filter(str_detect(Event, "AWD") == FALSE, # remove events outside the standard
12 each for boys and girls
         str_detect(Event, "Swim-off") == FALSE)
```

Scoring the Meet

We'll use `SwimmeR::results_score` to score each swim, then collect the scores to get overall totals for each state, by gender.

```
Results_Final <- results_score(
  results = Results,
  events = unique(Results$Event), # want to score all events
  meet_type = "timed_finals",
  lanes = 8,
  scoring_heats = 2,
  point_values = c(20, 17, 16, 15, 14, 13, 12, 11, 9, 7, 6, 5, 4, 3, 2, 1)
)

Scores <- Results_Final %>%
  group_by(State, Gender) %>% # score for each state and gender
  summarise(Score = sum(Points)) %>%
  group_by(Gender) %>%
  group_split() # new dataframe for boys and girls
```

Boys Scores

```
Scores[[1]] %>%
  arrange(desc(Score)) %>%
  mutate(Place = rank(desc(Score), ties.method = "min")) %>%
  select(Place, State, Score) %>%
  flextable_style()
```

Place State Score

1 CA 761.0

Place State Score

2 OH 462.0

3 PA 314.5

4 TX 250.0

5 IL 224.0

6 GA 221.5

7 FL 63.0

8 NY 29.0

Girls Scores

```
Scores[[2]] %>%
  arrange(desc(Score)) %>%
  mutate(Place = rank(desc(Score), ties.method = "min")) %>%
  select(Place, State, Score) %>%
  flextable_style()
```

Place State Score

1 CA 571.0

2 FL 422.5

3 TX 327.5

4 IL 317.5

5 OH 198.5

6 GA 176.0

7 PA 171.0

8 NY 141.0

Overall Scores

```
Combined_Scores <- Scores %>%
  bind_rows %>%
  group_by(State) %>%
  summarise(Score = sum(Score, na.rm = TRUE)) %>%
  arrange(desc(Score)) %>%
```

```
  mutate(Place = rank(desc(Score), ties.method = "min")) %>% # using rank here
is important because we actually do have a tie
  select(Place, State, Score)

Combined_Scores %>%
  flextable_style()
```

Place	State	Score
1	CA	1332.0
2	OH	660.5
3	TX	577.5
4	IL	541.5
5	FL	485.5
5	PA	485.5
7	GA	397.5
8	NY	170.0

California wins again, but the real story here is Ohio. Ohio was seeded 7th, but finished 2nd today. The heroes for Ohio were the boys from Cincinnati St Xavier, who won all three relays and scored 120 points in doing so.

```
Results_Final %>%
  filter(is.na(Name) == TRUE,
         School == "Cincinnati St Xavier") %>%
  select(Place, School, Finals_Time, Event, Points) %>%
  flextable_style()
```

Place	School	Finals_Time	Event	Points
1	Cincinnati St Xavier	1:20.86	Boys 200 Yard Freestyle Relay	40
1	Cincinnati St Xavier	1:27.97	Boys 200 Yard Medley Relay	40
1	Cincinnati St Xavier	2:59.30	Boys 400 Yard Freestyle Relay	40

Testing Seeding Assumptions

Original Assumption

Way back in July, when the State-Off started I made the decision to seed states based on their population. The assumption, which seemed reasonable at the time, was that states with larger populations would have a larger population pool (heh heh) from which to draw potential swimmers, and thus would have more, faster, swimmers than less populous states. Seeding by population made sense, with more populous states

expected to defeat less populous ones.

```
Pop_Data <- read.csv("http://www2.census.gov/programs-surveys/popest/datasets/2010-2019/national/totals/nst-est2019-alldata.csv?#") # download data
Seeds <- Pop_Data %>%
  mutate(STATE = as.numeric(STATE)) %>%
  filter(STATE >= 1) %>%
  select(NAME, POPESTIMATE2019) %>% # get populations
  arrange(desc(POPESTIMATE2019)) %>%
  top_n(8) %>% # cut down to top 8 most populated states
  mutate(Seed = 1:n(),
         POPESTIMATE2019 = round(POPESTIMATE2019 / 1000000, 2)) %>%
  select(Seed, "State" = NAME, "Population_mil" = POPESTIMATE2019) %>%
  mutate(State = state.abb[match(State,state.name)]) # convert names (California) to abbreviations (CA)

Seeds %>%
  flextable_style()
```

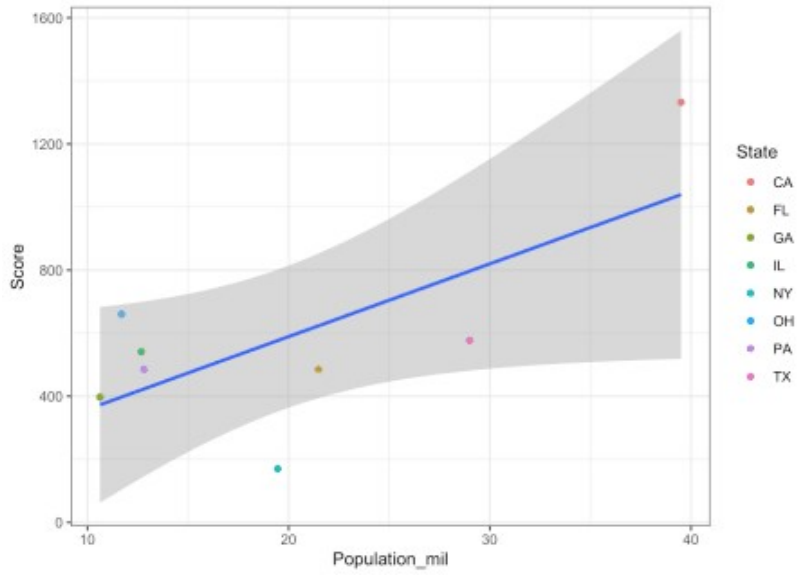
Seed	State	Population_mil
1	CA	39.51
2	TX	29.00
3	FL	21.48
4	NY	19.45
5	PA	12.80
6	IL	12.67
7	OH	11.69
8	GA	10.62

Actual Results

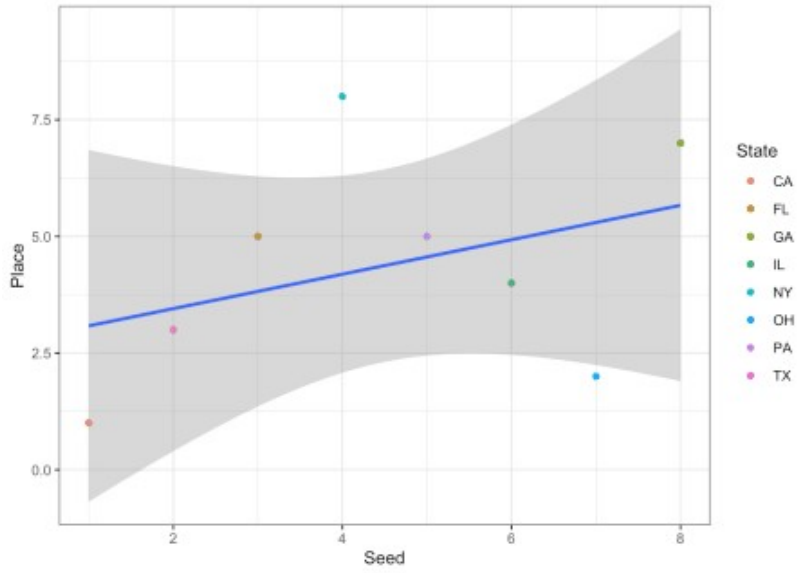
In the head-to-head matchups that followed my original assumption was largely born out. There was only one upset, when 5th seeded Pennsylvania upset 4th seeded New York. Otherwise more populous states did defeat less populous ones. Today's results however don't show the same trend between population and score, or seed and place.

```
Score_Pop <- Combined_Scores %>%
  left_join(Seeds, by = "State")

Score_Pop %>%
  ggplot(aes(x = Population_mil, y = Score)) +
  geom_point(aes(color = State)) +
  geom_smooth(method = "lm") + # add linear fit line, with standard error
  theme_bw()
```




```
Score_Pop %>%
  ggplot(aes(x = Seed, y = Place)) +
  geom_point(aes(color = State)) +
  geom_smooth(method = "lm") + # add linear fit line, with standard error
  theme_bw()
```



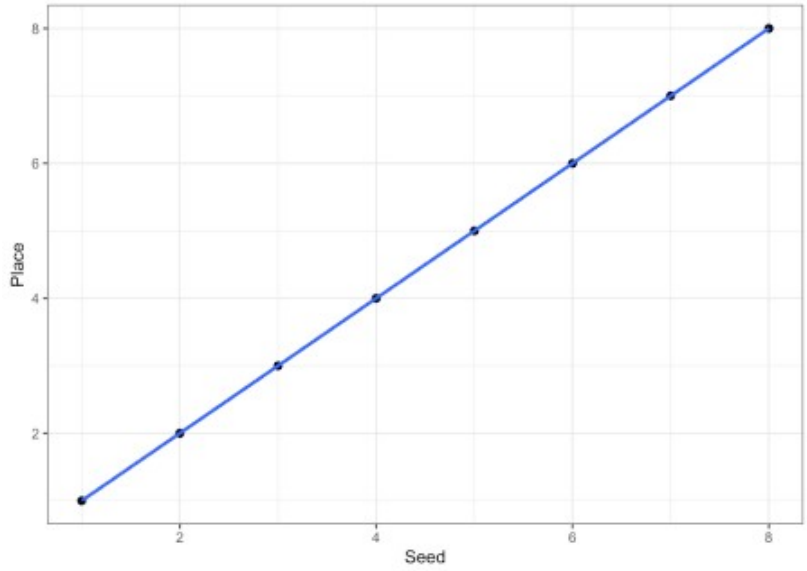
There doesn't look to be any particular relationship between either population and score, or seed and actual order of finish (place). Population vs. score looks more linear, but I suspect that's just because California is so far in the upper right corner that it's just dragging the fit out. We can, should, and in fact will, do some testing to confirm though. One possible relationship between to variables is a linear relationship. There are of course other relationships, but let's focus on linearity for a moment.

To be clear, my original assumption when I devised the seeding wasn't that population and score would be linearly related – it was more general. I assumed that score would increase as population increased. The increase could be linear, but could also be exponential, cubic, etc., any of which would have justified population as a seeding mechanism. Built into that original assumption though was a second assumption, which is linear. I assigned seeds based on population, and then earlier in this post we assigned places based on score. Seed and place are in effect ordinal values corresponding to the numeric values of population and score respectively, and if score does increase with population than seed and place should have a linear relationship. We can see such a relationship below. The fit line from `geom_smooth` intersects each point perfectly. Standard error is also enabled, as in the plots above, but doesn't show up, because there is no error. The linear fit is literally perfect.

```
df <- data.frame(Seed = seq(1, 8, 1),
```

```
                  Place = seq(1, 8, 1))

df %>%
  ggplot(aes(x = Seed, y = Place)) +
  geom_point(size = 2) +
  geom_smooth(method = "lm") +
  theme_bw()
```



Correlation Testing

I bring all this up because R has the built in function `cor.test`, which we can use to test for linear correlation. In addition to arguments specifying each of the two variables under test, `cor.test` also has an argument called `method`, used to specify the exact statistical test used. The two tests we'll consider here are Pearson's (`method = "pearson"`, the default) and Spearman's (`method = "spearman"`). Pearson's test assumes that
1. There is a linear relationship between the two variables. Again, this is the case for seed/place, but need not be for population/score
2. The variables are normally distributed. Let's test for this.

```
shapiro.test(Score_Pop$Score)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Score_Pop$Score
## W = 0.81856, p-value = 0.04507
```

```
shapiro.test(Score_Pop$Population_mil)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Score_Pop$Population_mil
## W = 0.85247, p-value = 0.1009
```

The null assumption for `shapiro.test` is that the variable *is* normally distributed. We see that for

`Score_Pop$Score` and `Score_Pop$Population_mil` the p-values are 0.045 and 0.101 respectively. The value for `Score` is less than 0.05, so we can reject the null hypothesis and conclude `Score` is not normally distributed. Between that and the need-not-be-linear relationship between population and score Pearson's test is out.

That leaves Spearman's. Spearman's test does not require variables to be normally distributed, and it doesn't require a the assumption of a linear relationship between variables. This is because Spearman's uses a rank method, ordering each variable, assigning a rank, and checking for correlation between those ranks. This sounds a lot like what we've already done, ranking population and assigning seed, and ranking score and assigning place. The null hypothesis for Spearman's test is that there is *not* a linear relationship between the two variables.

```
Spearman_Place_Seed <- cor.test(Score_Pop$Place, Score_Pop$Seed, method =
"spearman")
Spearman_Place_Seed
```

```
##
##  Spearman's rank correlation rho
##
## data:  Score_Pop$Place and Score_Pop$Seed
## S = 53.82, p-value = 0.3821
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.3592879
```

```
Spearman_Score_Pop <- cor.test(Score_Pop$Score, Score_Pop$Population_mil, method
= "spearman")
Spearman_Score_Pop
```

```
##
##  Spearman's rank correlation rho
##
## data:  Score_Pop$Score and Score_Pop$Population_mil
## S = 53.82, p-value = 0.3821
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.3592879
```

For both tests our p-value is 0.382, greater than 0.05. We cannot reject the null hypothesis, and so we conclude there is not a linear relationship between seed and place, or between population and score, in today's meet. Not only are our p-values the same for each test, Our Spearman's test results are in fact exactly the same, which we can confirm. First, let's look at the structure:

```
str(Spearman_Place_Seed)
```

```
## List of 8
##  $ statistic  : Named num 53.8
##   ..- attr(*, "names")= chr "S"
##  $ parameter  : NULL
##  $ p.value    : num 0.382
##  $ estimate   : Named num 0.359
```

```
##   ..- attr(*, "names")= chr "rho"
##  $ null.value : Named num 0
##   ..- attr(*, "names")= chr "rho"
##  $ alternative: chr "two.sided"
##  $ method     : chr "Spearman's rank correlation rho"
##  $ data.name  : chr "Score_Pop$Place and Score_Pop$Seed"
##  - attr(*, "class")= chr "htest"
```

Each result has 8 elements, the last of which are the variable names. These will of course be different between tests, because we're testing different variables. Everything else though is either a result of the test, or an input to the test. We can thus compare everything except the 8th element and expect them to be `identical`.

```
identical(Spearman_Place_Seed[1:7], Spearman_Score_Pop[1:7])
```

```
## [1] TRUE
```

The test results are identical because the Spearman's test does to population and score what we already did in assigning seed and place. As for the test results, correlations are of course not the same as causations. Even if population and score had been correlated for today's meet we wouldn't be able to prove that higher population actually causes faster swims and higher scores. For today's meet population proves to not have a linear relationship with score, nor does another relationship readily present itself.

Swimmers of the Meet

Swimmer of the Meet criteria is still the same as it's been for the entire State-Off. We'll look for athletes who have won two events, thereby scoring a the maximum possible forty points. In the event of a tie, where multiple athletes win two events, we'll use All-American standards as a tiebreaker.

```
Cuts_Link <-
  "https://raw.githubusercontent.com/gpilgrim2670/Pilgrim_Data/master/State_Cuts.csv"
Cuts <- read.csv(url(Cuts_Link))

Cuts <- Cuts %>% # clean up Cuts
  filter(Stroke %!in% c("MR", "FR", "11 Dives")) %>% # %!in% is now included in
SwimmeR
  rename(Gender = Sex) %>%
  mutate(
    Event = case_when((Distance == 200 & #match events
                         Stroke == 'Free') ~ "200 Yard Freestyle",
                      (Distance == 200 &
                         Stroke == 'IM') ~ "200 Yard IM",
                      (Distance == 50 &
                         Stroke == 'Free') ~ "50 Yard Freestyle",
                      (Distance == 100 &
                         Stroke == 'Fly') ~ "100 Yard Butterfly",
                      (Distance == 100 &
                         Stroke == 'Free') ~ "100 Yard Freestyle",
                      (Distance == 500 &
                         Stroke == 'Free') ~ "500 Yard Freestyle",
                      (Distance == 100 &
                         Stroke == 'Back') ~ "100 Yard Backstroke",
                      (Distance == 100 &
                         Stroke == 'Breast') ~ "100 Yard Breaststroke",
                      TRUE ~ paste(Distance, "Yard", Stroke, sep = " ")
    ),
```

```
    Event = case_when(
      Gender == "M" ~ paste("Boys", Event, sep = " "),
      Gender == "F" ~ paste("Girls", Event, sep = " ")
    )
  )

Ind_Swimming_Results <- Results_Final %>%
  filter(str_detect(Event, "Diving|Relay") == FALSE) %>% # join
Ind_Swimming_Results and Cuts
  left_join(Cuts %>% filter((Gender == "M" &
                               Year == 2020) |
                              (Gender == "F" &
                                 Year == 2019)) %>%
              select(AAC_Cut, AA_Cut, Event),
            by = 'Event')

Swimmer_Of_Meet <- Ind_Swimming_Results %>%
  mutate(
    AA_Diff = (Finals_Time_sec - sec_format(AA_Cut)) / sec_format(AA_Cut),
    Name = str_to_title(Name)
  ) %>%
  group_by(Name) %>%
  filter(n() == 2) %>% # get swimmers that competed in two events
  summarise(
    Avg_Place = sum(Place) / 2,
    AA_Diff_Avg = round(mean(AA_Diff, na.rm = TRUE), 3),
    Gender = unique(Gender),
    State = unique(State)
  ) %>%
  arrange(Avg_Place, AA_Diff_Avg) %>%
  group_split(Gender) # split out a dataframe for boys (1) and girls (2)
```

Boys

```
Swimmer_Of_Meet[[1]] %>%
  slice_head(n = 5) %>%
  select(-Gender) %>%
  ungroup() %>%
  flextable_style()
```

Name	Avg_Place	AA_Diff_Avg	State
Magahey, Jake	1.0	-0.042	GA
Brownstead, Matt	1.5	-0.050	PA
Aikins, Jack	1.5	-0.038	GA
Hu, Ethan	2.0	-0.052	CA
Dillard, Ben	2.5	-0.044	CA

Jake Magahey is a fresh face imn the Swimmer of the Meet arena. He didn’t win any Swimmer of the Meet

crowns during the State-Off Rounds. He wasn't ever even runner up. Here he is though, top of the heap, in the State-Off Battle Royale. Maybe we didn't predict it, but that's why they play the games. On the other hand he was tied for first in the CollegeSwimming 2020 cycle rankings so maybe we could have predicted it after all. Congratulations to Jake!

```
Results_Final %>%
  filter(Name == "Magahey, Jake") %>%
  select(Place, Name, School, Finals_Time, Event) %>%
  arrange(desc(Event)) %>%
  ungroup() %>%
  flextable_style()
```

Place	Name	School	Finals_Time	Event
1	Magahey, Jake	MLCR	4:16.5	Boys 500 Yard Freestyle
1	Magahey, Jake	MLCR	1:34.4	Boys 200 Yard Freestyle

Girls

```
Swimmer_Of_Meet[[2]] %>%
  slice_head(n = 5) %>%
  select(-Gender) %>%
  ungroup() %>%
  flextable() %>%
  bold(part = "header") %>%
  bg(bg = "#D3D3D3", part = "header") %>%
  autofit()
```

Name	Avg_Place	AA_Diff_Avg	State
Hartman, Zoie	1.0	-0.047	CA
Lillie Nordmann	1.0	-0.046	TX
Cronk, Micayla	1.5	-0.040	FL
Weyant, Emma	1.5	-0.033	FL
Kit Kat Zenick	2.0	-0.029	TX

Meet the new boss, same as the old boss. Zoie Hartman is well known around these parts, as she swept all the Swimmer of the Meet crowns in the standard tournament. Today is no different, in fact it's a replay of the California vs. Texas championship meet where Zoie edged out Lillie Nordmann. Congratulations again to Zoie for an extremely successful State-Off run!

```
Results_Final %>%
  filter(Name == "Hartman, Zoie") %>%
  select(Place, Name, School, Finals_Time, Event) %>%
  arrange(desc(Event)) %>%
  ungroup() %>%
  flextable_style()
```

Place	Name	School	Finals_Time	Event
1	Hartman, Zoie	Monte Vista_NCS	1:55.29	Girls 200 Yard IM
1	Hartman, Zoie	Monte Vista_NCS	59.92	Girls 100 Yard Breaststroke

In Closing

This was the last meet, and last post, in the State-Off Tournament. It's been a great ride. I'm not going to say it was as much fun as the Olympics, but at least it was something for these swimming-less months. Hopefully you've had some fun, and maybe learned a bit along the way – I know I have! Next week we'll be switching gears and talking about diving math here at Swimming + Data Science. We'll see you then!