Today we're going to use `SwimmeR` to build a large database of results – that's the overall goal. Having a more specific goal in mind can be useful though, so here goes.

My wife is a college administrator. It's part of her job to keep up on what Kids These Days (TM) are doing and one thing they're apparently doing is getting participation trophies. Looking back to my childhood I participated in a lot of swim meets, and swam a lot of races. For my trouble I've got piles of ribbons, enough medals to cover my entire head if I were to try and wear them all at once, and even a few plaques – but almost no trophies. My lack of trophies could be due to my modest swimming abilities, but I've checked with some of my faster friends and teammates and they report the same thing – very few trophies. I don't want swimmers to be left out of the zeitgeist, and if that means participation trophies then I've got some wrongs to right.

Who should get these participation trophies though? Not everyone of course, just handing them out would cheapen them. I want to award participation trophies to only the most participatory athletes, the crème de la crème of participation. To that end today I'm going to try and determine the most participatory swimmers in my USA Swimming LSC for the 2018-2019 season. These ultra-participatory athletes will win the coveted Swimming + Data Science Participation Trophies.

Rules are as follows:

1. The winners shall be the athlete(s) who have participated in, and legally completed, the most individual events in the 2018-2019 Niagara LSC age group season and the athlete(s) who have swam the furthest distance in their events during the same period
2. Any individual event swam at a meet in Niagara LSC during the 2018-2019 age group season will count. Sectionals will also count if the results are on the Niagara LSC website in .pdf or .html form
3. Only swimmers from Niagara LSC are eligible
4. Tiebreaker will be the other category – total distance swam for the events category, and number of events swam for the distance category

To do this I'll of course need results for all meets swam in Niagara LSC for 2018-2019 – `SwimmeR`, take your mark!

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

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

Please note the following analysis was done using the development version of `SwimmeR` current on November 11th, 2020. The most recent dev version of `SwimmeR` is available via `devtools::install_github("gpilgrim2670/SwimmeR")`

---

## Getting Links to Results

During the State-Off Tournament we sometimes built lists of links to pull results from. In those cases the link addresses followed a pattern, which we used to generate them within `R`. What do we do though if there are links we're interested in, but their addresses don't follow a pattern?

Niagara LSC has a repository of results for each season, and the 2018-2019 one is here. It's basically a list of meets, each with a link called "Results", and each with a different address. We want the content of all of those links, but there's a lot and typing all the addresses in manually is of course out of the question. To get them we need the CSS selector that defines them. If you're a CSS whiz maybe you can just read the website source code a determine it, but I use the Selector Gadget Google Chrome add-on. After a bit of clicking around I've determined that the selector in this case is called `span a`.

Now it's time to deploy the `rvest` package. `rvest` is another hit from the good folks at RStudio and for our purposes today it's pretty simple. We'll need to direct it to the results repository and then extract all the links from `span a`. We'll use `read_html` to read the results repository web page, then `html_nodes` to get the content of our selector, and also `html_attr` to make sure that content is a link. In this way we'll build a list of links.

```
web_url <- "https://www.teamunify.com/team/eznslsc/page/times/2018-2019-results"
selector <- "span a"

page_contents <- read_html(web_url)
links <- html_attr(html_nodes(page_contents, selector), "href")
```

If we look at our list of links we'll see that most of them look like .pdf files of results (what we want), but some of them, like the first one, are other junk.

```
head(links, 5)
## [1] "/cdn-cgi/l/email-protection#493e2c2b24283a3d2c3b092720282e
283b283a3e202467263b2e"
## [2] "/eznslsc/UserFiles/File/Meet-Results/2018-2019/
jets08082019_067546.pdf"
## [3] "/eznslsc/UserFiles/File/Meet-Results/2018-2019/
ttsc08062019_079354.pdf"
## [4] "/eznslsc/UserFiles/File/Meet-Results/2018-2019/2019-ez-senrior-
lc-champs-results_092276.pdf"
## [5] "/eznslsc/UserFiles/File/Meet-Results/2018-2019/2019-ez-senrior-
lc-champs-tt-results_029460.pdf"
```

We only want links that end in either ".pdf" or ".htm" because a) those are the results and b) `SwimmeR` can read those file types.

```
links <- links[links %>%
  str_detect("\\.pdf$|.htm")]
```

Looking at `head(links)` again we can also see that these aren't full links. They're partials, missing their beginnings. Links should begin with something like "https://www.r-bloggers.com/". In this case the desired beginning, as visible in your browser window, is "http://www.teamunify.com". We've got to be a bit careful though - there might be a few fully formed links in our list. Let's check.

```
links[str_detect(links, "http")]
## [1] "http://easternzoneswimming.org/meet_results/2019SpringSectionals_
LongCourse_Results_Women.html"
## [2] "http://easternzoneswimming.org/meet_results/2019SpringSectionals_
LongCourse_Results_Men.html"
## [3] "http://easternzoneswimming.org/meet_results/2019SpringSectionals_
LongCourse_Results_TimeTrials.html"
## [4] "http://easternzoneswimming.org/meet_results/2019SpringSectionals_
LongCourse_TeamScores.html"
## [5] "http://easternzoneswimming.org/meet_results/2019SpringSectionals_
LongCourse_HighPoint.html"
## [6] NA
```

So there's a few Sectional results that are already full links. Let's modify the rest to make them complete as well.

```
links <- ifelse(str_detect(links, "http"), paste0("", links),
paste0("http://www.teamunify.com", links))
```

---

## Reading in Results

Now that we have our list of links we can use `read_results` to pull their contents into `R`. Because there might still be some bad links we'll use the `safely` function inside of `map`. `map` will apply `read_results` to every element of `links`, but because `read_results` is wrapped in `safely` if it fails rather than posting an error and aborting it will note the issue and keep going, reading in the rest of the elements of `links`.

We'll then name each element of the list of results produced by `read_results` with the link from whence it came.

**Please note** - running the following code will take several minutes. I also have the results hosted on github. You can download them directly using this code: `raw_results <- readRDS(url("https://github.com/gpilgrim2670/Pilgrim_Data/raw/master/Niagara%20LSC%202018-2019%20Dataset/Niagara_raw_results.rds))`.

```
raw_results <- map(links, safely(read_results, otherwise = NA))

names(raw_results) <- links
```

We now have a list of lists. Each sublist contains results (hopefully) and also an error register as elements 1 and 2 respectively. We want to remove sublists where the error register isn't `NULL` - that is, sublists that have an error. We also then want only the first element (the results) of the remaining sublists - we don't want the empty error register. Here's a handy little function to do just that, which we can apply to `raw_results` to clean them.

```
discard_errors <- function(results) {
  element_extract <- function(lst, n) {
    sapply(lst, `[`, n)
  }
  clean_results <- discard(results, ~ !is.null(.x$error))
  clean_results <- element_extract(clean_results, 1)
  return(clean_results)
}

clean_results <- discard_errors(raw_results)
```

---

## Parsing Results

With our `clean_results` in hand we now can pass them to `swim_parse`, again via `map`, to apply over every element of `clean_results`, and again inside `safely`, to power through (but note) any errors.

The `typo` and `replacement` arguments to `swim_parse` are used for correcting issues in the source materials. Their use requires a bit of practice so that typos can be correctly identified and fixed, but in general the issues relate to one of two things.

1. Special strings. Strings like “DQ” and “–” (two or three dashes together) have special meaning in swimming results, and therefore inside `swim_parse`. “DQ” obviously means a swimmer has been disqualified. The two (or three) dashes, “–” are used instead of a place number for athletes who were DQ’d, or scratched etc. Incorrect usage of these strings within results can cause problems, but are easily corrected with `typo` and `replace`
2. White space issues. `swim_parse` uses instances of two or more spaces to split results into dataframe columns. If those spaces aren’t as they should be, for example if an athlete or team name is long enough that it encroaches on the next column parsing issues will result.

**Please note** - running the following code will take a long time, approximately 30 minutes. I also have the results hosted on github. You can download them directly using this code: `Niagara_Results <- readRDS(url("`https://github.com/gpilgrim2670/Pilgrim_Data/raw/master/Niagara%20LSC%202018-2019%20Dataset/Niagara_2018_2019.rds`"))`.

```
Niagara_Results <-
  map(
    clean_results,
    safely(Swim_Parse, otherwise = NA),
    typo = c(
      "Greater Rochester Area YMCA --NI", # incorrect usage of special
string "--"
      "(?<=[:alpha:]) (?=[:digit:])", # regex to identify any space
that is preceded by a letter and followed by a number, so that it can
be expanded to two+ spaces - a white space issue
      "PENINSULA WAVE RIDERS SWIMMING-A", # team name so long that it
got to close to athlete's times - a white space issue
      "DQY", # wrong DQ string
      "XDQ", # wrong DQ string
      "111 ", # an actual typo in a kid's name
      "Alexis/Lexi J", # an actual typo in a kid's name
      "La Face, Isabella or Bella F", # an actual typo in a kid's name
      "Cunningham, Rhys*", # an actual typo in a kid's name
      " FUT " # extraneous string used to signify a Futures cut
    ),
    replacement = c(
      "Greater Rochester Area YMCA-NI",
      "   ",
      "PENINSULA WAVE RIDERS SWIMMING  ",
      "DQ",
      "DQ",
      "III",
      "Alexis J",
      "La Face, Isabella",
      "Cunningham, Rhys",
      "",
    )
  )
```

We've got a list of results, but a few of them will contain errors. For example http://easternzoneswimming.org/meet_results/2019SpringSectionals_LongCourse_HighPoint.html doesn't actually contain results, just a high-point scores. That one won't read. We'll just use our discard errors function again and then `data.table::rbindlist` to stick all the results together with a column called `Source`, containing the element name, which we already made the link it was downloaded from.

```
Niagara_Results <- discard_errors(Niagara_Results)

Niagara_Results <- data.table::rbindlist(Niagara_Results, use.names =
TRUE, idcol = "Source", fill = TRUE)
```

How big is our data set? `dim` can tell us.

```
dim(Niagara_Results)
## [1] 108127     11
```

We've got 108127 rows and 11 columns. There are bigger sets for sure, but this one isn't small.

---

## Cleaning the Data

A few minor things before we get going. The Source column has appended a string to the end of our links. Let's get rid of that. We'd also like to rename the `Grade` column and `School` column output by `swim_parse` to `Age` and `Team` respectively. This change will be implemented in a future version of `SwimmeR`.

```
Niagara_Results  <- Niagara_Results  %>%
  mutate(Source = str_remove(Source, "\\.result\\.result")) %>%
  mutate(Age = as.numeric(Grade),
         Team = School) %>%
  select(Source, Place, Name, Age, Event, everything()) %>%
  select(-Grade, - School)
```

---

## Names - A Thorny Problem

It's tempting to just rush in and count events by athlete name - let's try it.

```
Niagara_Results %>%
  filter(is.na(Name) == FALSE,
         DQ == 0) %>% # DQs don't count
  group_by(Name) %>%
  summarise(No_Swims = n(), # number of events swam by each athlete
            Team = get_mode(Team), # use only most common team name
            Age = round(mean(Age, na.rm = TRUE), 0)) %>%
  arrange(desc(No_Swims)) %>%
  head(5) %>%
  flextable_style()
```

| Name | No_Swims | Team | Age |
|---|---|---|---|
| Chung, Emily | 99 | Clarence Swim Club-NI | 13 |
| Anderson, Max | 96 | VELO-NI | 10 |
| Cavallerano, Jack S | 96 | Liverpool Jets Swim Club-NI | 11 |
| Anderson, Madden | 91 | VELO-NI | 11 |
| Barillari, Emma | 91 | Velocity Aquatics, Llc-NI | 12 |

It worked I guess, we get some names, with Emily Chung on top. Let's look a bit more closely at Ms. Chung's results though.

```
Niagara_Results %>%
  filter(str_detect(Name, "Chung")) %>%
    filter(str_detect(Name, "Emily")) %>%
  group_by(Name) %>%
  summarise(No_Swim = n(), # number of events swam
            Team = get_mode(Team)) %>% # use only most common team name
  flextable_style()
```

| Name | No_Swim | Team |
|---|---|---|
| Chung, Emily | 100 | Clarence Swim Club-NI |
| Emily Chung | 4 | Clarence Swim Club-NI |

Turns out she's here twice, once as "Chung, Emily" and again as 'Emily Chung". Our first analysis didn't catch her as"Emily Chung" though. We'll need to count both together, and other athletes may be similarly undercounted.

There's another issue highlighted by Sidra El Ghissassi. Let's take a look at her doings.

```
Niagara_Results %>%
  filter(str_detect(Name, "El Ghissassi")) %>%
  filter(str_detect(Name, "Sidra")) %>%
  group_by(Name) %>%
  summarise(No_Swim = n(), # number of events swam
            Team = get_mode(Team), # use only most common team name
            Age = round(mean(Age, na.rm = TRUE), 0)) %>%
  flextable_style()
```

| Name | No_Swim | Team | Age |
|---|---|---|---|
| El Ghissassi, Sidra | 15 | UNATTACHED-NI | 9 |
| El Ghissassi, Sidra S | 86 | Vestal Area Swim Club-NI | 9 |
| Sidra El Ghissassi | 3 | Vestal Area Swim Team-NI | 9 |

She's here three times, sometimes with a middle initial and sometimes not, sometimes with her last name first, sometimes with her first name first. What's more she has two words in her last name. We need to keep track of all this stuff.

There's also the question of nicknames. For example everyone calls me Greg, but my name is acutally Gregory. In swimming results sometimes I'm Gregory and sometimes I'm Greg. The goal here is to be able to recognize 'Emily Chung" and "Chung, Emily" as the same person, and also "Greg Pilgrim" and "Pilgrim, Gregory A" as the same person, plus "El Ghissassi, Sidra S" and "Sidra El Ghissassi" as the same person. Let's take a whack at assigning everyone an `ID` based on the first 3 letters of their first name and the last 5 letters of their last name. To do that we'll need to get everyone's name in the same order - let's do Firstname Lastname.

```
Niagara_Results <- Niagara_Results %>%
  mutate(Name_2 = str_replace(Name, " [:upper:]$", "")) %>% # removes
middle initials
  mutate(Last_Name = case_when( # pulls out last name when it's
followed by a comma
    str_detect(Name_2, ",") ~ str_split_fixed(Name_2, ",", n = 2)[, 1],
    TRUE ~ ""
  )) %>%
  na_if("") %>%
  mutate(First_Name = case_when(is.na(Last_Name) == FALSE ~
str_remove(Name_2, paste0(Last_Name, ",", " ")), # gets first name if
last name was identified above
                                TRUE ~ "")) %>%
  na_if("") %>%
  mutate(Name_3 = case_when( # recombins first names and last names
identified above into Firstname Lastname order
    is.na(First_Name) == FALSE &
      is.na(Last_Name) == FALSE ~ paste(First_Name, Last_Name, sep = "
"),
    TRUE ~ Name_2 # names not identified above are already in Firstname
Lastname order
  )) %>%
  mutate(ID = paste0( # make an ID value based on first 3 letters of
first name, last 5 letters of last name
    stringr::str_extract(Name_3, "^.{3}"),
    str_extract(Name_3, ".{5}$")
  )) %>%
  select("Name_Rework" = Name_3, # cleanup columns
         ID,
         everything(),
         -Name_2,
         -First_Name,
         -Last_Name)
```

As a check, let's look at Emily again and see if her `ID` is doing what we'd like.

```
Niagara_Results %>%
  filter(str_detect(Name, "Chung")) %>%
  filter(str_detect(Name, "Emily")) %>%
  group_by(Name) %>%
  summarise(No_Swim = n(),
            Team = get_mode(Team), # use only most common team name
            Age = round(mean(Age, na.rm = TRUE), 0),
            ID = unique(ID)) %>% # show ID
  flextable_style()
```

| Name | No_Swim | Team | Age | ID |
|---|---|---|---|---|
| Chung, Emily | 100 | Clarence Swim Club-NI | 12 | EmiChung |
| Emily Chung | 4 | Clarence Swim Club-NI | 13 | EmiChung |

And Sidra:

```
Niagara_Results %>%
  filter(str_detect(Name, "El Ghissassi")) %>%
  filter(str_detect(Name, "Sidra")) %>%
  group_by(Name) %>%
  summarise(No_Swim = n(),
            Team = get_mode(Team), # use only most common team name
            Age = round(mean(Age, na.rm = TRUE), 0),
            ID = unique(ID)) %>%
  flextable_style()
```

| Name | No_Swim | Team | Age | ID |
|---|---|---|---|---|
| El Ghissassi, Sidra | 15 | UNATTACHED-NI | 9 | Sidsassi |
| El Ghissassi, Sidra S | 86 | Vestal Area Swim Club-NI | 9 | Sidsassi |
| Sidra El Ghissassi | 3 | Vestal Area Swim Team-NI | 9 | Sidsassi |

Both girls have their own ID, which is consistent across the multiple variants of their names. This looks good and we'll stop here, but it is worth noting that there are real and sometimes unsolvable problems of this type.

If there were two athletes named Emily Chung we wouldn't be able to tell with this approach. We could maybe differentiate them based on team, but people do sometimes change teams, so even if there was a Emily Chung who swam for two different teams it could be the same person. We could also check their ages, but if you look back to Emily's table above you'll see that her age changes. She was 12, but then she turned 13. Probably. It's also possible that there are two Emily Chungs who both swim for Clarence Swim Club, but one is a year older than the other. We could go even further and try to extract dates for each event, and see if the (potentially) two Emily Chungs ever swim in the same meet but if they didn't we still couldn't be sure there was only one Emily Chung.

Additionally if there was a third girl named Emily ZChung her ID would also be EmiChung, same as our Emily Chung(s) - it's tricky! We could instead make the ID from the last 6 characters of each name, but that doesn't solve the fundamental issue. My point isn't that this is hopeless, but rather that data science is hard, and it requires some checking of assumptions to make sure your code is actually doing what you think it's doing.

---

## Distance

With our IDs in had there's now the matter of the tiebreaker - distance swam. Distance is included in Event, but there are two wrinkles. The first is getting the number of yards/meters swam out, and the second is converting them all to a consistent unit, in our case yards.

We'll extract all groups of digits from each Event using str_extract_all. This will get us the distances, but also the age groups, like "12" from "12 & Under" or "13" and "14" from "Boys 13-14…". Crucially though the distance will always be the last set of numbers, so we can use tail to get it out.

Then we'll do something similar with the strings "Yard" and "Meter", multiplying distance by 1.1 whenever str_detect detects "Meter" in Event and keeping distance the same when str_detect detects "Yard". A meter is 1.1 yards.

```
Niagara_Results <- Niagara_Results %>%
  mutate(Event_Numb = str_extract_all(Event, "\\d{1,}")) %>% # get all
groups of digits as a list
  rowwise() %>% # since rows contain lists we need this to act across
rows rather than down columns
  mutate(Distance = as.numeric(tail(Event_Numb, 1))) %>% # take the
last element of each list
  mutate(Distance = case_when(str_detect(Event, "Yard") ~ Distance ,
                              str_detect(Event, "Meter") ~ Distance *
1.1, # convert distances swam in meters to yards
                              TRUE ~ Distance)) %>%
  select(-Event_Numb)

Niagara_Results <- Niagara_Results %>%
  filter(is.na(Name) == FALSE,
         DQ == 0) %>% # DQ events don't count
  group_by(ID) %>%
  summarise(No_Swims = n(), # count up total swims
            Name = get_mode(Name), # use only most common name
            Team = get_mode(Team), # use only most common team
            Age = round(mean(Age, na.rm = TRUE), 0),
            Distance = sum(Distance, na.rm = TRUE)) %>%
  ungroup() %>%
  select(-ID)
```

## Final Results

We've reached the end, who will win the coveted Swimming + Data Science Participation Trophy for most events swam?

```
Niagara_Results %>%
  arrange(desc(No_Swims)) %>% # sort by number of swims, highest to
lowest
  head(5) %>%
  flextable_style()
```

| No_Swims | Name | Team | Age | Distance |
|---|---|---|---|---|
| 183 | Alessi, Luciana M | BAAC-NI | 9 | 15635 |
| 138 | Upcraft, Malia L | Mexico Tiger Sharks-NI | 12 | 20535 |
| 118 | Moran, Aiden | Clarence Swim Club-NI | 10 | 20885 |
| 118 | Cole, Matthew J | Oswego Laker Swim Club-NI | 14 | 32655 |
| 117 | Seelig, Jenna | Clarence Swim Club-NI | 12 | 16845 |

It's Luciana M. Alessi from Buffalo Aquatics (BAAC)! She swam a gargantuan 183 races averaging just over 85 yards per race. That's a lot!

And now for the Swimming + Data Science Participation Trophy for most furthest distance swam…

```
Niagara_Results %>%
  arrange(desc(Distance)) %>% # sort by distance swam, highest to
lowest
  head(5) %>%
  flextable_style()
```

| No_Swims | Name | Team | Age | Distance |
|---|---|---|---|---|
| 118 | Cole, Matthew J | Oswego Laker Swim Club-NI | 14 | 32655 |

| No_Swims | Name | Team | Age | Distance |
|---|---|---|---|---|
| 73 | Kruglov, Max | Star Swimming-NI | 15 | 28345 |
| 80 | Senglaub, Katie | Victor Swim Club-NI | 15 | 26435 |
| 66 | Signore, Christopher A | Town of Tonawanda Titans-NI | 15 | 26275 |
| 82 | Creed, Jason | VELO-NI | 15 | 25755 |

It’s Matthew J. Cole from the Oswego Lakers, who swam 32,655 yards over 118 events for a ~275 yard per event average! That’s over 18 miles by the way.

A massive congratulations to both Luciana and Matthew for your staggering levels of participation! If either of you happen to be reading this (or if you’re a coach from BAAC or the Lakers), get in touch, I really will have trophies made and sent.

---

## Conclusion

Thank you for joining us here at Swimming + Data Science where we’ve used `SwimmeR` to get a big dataset of swimming results and then answered the questions - who swam the most events and furthest distance. I hope you;ve enjoyed yourself - be sure to check back in next time for another adventure.