Lane Placement and Performance:
A Causal Exploration

Team #1
Noah Becker, Qian Fu, Jiahui Jiang, Yassine Manane, Xue Ni
University of Minnesota - Carlson School of Management
March 1st, 2020

Table of Contents

Executive Summary 3

Context 3

Causal Question 3

Analysis Performed 3

Main Takeaways 3

Introduction 4

Data Overview 6

Main Analysis 8

Naïve Regression 8

Propensity Score Matching 10

Conclusion 12

Policy Implications 12

Limitations 13

References 14

Appendix 15

A: Python Script 15

B: R Script 17

C: Regression Output 20

D: Event Scraping 21

EXECUTIVE SUMMARY

Context:

In competitive swimming, lane placement is determined either by seed time from past races or time in preliminary races. The fastest competitors are assigned lanes closest to the middle of the pool while each lane away from the center is subsequently slower. There is great debate amongst the swimming community whether there is a performance benefit to being placed in an inside lane.

Causal Question:

Using an observational study of championship swim meets, we hope to assess and estimate whether lane placement has a causal effect upon race performance.

Analysis Performed:

First, with raw competition data, we used linear regression to model time in finals based on prelims time, gender, stroke, time improvement from seed to prelims, and lane placement. To better simulate a randomized controlled trial, we used an econometric technique called propensity score matching. Next, we reapplied a linear regression model to the matched dataset and estimated the effect of lane placement upon performance. Various sensitivity analyses were performed to assess the robustness of our findings.

Main Takeaways:

A. We estimate that swimming in an outside lane for a 100 yard competition results in a significant .3342 second increase in time as compared to swimming in an inside lane.
B. Sensitivity analysis confirms that these findings are robust.

INTRODUCTION

When watching competitive swimming, it seems like the swimmers in the middle lanes always win the race. Although this is partly by design, as the fastest qualifiers are placed in the innermost lanes, there is great debate amongst the swimming world whether there is additional advantage unintentionally given to these swimmers. In this paper, we hope to address whether lane placement leads to an unfair competitive advantage. To illustrate this notion, take the example of Michael Phelps, widely regarded as the greatest swimmer of all time. In the 2012 Olympic Games, Phelps barely qualified for the finals of the 400 IM. As the slowest qualifier, he was placed in one of the outermost positions, lane 1. Unphased, Phelps didn't think much of it. "The only thing that matters is getting a spot" he rationalized, "you can't get a gold medal from the morning." In retrospect, it's understandable that Phelps was this confident despite having the slowest prelims time amongst qualifiers; he was, after all, the reigning gold medalist and world record holder in the event. However, he was not able to defend his title when swimming from an outside lane. As a matter of fact, he failed to medal at all, placing 4^{th} in the event and adding over 5 seconds to his time from the prior Olympics. To put this in perspective, out of the 30 Olympic events that Phelps swam, the only other time he didn't medal was at the age of 15.

Although this example gives some credence to the claim that lane placement may affect performance, it is quite anecdotal. To determine if similar research has been done, we turned to the internet. In 2016, Wall Street Journal, Washington Post, and Quartz all published articles detailing quantitative research on lane advantages in swimming. Unfortunately, these studies were all specific to the Olympic pool in Rio de Janeiro regarding design concerns. As our interest lies with competitive swimming in general, our question is left unanswered. A more generalized article on

the subject was published by BBC, however, it is entirely theoretical. In our study, we will use real competition data to assess the effect of lane placement on performance.

Before we begin our analysis, it is important to establish some intuition as to why an effect may be plausible. First, there are physical reasons. Swimming in an outside lane typically means swimming with a wall directly to one side of you. When waves form, they crash off these walls and can cause turbulence or currents in the water with the greatest impact on those closest to the wall. Next, there are psychological factors at play. Being placed in a center lane implies that you are a favorite to win the heat whereas swimmers in the outside lanes are expected to finish last. Mentally, this could go two ways. In an inside lane, there may be extra motivation to win the heat as a top qualifier, but at the same time, there may be additional pressure and expectations. Conversely, from an outside lane, it may be discouraging that you are an underdog but there may not be the same pressure or expectations that a favorite has. Lastly, there are visual reasons that lane advantages could exist. Because they are swimming against a wall, swimmers in outside lanes do not have a clear view of their competition. As a matter of fact, they can only see competitors to one side of them and are multiple lanes away from the top seeded swimmers. In contrast, swimmers in the inside lanes can see other top competitors right next to them on either side. It's possible that this could affect pacing significantly.

In order to determine the effect of lane placement conclusively, we would ideally conduct an experiment. This would involve gathering a sufficiently large group of swimmers, randomly assigning them to inside or outside lanes, subjecting both groups to the exact same preparation (e.g. bedtime, wake-up time, meals, warm-up, etc.), and having the swimmers race, recording the times. Then, results of the swimmers in inside and outside lanes could be compared to estimate an

effect. Alas, we had neither the time nor the resources to conduct a proper experiment. Instead, we will be performing an observational study.

DATA OVERVIEW

The data used for our analysis comprises of meet results from conference championship meets in the California Community College Athletic Association between 2017 and 2019, scraped from SwimPhone.com. Each meet included in our dataset took place at the end of the season in either April or May, was swam in an 8 lane pool, and follows the same championship meet format with prelims of an event in the morning and the top 16 qualifiers racing again in the evening for finals. To simplify our analysis, we restrict the scope to events of 100 yards, the most common distance. Each unit of observation is a finalist in a specific event. The dependent variable we will be modeling is the finals time while the treatment is the lane placement in finals. Lanes 4 and 5 are considered inside lanes while lanes 1 and 8 are considered outside lanes. Other covariates included are the prelims time, the improvement from seed time to prelims time, the gender of the swimmer, and the stroke of the event. The Python script used to obtain this data is included in Appendix A/D.

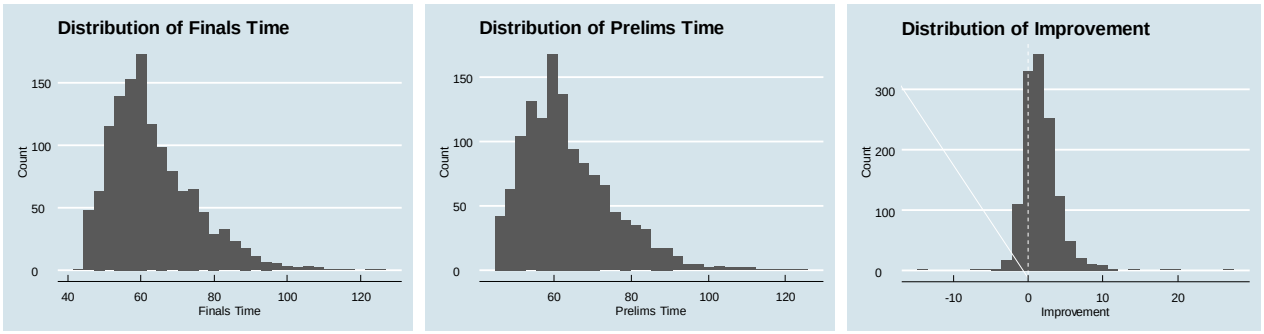

Figure 1: Continuous Data Distributions

When examining the distributions of the time variables (Figure 1), there are several characteristics of note. It appears that both prelims time and finals time follow similar right skewed distributions.

Most of the data is around 60 seconds with a few observations reaching 80 or even over 100 seconds but no extreme outliers. Additionally, the finals time seems to be slightly more concentrated at lower values. The time improvement from seed to prelims is centered above 0, indicating that most finalists improved upon their in-season seed time in prelims. This distribution is fairly symmetric with some outliers gaining more than 10 seconds or improving by over 20 seconds.

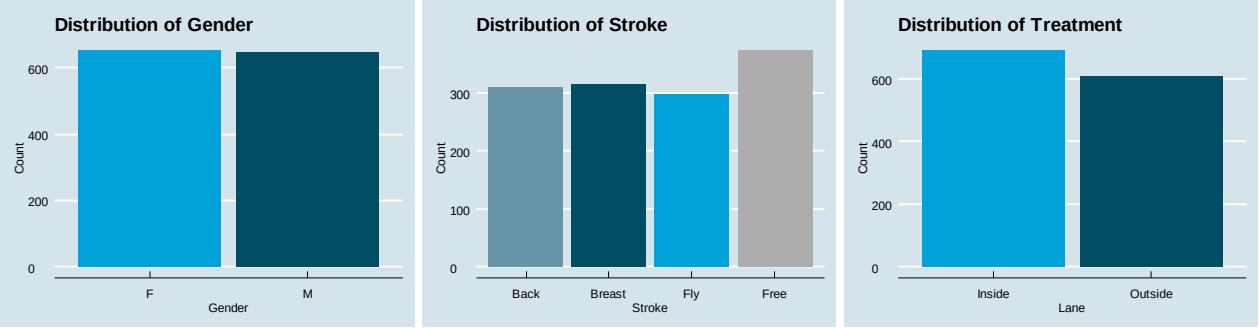

Figure 2: Discrete Data Distributions

Looking at the distributions of categorical variables (Figure 2), we see some interesting characteristics. As the data collected consists of the 100 yard races of each stroke across 22 meets, it is unsurprising that gender is quite evenly distributed. However, the distributions of stroke and lane are somewhat unexpected. Because there was an equal number of events scraped for each stroke, we would expect a balanced number of observations for each stroke. The dominance of freestyle in our dataset implies that swimmers were more likely to scratch out of finals in non-free strokes or some events did not have enough participants to fill two finals heats. Similar rationale might explain why our dataset is weighted more heavily towards inside lanes. Those who are expected to finish last in their heat may be more likely to scratch the event or, since lanes are seeded progressively slower towards the outside, events without enough participants for two full heats would result in empty outside lanes.

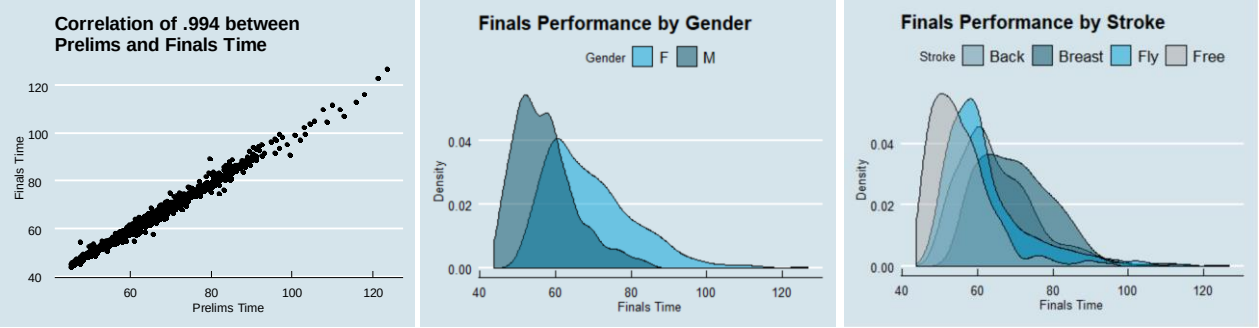

Figure 3: Relationships Amongst Variables

To better understand the variables that will be used in our analysis, we examined some relationships present (Figure 3). Intuitively, a swimmer's performance in prelims is a very strong indicator of his or her performance in finals, as indicated by a nearly perfect correlation between the two times. In addition, we see that male swimmers' times are typically faster and less varied than those of female swimmers. In terms of strokes, it's evident that freestyle is the fastest stroke with the least variance, followed by butterfly, backstroke, and breaststroke, in that order. Now that we have a clearer picture of the data that we will be using, we can conduct our analyses.

MAIN ANALYSIS

Naïve Regression:

Before applying any econometric techniques to make causal inference in an observational setting, we ran a naïve regression model regressing finals time on prelims time, stroke, gender, time improvement, and lane. However, as discussed previously, lane placement for finals is not randomly assigned and there is clear endogeneity present. Therefore, we are unable to imply causality in our interpretation of the treatment effect. Results of this regression are shown below (Table 1). To be concise, insignificant predictors are excluded from this discussion. Full regression output is included in Appendix C.

Variable	Estimate	p-value
Prelims Time	.9847	< 2e-16
Improvement	.1447	< 2e-16
Lane (Treatment)	**.2830**	**.000215**

Table 1: Naïve Regression Results

We observe that for a one second increase in a swimmer's prelims time, we would expect his or her finals time to increase by .9847 seconds, holding all other variables constant. Unsurprisingly, this indicates that swimmers are typically faster in finals heats than in prelims heats. Next, we can see that for each additional second of improvement between seed time and prelims time, we expect an increase in finals time by .1447 seconds, holding all else constant. Again, this makes sense. Swimmers who have improved significantly from their seed time likely do not have more effort to give for further improvement, whereas those who did not improve greatly from their in-season time may have more to give in finals. Next, we observe that swimmers in outside lanes swim, on average, .2830 seconds slower in finals than those in inside lanes, all else equal.

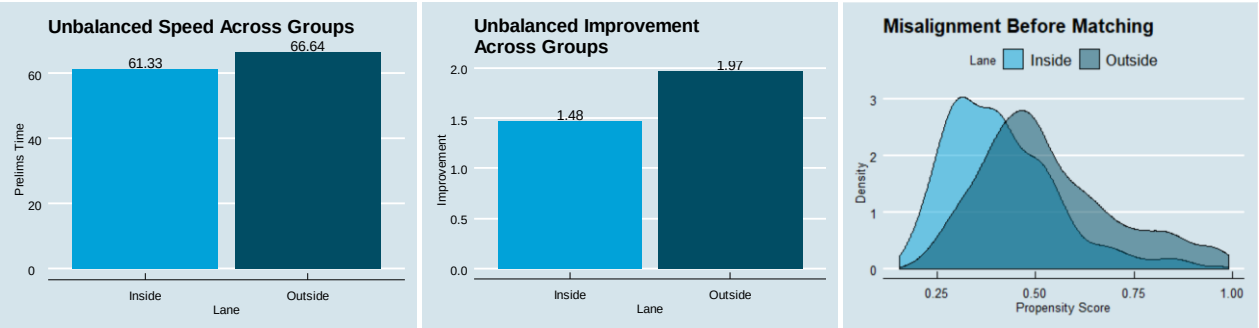

Figure 4: Covariate Imbalance Before Matching

As alluded to before, since treatment groups are not randomly assigned, we see significant differences between the two groups (Figure 4). Because lane placement is determined by prelims time, we see that swimmers in inside lanes are an average of around 5 seconds faster than those in outside lanes. Additionally, those in outside lanes had an average improvement in prelims of about half a second more than those in inside lanes. Last but not least, if the groups were truly similar, they should have a nearly identical distribution of propensity scores. A propensity score is a measure of the predicted probability of being placed in an outside lane based on all other factors. Intuitively, it can be thought of as a measure encompassing all covariates such that two observations with the same propensity score would have virtually the same characteristics. As we can see in the graphs above, it's clear that this observational data cannot be treated like a randomized controlled trial.

Propensity Score Matching:

To address this non-random lane assignment, we employed an econometric technique called propensity score matching. This method finds swimmers in outside lanes with very similar propensity scores to swimmers placed in inside lanes and matches them. Those without a good match are discarded. The resulting dataset should eliminate covariate imbalance and recover treatment assignment as good as random, post hoc.

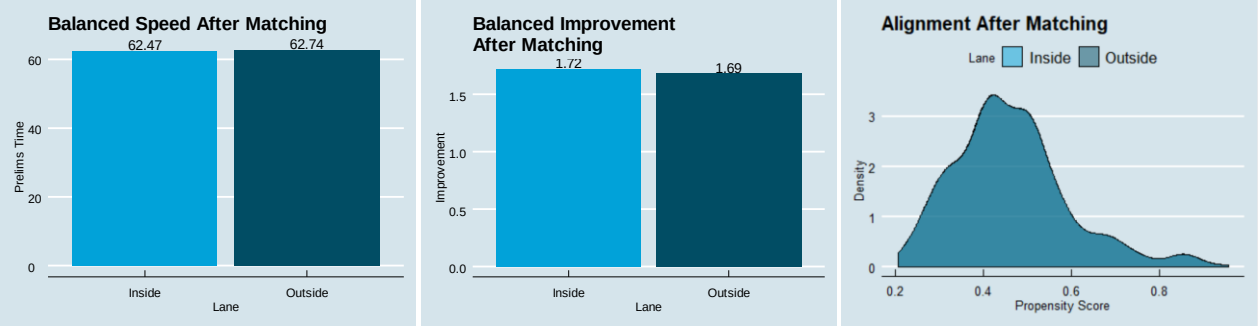

Figure 5: Covariate Balance After Matching

After matching, it is evident that the characteristics of swimmers in outside and inside lanes are much more similar. Statistical tests confirm that there are not significant differences between the two groups in prelims time or time improvement. Moreover, we can see that the distribution of propensity scores is essential identical between those placed in inside or outside lanes. Now that we have obtained a dataset that simulates a randomly assigned treatment, we can refit a regression model of finals time based on prelims time, stroke, gender, time improvement, and lane placement. Results of this model are shown below (Table 2). Again, we omit the estimates of insignificant regressors in the interest of brevity, but full regression output is included in Appendix C.

Variable	Estimate	p-value
Prelims Time	.9945	< 2e-16
Improvement	.1347	1.25e-11
Lane (Treatment)	**.3342**	**.000114**

Table 2: Matched Regression Results

The significant predictors and estimates are quite similar to those before matching. Here, we see that for each additional second of time in prelims, we expect the time in finals to increase by .9945 seconds, holding all other variables constant. Again, we observe an expected increase in finals time of .1347 seconds for each incremental second of improvement from seed to prelims while controlling other predictors. Finally, we are able to estimate the effect of lane placement. All else equal, we estimate that swimming in an outside lane increases time by .3342 seconds. For reference, the difference between 1st place and 4th place of the men's 100 meter freestyle in the 2016 Olympics was .30 seconds. Full code for implementing these analyses is included in Appendix B.

CONCLUSION

Policy Implications:

Moving forward, there are a few things that can be done to level the playing field. Recall the three reasons we may see an effect of lane placement upon performance (physical, psychological, visual). Barring a radical redesign of the competition pool, we can only address two of these potential reasons: physical and mental factors.

To combat the physical disadvantages of swimming in an outside lane, we first recommend ensuring that high level competitions have buffer lanes on the outside. This is already done in many top competitions. By having an extra lane against the wall, competitors will no longer be subject to waves crashing against the wall. In addition, there are a couple pool design characteristics that should be considered. There are gutters and lane lines that are specifically designed to disperse turbulence in the water. Investment in these technologies can help ensure that nobody is given an unfair advantage. Also, pools of greater depth reduce the likelihood of currents forming. This should be taken into consideration when building new competition pools. To address psychological factors that impair swimmers in the outside lanes, we propose random assignment of lanes to finalists. This could help eliminate the stigma and expectation of losing the race simply because of lane placement.

There are also implications of our findings from a coaching and swimming perspective in the status quo. As a coach, it may be beneficial to educate swimmers of this effect and try to eliminate negative psychological effects of lane placement. As a swimmer, it's important to realize that there is a significant performance benefit to being placed in inside lanes. Therefore, to maximize perfor-

mance in finals, it's necessary to give enough effort in prelims to be rewarded with the competitive advantage of an inside lane in finals.

Limitations:

Our analysis and findings are not without limitations, however. There are several threats to causal inference that may bias our estimates. First, there is significant selection bias as we use a convenience sample. Since we are simply using data that was easily accessible, we do not have a random sample and it may not necessarily be representative of all competitive swimmers. Because of this, our findings may not generalize. Next, the key assumption of matching is that lane placement is fully determined by observed variables. Although lane placement is directly based on prelims time, there are a number of omitted variables that may affect both finals time and treatment. Unobservable characteristics such as energy level, motivation, and strategic decisions may all influence performance in prelims (and therefore, lane placement) and finals. Another variable we do not capture is the quality of competition swimsuit. Swimmers who are confident they will qualify for finals may save a faster suit for finals, biasing their speed in prelims and lane placement. Lastly, amount of competitive experience and age may be related to lane placement and performance.

From a more technical perspective, the functional form of logistic regression used for matching is quite arbitrary. However, sensitivity analyses varying functional form, matching algorithm, parameter values, and model specifications show our results are quite robust. An R script of this sensitivity analysis is included in Appendix B. Finally, we are unable to account for differing lanes in the prelims swim. When including this as a covariate, we would essentially be estimating the treatment effect of lane placement twice. This would lead to a difficult to interpret and fairly unreliable estimate.

REFERENCES

Various articles referenced in the introduction section.

- https://www.wsj.com/articles/did-the-olympic-pool-give-some-swimmers-an-advantage-1471470741
- https://www.washingtonpost.com/news/wonk/wp/2016/09/01/these-charts-clearly-show-how-some-olympic-swimmers-may-have-gotten-an-unfair-advantage/
- https://qz.com/761280/researchers-believe-certain-lanes-in-the-olympic-pool-may-have-given-some-swimmers-an-advantage/
- http://www.bbc.co.uk/newsbeat/article/37083059/are-there-lane-advantages-in-athletics-swimming-and-track-cycling

All data used for analysis was retrieved via SwimPhone.com from CCCAA meets.

- https://www.swimphone.com/
- https://www.cccaasports.org

Olympic swimming data was provided by Sports Reference.

- https://www.sports-reference.com/olympics

APPENDIX

Appendix A: Python Script

```
# Import necessary packages
import numpy as np
import pandas as pd
import urllib
from bs4 import BeautifulSoup
from datetime import datetime

# Function to scrape finals event tables on SwimPhone.com
def scrape_event(url, gender, distance, stroke, lanes):
    r = urllib.request.urlopen(url)
    soup = BeautifulSoup(r)

    names = soup.find_all('tr')

    labels = []
    data = []

    for h in names[0].find_all(['th', 'td']):
        labels.append(h.get_text().strip().replace('\t', '').replace('\n', '').replace('\r\r', '
').replace('\r', ' '))

    for swimmer in names[1:]:
        swimmer_list = []

        for s in swimmer.find_all(['td', 'th']):
            s_str = s.get_text().strip()

            if s_str.isnumeric():
                swimmer_list.append(float(s_str))
            elif s_str == '':
                swimmer_list.append(np.NaN)
            else:
                swimmer_list.append(s_str)

        data.append(swimmer_list)

    df = pd.DataFrame(data, columns=labels)

    df['Gender'] = gender
    df['Distance'] = float(distance)
    df['Stroke'] = stroke
    df['Lanes'] = float(lanes)

    df[['Finals Heat', 'Finals Lane']] = df['Finals HT/LN'].str.split('/', expand=True)

    df[['Prelims Heat', 'Prelims Lane']] = df['Prelims HT/LN'].str.split('/', expand=True)

    df = df.dropna(subset=['Finals Pl'])

    if 'Club' in df.columns:
        df = df.rename(columns = {'Club':'School'})

    return df

# Create empty dataset with all necessary columns and then append to it after scraping each event
one by one
df = pd.DataFrame(columns=['Swimmer', 'Gender', 'Stroke', 'Distance', 'Prelims Time', 'Finals
Time', 'Finals Lane', 'Lanes', 'School', 'Seed Time', 'Prelims Lane', 'Prelims Heat', 'Finals
Heat', 'Prelims Pl', 'Finals Pl', 'Pts'])

# Individual event scraping omitted here (Appendix D)
# Scraping format: df = pd.concat([df, scrape_event('url', 'gender', distance, 'stroke', number
of lanes)], join='inner')

# Create treatment "Lane" variable
df['Lane'] = np.NaN
df.loc[(df['Finals Lane'] == '4') | (df['Finals Lane'] == '5'), 'Lane'] = 'Inside'
df.loc[(df['Finals Lane'] == '1') | (df['Finals Lane'] == '8'), 'Lane'] = 'Outside'

# Keep only swimmers who swam in inside or outside lanes in finals (discard those that swam in
lanes 2/3 or 6/7)
df = df.dropna(subset=['Lane'])
```

```
# Function to convert time in mm:ss format to seconds
def time_convert(x):
    if ':' not in x:
        m = 0
        s = float(x)
    else:
        m, s = map(float, x.split(':'))
    return (m * 60) + s

# Function to convert seed time where NT (no time) may be present
def seed_cleaner(x):
    if x == 'NT':
        x = np.NaN
    else:
        x = time_convert(x)
    return x

# Apply functions to time variables to convert to seconds
df['Prelims Time'] = df['Prelims Time'].apply(time_convert)
df['Finals Time'] = df['Finals Time'].apply(time_convert)
df['Seed Time'] = df['Seed Time'].apply(seed_cleaner)

# Drop records that have no seed time
df = df.dropna(subset=['Seed Time'])

# Drop columns unnecessary for analysis
df = df.drop(columns = ['Swimmer', 'Distance', 'School', 'Finals Lane', 'Prelims Heat',
                        'Finals Heat', 'Prelims Pl', 'Finals Pl', 'Prelims Lane', 'Pts'])

# Get rid of spaces in variable names
df = df.rename(columns = {'Finals Time':'FinalsTime', 'Prelims Time':'PrelimsTime',
                          'Seed Time':'SeedTime'})

# Save output to .csv file
df.to_csv('clean.csv', index = False)
```

Appendix B: R Script

```
# Load necessary packages
library(dplyr)
library(ggplot2)
library(ggthemes)
library(MatchIt)


### Data Preparation ###

df = read.csv('clean.csv')

# Define improvement as difference from seed time to prelims time
df$Improvement = df$SeedTime - df$PrelimsTime

# Keep name of lane (i.e. inside or outside) for graphing purposes
df$LaneName = df$Lane
df$Lane = ifelse(df$Lane == 'Outside', 1, 0)


### Data Exploration ###

## Figure 1 ##
ggplot(df, aes(x = PrelimsTime)) + theme_economist() + geom_histogram() + labs(x = 'Prelims
Time', y = 'Count', title = 'Distribution of Prelims Time') + scale_fill_economist()
ggplot(df, aes(x = FinalsTime)) + theme_economist() + geom_histogram() + labs(x = 'Finals Time',
y = 'Count', title = 'Distribution of Finals Time') + scale_fill_economist()
ggplot(df, aes(x = Improvement)) + theme_economist() + geom_histogram() + geom_vline(xintercept =
0, color = 'white', linetype = 'dashed') + labs(x = 'Improvement', y = 'Count', title =
'Distribution of Improvement') + scale_fill_economist()

## Figure 2 ##
ggplot(df, aes(x = Gender, fill = Gender)) + theme_economist() + geom_bar() +
theme(legend.position = 'none') + labs(y = 'Count', title = 'Distribution of Gender') +
scale_fill_economist()
ggplot(df, aes(x = Stroke, fill = Stroke)) + theme_economist() + geom_bar() +
theme(legend.position = 'none') + labs(y = 'Count', title = 'Distribution of Stroke') +
scale_fill_economist()
ggplot(df, aes(x = LaneName, fill = LaneName)) + theme_economist() + geom_bar() +
theme(legend.position = 'none') + labs(y = 'Count', title = 'Distribution of Treatment') +
scale_fill_economist()

## Figure 3 ##
ggplot(df, aes(x = PrelimsTime, y = FinalsTime)) + theme_economist() + scale_fill_economist() +
geom_point() + labs(x = 'Prelims Time', y = 'Finals Time', title = 'Correlation of .994 between
\nPrelims and Finals Time')
cor(df$PrelimsTime, df$FinalsTime)
ggplot(df, aes(x = FinalsTime, fill = Gender)) + theme_economist() + scale_fill_economist() +
geom_density(alpha = 0.5) + labs(x = 'Finals Time', y = 'Density', title = 'Finals Performance by
Gender')
ggplot(df, aes(x = FinalsTime, fill = Stroke)) + theme_economist() + scale_fill_economist() +
geom_density(alpha = 0.5) + labs(x = 'Finals Time', y = 'Density', title = 'Finals Performance by
Stroke')


### Main Analysis ###

# Run a regression without any matching
model_naive = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data = df)
summary(model_naive)

# t-tests show that both prelims time and improvement vary greatly between treatment groups
t.test(PrelimsTime ~ Lane, data = df)
t.test(Improvement ~ Lane, data = df)

# Gender is very balanced between groups
summary(df[df$Lane == 1,]$Gender) / summary(factor(df$Lane))[2]
summary(df[df$Lane == 0,]$Gender) / summary(factor(df$Lane))[1]

# Strokes are also quite balanced across treatment groups
summary(df[df$Lane == 1,]$Stroke) / summary(factor(df$Lane))[2]
summary(df[df$Lane == 0,]$Stroke) / summary(factor(df$Lane))[1]

# Preparing dataframes for plotting Figure 4
prelims_outside = mean(df[df$Lane == 1,]$PrelimsTime)
prelims_inside = mean(df[df$Lane == 0,]$PrelimsTime)
improve_outside = mean(df[df$Lane == 1,]$Improvement)
improve_inside = mean(df[df$Lane == 0,]$Improvement)
```

```
graph = data.frame(PrelimsTime = c(prelims_outside, prelims_inside))
graph2 = data.frame(Improvement = c(improve_outside, improve_inside))
graph$Lane = c('Outside', 'Inside')
graph2$Lane = c('Outside', 'Inside')

# Fit propensity score model
ps_model = glm(Lane ~ PrelimsTime + Gender + Stroke + Improvement, data = df, family =
'binomial')
df$PS = ps_model$fitted.values

## Figure 4 ##
ggplot(graph, aes(x = Lane, y = PrelimsTime, fill = Lane)) + theme_economist() +
scale_fill_economist() + geom_col() + geom_text(aes(label = round(PrelimsTime, 2)), stat =
"identity", vjust = -.15) + theme(legend.position = 'none') + labs(y = 'Prelims Time', title =
'Unbalanced Speed Across Groups')
ggplot(graph2, aes(x = Lane, y = Improvement, fill = Lane)) + theme_economist() +
scale_fill_economist() + geom_col() + geom_text(aes(label = round(Improvement, 2)), stat =
"identity", vjust = -.15) + theme(legend.position = 'none') + labs(title = 'Unbalanced
Improvement \nAcross Groups')
ggplot(df, aes(x = PS, fill = LaneName)) + theme_economist() + scale_fill_economist() +
geom_density(alpha = 0.5) + labs(x = 'Propensity Score', y = 'Density', title = 'Misalignment
Before Matching')

# Match based on propensity scores
match_output = matchit(Lane ~ PrelimsTime + Gender + Stroke + Improvement, data = df, method =
'nearest', distance = 'logit', caliper = 0.01, replace = FALSE, ratio = 1)
summary(match_output)
df_match = match.data(match_output)

# t-tests show insignificant differences in prelims time and improvement across treatment groups
t.test(PrelimsTime ~ Lane, data = df_match)
t.test(Improvement ~ Lane, data = df_match)

# Gender seems balanced across treatment groups
summary(df_match[df_match$Lane == 1,]$Gender) / summary(factor(df_match$Lane))[2]
summary(df_match[df_match$Lane == 0,]$Gender) / summary(factor(df_match$Lane))[1]

# Strokes also appear fairly balanced between groups, although there is slight imbalance in
backstroke and freestyle
summary(df_match[df_match$Lane == 1,]$Stroke) / summary(factor(df_match$Lane))[2]
summary(df_match[df_match$Lane == 0,]$Stroke) / summary(factor(df_match$Lane))[1]

# Preparing dataframes for plotting Figure 5
match_prelims_outside = mean(df_match[df_match$Lane == 1,]$PrelimsTime)
match_prelims_inside = mean(df_match[df_match$Lane == 0,]$PrelimsTime)
match_improve_outside = mean(df_match[df_match$Lane == 1,]$Improvement)
match_improve_inside = mean(df_match[df_match$Lane == 0,]$Improvement)
match_graph = data.frame(PrelimsTime = c(match_prelims_outside, match_prelims_inside))
match_graph2 = data.frame(Improvement = c(match_improve_outside, match_improve_inside))
match_graph$Lane = c('Outside', 'Inside')
match_graph2$Lane = c('Outside', 'Inside')

## Figure 5 ##
ggplot(match_graph, aes(x = Lane, y = PrelimsTime, fill = Lane)) + theme_economist() +
scale_fill_economist() + geom_col() + geom_text(aes(label = round(PrelimsTime, 2)), stat =
"identity", vjust = -.15) + theme(legend.position = 'none') + labs(y = 'Prelims Time', title =
'Balanced Speed After Matching')
ggplot(match_graph2, aes(x = Lane, y = Improvement, fill = Lane)) + theme_economist() +
scale_fill_economist() + geom_col() + geom_text(aes(label = round(Improvement, 2)), stat =
"identity", vjust = -.15) + theme(legend.position = 'none') + labs(title = 'Balanced Improvement
\nAfter Matching')
ggplot(df_match, aes(x = PS, fill = LaneName)) + theme_economist() + scale_fill_economist() +
geom_density(alpha = 0.5) + labs(x = 'Propensity Score', y = 'Density', title = 'Alignment After
Matching')

# Re-run regression on matched dataframe
model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
df_match)
summary(model_match)


### Sensitivity Analysis ###

# Different caliper values
c1_match_output = matchit(Lane ~ PrelimsTime + Gender + Stroke + Improvement, data = df, method =
'nearest', distance = 'logit', caliper = 0.005, replace = FALSE, ratio = 1)
summary(c1_match_output)
c1_df_match = match.data(c1_match_output)
c1_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
c1_df_match)
summary(c1_model_match)
```

```
c2_match_output = matchit(Lane ~ PrelimsTime + Gender + Stroke + Improvement, data = df, method =
'nearest', distance = 'logit', caliper = 0.1, replace = FALSE, ratio = 1)
summary(c2_match_output)
c2_df_match = match.data(c2_match_output)
c2_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
c2_df_match)
summary(c2_model_match)

# Different functional form
p_match_output = matchit(Lane ~ PrelimsTime + Gender + Stroke + Improvement, data = df, method =
'nearest', distance = 'probit', caliper = 0.01, replace = FALSE, ratio = 1)
summary(p_match_output)
p_df_match = match.data(p_match_output)
p_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
p_df_match)
summary(p_model_match)

# Different ratio of control and treatment
r_match_output = matchit(Lane ~ PrelimsTime + Gender + Stroke + Improvement, data = df, method =
'nearest', distance = 'logit', caliper = 0.01, replace = FALSE, ratio = 2)
summary(r_match_output)
r_df_match = match.data(r_match_output)
r_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
r_df_match)
summary(r_model_match)

# Sampling with replacement
wr_match_output = matchit(Lane ~ PrelimsTime + Gender + Stroke + Improvement, data = df, method =
'nearest', distance = 'logit', caliper = 0.01, replace = TRUE, ratio = 1)
summary(wr_match_output)
wr_df_match = match.data(wr_match_output)
wr_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
wr_df_match)
summary(wr_model_match)

# Different matching methods
m1_match_output = matchit(Lane ~ PrelimsTime + Gender + Stroke + Improvement, data = df, method =
'genetic', distance = 'logit', caliper = 0.2, replace = FALSE, ratio = 1) # Higher caliper
necessary to find sufficient matches
summary(m1_match_output)
m1_df_match = match.data(m1_match_output)
m1_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
m1_df_match)
summary(m1_model_match)
m2_match_output = matchit(Lane ~ PrelimsTime + Gender + Stroke + Improvement, data = df, method =
'optimal', distance = 'logit', caliper = 0.01, replace = FALSE, ratio = 1)
summary(m2_match_output)
m2_df_match = match.data(m2_match_output)
m2_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
m2_df_match)
summary(m2_model_match)

# Matching on subsets of variables
s1_match_output = matchit(Lane ~ PrelimsTime, data = df, method = 'nearest', distance = 'logit',
caliper = 0.01, replace = FALSE, ratio = 1)
summary(s1_match_output)
s1_df_match = match.data(s1_match_output)
s1_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
s1_df_match)
summary(s1_model_match)
s2_match_output = matchit(Lane ~ PrelimsTime + Gender + Stroke, data = df, method = 'nearest',
distance = 'logit', caliper = 0.01, replace = FALSE, ratio = 1)
summary(s2_match_output)
s2_df_match = match.data(s2_match_output)
s2_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane, data =
s2_df_match)
summary(s2_model_match)

# Regression with interactions
i1_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane +
Gender:Stroke, data = df_match)
summary(i1_model_match)
i2_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane +
Gender:Lane, data = df_match)
summary(i2_model_match)
i3_model_match = lm(FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement + Lane +
Stroke:Lane, data = df_match)
summary(i3_model_match)
```

Appendix C: Regression Output

Naïve Regression:

```
Call:
lm(formula = FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement +
    Lane, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-6.6654 -0.6553  0.0211  0.6266 10.0486

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.045716   0.326337   0.140 0.888611
PrelimsTime   0.984737   0.004568 215.588  < 2e-16 ***
GenderM      -0.038640   0.091727  -0.421 0.673643
StrokeBreast  0.057400   0.106711   0.538 0.590738
StrokeFly    -0.018763   0.105484  -0.178 0.858851
StrokeFree    0.144226   0.109073   1.322 0.186308
Improvement   0.144706   0.014376  10.066  < 2e-16 ***
Lane          0.282957   0.076239   3.711 0.000215 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.294 on 1292 degrees of freedom
Multiple R-squared:  0.9882,    Adjusted R-squared:  0.9881
F-statistic: 1.546e+04 on 7 and 1292 DF,  p-value: < 2.2e-16
```

Matched Regression[1]:

```
Call:
lm(formula = FinalsTime ~ PrelimsTime + Gender + Stroke + Improvement +
    Lane, data = df_match)

Residuals:
    Min      1Q  Median      3Q     Max
-5.8515 -0.6365  0.0191  0.6129  7.0851

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.459059   0.549729  -0.835 0.403940
PrelimsTime   0.992913   0.007630 130.127  < 2e-16 ***
GenderM      -0.047241   0.126528  -0.373 0.708980
StrokeBreast  0.001001   0.132021   0.008 0.993954
StrokeFly     0.116149   0.131371   0.884 0.376903
StrokeFree    0.266222   0.139353   1.910 0.056449 .
Improvement   0.139384   0.019498   7.149 2.03e-12 ***
Lane          0.324239   0.085981   3.771 0.000175 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.199 on 772 degrees of freedom
Multiple R-squared:  0.9867,    Adjusted R-squared:  0.9865
F-statistic:  8151 on 7 and 772 DF,  p-value: < 2.2e-16
```

[1] Note that coefficient estimates and p-values may vary slightly as matching algorithm will produce different matches each time it is run.

Appendix D: Event Scraping

```
# 2019 Big 8 Swimming Championships (Apr 18-20)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11370&meid=289527&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11370&meid=289528&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11370&meid=289531&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11370&meid=289532&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11370&meid=289533&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11370&meid=289534&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11370&meid=289543&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11370&meid=289544&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2019 South Coast Conference Swim and Dive Championships (Apr 18-20)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11366&meid=289356&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11366&meid=289357&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11366&meid=289360&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11366&meid=289361&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11366&meid=289362&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11366&meid=289363&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11366&meid=289372&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11366&meid=289373&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2019 Orange Empire Conference Champs (Apr 18-20)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11368&meid=289452&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11368&meid=289453&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11368&meid=289436&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11368&meid=289437&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11368&meid=289440&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11368&meid=289441&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11368&meid=289442&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11368&meid=289443&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
```

```
# 2019 Western States Conference Championships (Apr 18-20)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11365&meid=289316&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11365&meid=289317&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11365&meid=289320&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11365&meid=289321&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11365&meid=289322&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11365&meid=289323&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11365&meid=289332&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11365&meid=289333&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2019 Coast Conference Championships (Apr 18-20)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11938&meid=310003&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11938&meid=310004&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11938&meid=310007&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11938&meid=310008&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11938&meid=310009&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11938&meid=310010&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11938&meid=310019&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11938&meid=310020&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2019 Pacific Coast Athletic Conference Championships (Apr 18-20)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11367&meid=289396&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11367&meid=289397&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11367&meid=289400&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11367&meid=289401&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11367&meid=289402&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11367&meid=289403&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11367&meid=289412&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=11367&meid=289413&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2018 CCCAA Swim & Drive State Championships (May 3-5)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10093&meid=252657&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
```

```
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10093&meid=252658&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10093&meid=252661&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10093&meid=252662&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10093&meid=252663&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10093&meid=252664&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10093&meid=252673&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10093&meid=252674&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2018 Bay Valley Conference Championships (Apr 19-21)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10095&meid=252743&e=23&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10095&meid=252744&e=24&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10095&meid=252747&e=27&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10095&meid=252748&e=28&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10095&meid=252749&e=29&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10095&meid=252750&e=30&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10095&meid=252761&e=41&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10095&meid=252762&e=42&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2018 Big 8 Swimming Championships (Apr 19-21)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10169&meid=255765&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10169&meid=255766&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10169&meid=255769&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10169&meid=255770&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10169&meid=255771&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10169&meid=255772&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10169&meid=255781&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10169&meid=255782&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2018 Coast Conference Championships (Apr 19-21)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10096&meid=252785&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10096&meid=252786&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
```

```
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10096&meid=252789&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10096&meid=252790&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10096&meid=252791&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10096&meid=252792&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10096&meid=252801&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10096&meid=252802&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2018 Orange Empire Conference Champs (Apr 19-21)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10097&meid=252825&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10097&meid=252826&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10097&meid=252829&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10097&meid=252830&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10097&meid=252831&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10097&meid=252832&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10097&meid=252841&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10097&meid=252842&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2018 Pacific Coast Athletic Conference Championships (Apr 19-21)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10098&meid=252865&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10098&meid=252866&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10098&meid=252869&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10098&meid=252870&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10098&meid=252871&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10098&meid=252872&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10098&meid=252881&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10098&meid=252882&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2018 South Coast Conference Swim and Dive Championships (Apr 19-21)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10099&meid=252905&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10099&meid=252906&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10099&meid=252909&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
```

```
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10099&meid=252910&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10099&meid=252911&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10099&meid=252912&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10099&meid=252921&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10099&meid=252922&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2018 Western States Conference Championships (Apr 19-21)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10100&meid=252945&e=17&s=fin
als', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10100&meid=252946&e=18&s=fin
als', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10100&meid=252949&e=21&s=fin
als', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10100&meid=252950&e=22&s=fin
als', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10100&meid=252951&e=23&s=fin
als', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10100&meid=252952&e=24&s=fin
als', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10100&meid=252961&e=33&s=fin
als', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=10100&meid=252962&e=34&s=fin
als', 'M', 100, 'Free', 8)], join='inner')

# 2017 CCCAA Swim & Dive State Championships (May 4-6)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9070&meid=231483&e=17&s=fina
ls', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9070&meid=231484&e=18&s=fina
ls', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9070&meid=231487&e=21&s=fina
ls', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9070&meid=231488&e=22&s=fina
ls', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9070&meid=231489&e=23&s=fina
ls', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9070&meid=231490&e=24&s=fina
ls', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9070&meid=231499&e=33&s=fina
ls', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9070&meid=231500&e=34&s=fina
ls', 'M', 100, 'Free', 8)], join='inner')

# 2017 Central Valley Conference Championships (Apr 20-22)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9042&meid=230405&e=23&s=fina
ls', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9042&meid=230406&e=24&s=fina
ls', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9042&meid=230409&e=27&s=fina
ls', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9042&meid=230410&e=28&s=fina
ls', 'M', 100, 'Breast', 8)], join='inner')
```

```
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9042&meid=230411&e=29&s=fina
ls', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9042&meid=230412&e=30&s=fina
ls', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9042&meid=230423&e=41&s=fina
ls', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9042&meid=230424&e=42&s=fina
ls', 'M', 100, 'Free', 8)], join='inner')

# 2017 Coast Conference Championships (Apr 20-22)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8525&meid=211360&e=17&s=fina
ls', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8525&meid=211361&e=18&s=fina
ls', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8525&meid=211364&e=21&s=fina
ls', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8525&meid=211365&e=22&s=fina
ls', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8525&meid=211366&e=23&s=fina
ls', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8525&meid=211367&e=24&s=fina
ls', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8525&meid=211376&e=33&s=fina
ls', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8525&meid=211377&e=34&s=fina
ls', 'M', 100, 'Free', 8)], join='inner')

# 2017 South Coast Conference Swim and Dive Championships (Apr 20-22)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9079&meid=231849&e=17&s=fina
ls', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9079&meid=231850&e=18&s=fina
ls', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9079&meid=231853&e=21&s=fina
ls', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9079&meid=231854&e=22&s=fina
ls', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9079&meid=231855&e=23&s=fina
ls', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9079&meid=231856&e=24&s=fina
ls', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9079&meid=231865&e=33&s=fina
ls', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9079&meid=231866&e=34&s=fina
ls', 'M', 100, 'Free', 8)], join='inner')

# 2017 Western States Conference Championships (Apr 20-22)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9033&meid=230056&e=17&s=fina
ls', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9033&meid=230057&e=18&s=fina
ls', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9033&meid=230060&e=21&s=fina
ls', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9033&meid=230061&e=22&s=fina
ls', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9033&meid=230062&e=23&s=fina
ls', 'F', 100, 'Back', 8)], join='inner')
```

```
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9033&meid=230063&e=24&s=fina
ls', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9033&meid=230072&e=33&s=fina
ls', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9033&meid=230073&e=34&s=fina
ls', 'M', 100, 'Free', 8)], join='inner')

# 2017 Big 8 Championships (Apr 20-22)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8693&meid=217883&e=17&s=fina
ls', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8693&meid=217884&e=18&s=fina
ls', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8693&meid=217887&e=21&s=fina
ls', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8693&meid=217888&e=22&s=fina
ls', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8693&meid=217889&e=23&s=fina
ls', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8693&meid=217890&e=24&s=fina
ls', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8693&meid=217899&e=33&s=fina
ls', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8693&meid=217900&e=34&s=fina
ls', 'M', 100, 'Free', 8)], join='inner')

# 2017 Orange Empire Conference Champs (Apr 20-22)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8548&meid=211870&e=17&s=fina
ls', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8548&meid=211871&e=18&s=fina
ls', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8548&meid=211874&e=21&s=fina
ls', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8548&meid=211875&e=22&s=fina
ls', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8548&meid=211876&e=23&s=fina
ls', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8548&meid=211877&e=24&s=fina
ls', 'M', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8548&meid=211886&e=33&s=fina
ls', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=8548&meid=211887&e=34&s=fina
ls', 'M', 100, 'Free', 8)], join='inner')

# 2017 Pacific Coast Athletic Conference Championships (Apr 20-22)
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9045&meid=230551&e=17&s=fina
ls', 'F', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9045&meid=230552&e=18&s=fina
ls', 'M', 100, 'Fly', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9045&meid=230555&e=21&s=fina
ls', 'F', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9045&meid=230556&e=22&s=fina
ls', 'M', 100, 'Breast', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9045&meid=230557&e=23&s=fina
ls', 'F', 100, 'Back', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9045&meid=230558&e=24&s=fina
ls', 'M', 100, 'Back', 8)], join='inner')
```

```
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9045&meid=230567&e=33&s=finals', 'F', 100, 'Free', 8)], join='inner')
df = pd.concat([df,
scrape_event('https://www.swimphone.com/meets/event_results.cfm?smid=9045&meid=230568&e=34&s=finals', 'M', 100, 'Free', 8)], join='inner')
```