Final results for the 2021 USMS ePostal National Championships have been posted. We’ll take a look at them using some basic summary stats and charts, then work up the data from inclusion in Shiny application built to present it interactively.

This will be a tutorial style post, where I’ll prioritize making the code clear and readable, while perhaps sacrificing efficiency. No building functions and them mapping them like I sometimes do.

With that in mind let’s grab some packages. We’ll use readr to actually read in the data set. Then dplyr and stringr to work it up and finally ggplot2 and flextable to present it at various points throughout the article.

library(readr) library(dplyr) library(stringr) library(ggplot2) library(flextable)

flextable\_style <- function(x) { x %>%

flextable() %>%

bold(part = "header") %>% # bolds header

bg(bg = "#D3D3D3", part = "header") %>% # puts gray background behind the header row

autofit()

}

Postal <- read\_csv("https://raw.githubusercontent.com/gpilgrim2670/MastersPostal/ master/Postal\_Raw.csv")

Last year I did a big analysis of the 2020 ePostal results and another on the ePostal over the past two+ decades. Rather than just repurposing all that code (which you’re welcome to do if you’d like) we’ll just take a quick look at the 2021 results before working up all the data, from all the years for the Shiny app.

# How Many People Participated This Year?

Postal %>%

filter(Year == "2021") %>% group\_by(Gender) %>% summarise(Count = n()) %>% flextable\_style()

Gender Count

Gender Count

M 317

W 372

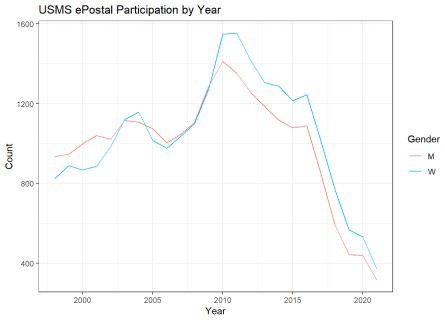
# How Does That Compare to Previous Years?

Postal %>%

group\_by(Year, Gender) %>% summarise(Count = n()) %>% ggplot() +

geom\_line(aes(x = Year, y = Count, color = Gender)) + theme\_bw() +

labs(title = "USMS ePostal Participation by Year")



# Shiny App

## Cleaning Data for a Shiny App

The data we’ve already downloaded and named Postal contains the as-reported results from all ePostals from 1998 through 2021. Let’s take a look.

names(Postal)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | [1] | "Place" | "Name" | "Age" | "USMS\_ID" |
| ## | [5] | "Distance" | "Club" | "Gender" | "Year" |
| ## | [9] | "National\_Record" |  |  |  |

Pretty self explanatory column names, and quite a bit of information. There’s more though that we can tease out.

I’d like to add to Postal in the following ways:

1. Age groups. The ePostal is scored by age group, 18-24 and then by 5 year windows thereafter (25-29 etc.)
2. Athlete’s relative place within their age and gender category by year
3. An athlete’s average 50 split, based on distance traveled and assuming a 25 yard pool
4. Cleaning up USMS identification to identify athletes across years
5. Club sizes. USMS defines club sizes and scores based on those sizes, both by gender and total size by year
6. Summary stats for clubs by year Total distance swam

Club rankings by gender

Average distance traveled and 50 split by gender Average age by gender

Since this is headed for a Shiny app, where it will be displayed the goal will be to make information readable, which will involve some sacrifices. For example an athlete’s relative place will be a string in the form of “Athlete place of Total place” (i.e. “1 of 54”) rather than just a naked numeric.

## Age Groups

Since we already have an Age column making an age group column is a simple matter of using dplyr::case\_when to match the appropriate age range to the Age. We’ll do this using dplyr::between. between takes three arguments - a value x, a left and a right. If x >= left & x <= right then between returns TRUE. Otherwise between returns FASLE. The resulting code is perhaps long, and there are more compact ways to do this, but it’s also very readable, approximating written English. Readability is useful in the context of a blog entry and I’ll continue to prioritize it throughout.

Postal <- Postal %>% mutate(

Age\_Group = case\_when( between(Age, 0, 24) ~ "18-24",

|  |  |  |  |
| --- | --- | --- | --- |
| between(Age, | 25, | 29) | ~ "25-29", |
| between(Age, | 30, | 34) | ~ "30-34", |
| between(Age, | 35, | 39) | ~ "35-39", |
| between(Age, | 40, | 44) | ~ "40-44", |
| between(Age, | 45, | 49) | ~ "45-49", |
| between(Age, | 50, | 54) | ~ "50-54", |
| between(Age, | 55, | 59) | ~ "55-59", |
| between(Age, | 60, | 64) | ~ "60-64", |
| between(Age, | 65, | 69) | ~ "65-69", |
| between(Age, | 70, | 74) | ~ "70-74", |
| between(Age, | 75, | 79) | ~ "75-79", |
| between(Age, | 80, | 84) | ~ "80-84", |
| between(Age, | 85, | 89) | ~ "85-89", |
| between(Age, | 90, | 94) | ~ "90-94", |

between(Age, 95, 99)~ "95-99",

between(Age, 100, 104) ~ "100-104",

TRUE ~ "NA" # if for some reason Age doesn't match any of the above this will catch it and write the string 'NA'

)

) %>%

mutate(Age\_Group = factor(Age\_Group, levels = c("18-24", "25-29",

"30-34", "35-39", "40-44", "45-49", "50-54", "55-59", "60-64", "65-69",

"70-74", "75-79", "80-84", "85-89", "90-94", "95-99")))

# demonstration table Postal %>%

select(Place, Name, Distance, Year, Age, Age\_Group, Gender) %>% head(5) %>%

flextable\_style()

Place Name Distance Year Age Age\_Group Gender

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 Robert Wagner | 5450 | 2021 19 | 18-24 | M |
| 2 Elliott Roman | 5215 | 2021 24 | 18-24 | M |
| 3 William Kemp | 4700 | 2021 22 | 18-24 | M |
| 4 Dylan Ogle | 4250 | 2021 23 | 18-24 | M |
| 5 Gregory Willett | 4035 | 2021 24 | 18-24 | M |

## Athlete Relative Place

As discussed above the athlete place column will contain a string giving an individual athlete’s place within their age and gender category for a given year. The point is to provide a means of comparing two athletes in the Shiny app. If Athlete A and Athlete B both finished 6th, but Athlete A did so in a category with 50 entrants vs. only 10 in Athlete B’s category then that’s worth knowing when comparing them.

Here we’ll just paste an athlete’s Place, the string " of " and the maximumn place from that age group together.

Postal <- Postal %>%

group\_by(Gender, Age\_Group, Year) %>% # places are caluclated by gender, age group and year

mutate(Relative\_Place = paste(Place, max(Place, na.rm = TRUE), sep = " of ")) # use paste to build string

# demonstration table Postal %>%

select(Place, Name, Distance, Year, Age\_Group, Gender, Relative\_Place) %>%

head(5) %>% flextable\_style()

Place Name Distance Year Age\_Group Gender Relative\_Place

Place Name Distance Year Age\_Group Gender Relative\_Place

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 Robert Wagner | 5450 | 2021 18-24 | M | 1 of 5 |
| 2 Elliott Roman | 5215 | 2021 18-24 | M | 2 of 5 |
| 3 William Kemp | 4700 | 2021 18-24 | M | 3 of 5 |
| 4 Dylan Ogle | 4250 | 2021 18-24 | M | 4 of 5 |
| 5 Gregory Willett | 4035 | 2021 18-24 | M | 5 of 5 |

## Average 50 Split

I’m accustomed to looking at 50 splits in swimming, and I think it’s an interesting metric by which to evaluate ePostal results as well. An athlete’s split can be calculated by converting the time (1 hour), then multiplying that number of seconds by 50 and then multiplying again by reciprocal distance to get a number in units of seconds per 50 yards. We can then round and format the result to make it pleasant to look at.

Postal <- Postal %>%

mutate(Avg\_Split\_50 = (1 / Distance) \* 60 \* 60 \* 50) %>% # compute split

mutate(Avg\_Split\_50 = format(round(Avg\_Split\_50, 2), nsmall = 2)) # want two decimal places, even if the last one is a zero

# demonstration table Postal %>%

select(Place, Name, Distance, Year, Age\_Group, Gender, Avg\_Split\_50)

%>%

head(5) %>% flextable\_style()

Place Name Distance Year Age\_Group Gender Avg\_Split\_50

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 Robert Wagner | 5450 | 2021 18-24 | M | 33.03 |
| 2 Elliott Roman | 5215 | 2021 18-24 | M | 34.52 |
| 3 William Kemp | 4700 | 2021 18-24 | M | 38.30 |
| 4 Dylan Ogle | 4250 | 2021 18-24 | M | 42.35 |
| 5 Gregory Willett | 4035 | 2021 18-24 | M | 44.61 |

## USMS\_ID

The value in USMS\_ID has two parts. The first four characters, before the "-" vary year to year. The last five characters, after the "-" are an athlete’s permanent identification string, used to identify athletes without relaying on names. We discuss this records matching problem a lot around here, because athlete names aren’t a stable means of identification. Sometimes people get married and change their names. Sometimes people use nicknames. Things happen and the permanant portion of the USMS\_ID is a good way of handling things. Sadly ePostal results only include USMS\_ID after 2010, so we need to come up with something for pre-2011 results.

Here’s the plan. We’ll break off the permanent portion of USMS\_ID into a new column called Perm\_ID based on the location of "-" using str\_split\_fixed. Rows without a USMS\_ID will have an empty character string "" in Perm\_ID, which we’ll convert to NA with na\_if. Then we’ll group athletes by name. If any entry for that athlete, using that name, is from post-2010 they’ll have a Perm\_ID which we’ll be able to copy to all rows involving that name using the tidyr::fill function. It’s of course possible to have two athletes with the same name. We could be stricter here and group\_by name and club for example, but athletes do change clubs. There are other things we could try as well, like matching on age progression, but for right now we’re just going to accept that there might be some false matches.

Postal <- Postal %>%

mutate(Perm\_ID = str\_split\_fixed(USMS\_ID, "-", n = 2)[, 2], # only want the second part of the split

Perm\_ID = na\_if(Perm\_ID, "")) %>% # turn empty strings ""

into NAs group\_by(Name) %>%

tidyr::fill(Perm\_ID, .direction = "updown") # all instances of name will get non-NA value of Perm\_ID - assumes there's only one unique Perm\_ID for each name

Some athletes still don’t have a Perm\_ID though, because they don’t have a USMS\_ID listed.

# demonstration table Postal %>%

filter(Year == 1998) %>%

select(Place, Name, Year, USMS\_ID, Perm\_ID) %>% head(5) %>%

flextable\_style()

|  |  |  |  |
| --- | --- | --- | --- |
| Place | Name | Year USMS\_ID | Perm\_ID |
| 1 | Becky Crowe | 1998 |  |
| 2 | Johanna Hardin | 1998 |  |
| 3 | Sarah Anderson | 1998 |  |
| 4 | Sarah Baker | 1998 | 09RB9 |

Place Name Year USMS\_ID Perm\_ID

5 Jane Kelsey 1998

We’re going to make them fake Perm\_IDs. Each fake Perm\_ID will be a six character string of upper case letters. We’ll use this slightly different format (compared to the five character alphanumeric real Perm\_IDs) so that it’s possible to differentiate the fake from the real.

Each unique name without a real Perm\_ID needs a fake Perm\_ID. Let’s first collect those unique names.

Unique\_Names <- Postal %>% ungroup() %>%

select(Name, Perm\_ID) %>% # don't need all the columns, only these two

filter(is.na(Perm\_ID) == TRUE) %>% # want only rows where there isn't a Perm\_ID

unique() # don't need duplicates

Unique\_Names %>% head(5) %>% flextable\_style()

Name Perm\_ID

Glen Christiansen

Lou Hill

Sue Lyon

Masao Miyasaka

Karina Horton

Now we need a list of random strings, one for each name. Since we’re generating something random we’ll use set.seed to make it reproducible. Then we’ll use the stringi::stri\_rand\_strings function to generate a list of random strings with length 6. The length of the list will be the number of rows in unique\_names.

set.seed(1) # to make random strings reproducible

Unique\_Names <- Unique\_Names %>%

mutate(Perm\_ID = stringi::stri\_rand\_strings( # make random strings n = nrow(Unique\_Names), # number of random strings to make length = 6, # number of characters in each string

pattern = "[A-Z]" # what to make the string out of, in this case

all capital letters

))

# demonstration table Unique\_Names %>%

head(5) %>% flextable\_style()

Name Perm\_ID

Glen Christiansen GJOXFX

Lou Hill YRQBFE

Sue Lyon RJUMSZ

Masao Miyasaka JUYFQD

Karina Horton GKAJWI

Now we’ll join Postal and Unique\_Names back up by Name and use coalesce to get non-NA values of Perm\_ID for each row. Each Name, which hopefully means each athlete, now has a unique Perm\_ID.

Postal <- Postal %>%

left\_join(Unique\_Names, by = "Name") %>% # attach newly made strings back to original data frame based on name

mutate(Perm\_ID = coalesce(Perm\_ID.x, Perm\_ID.y)) %>% # use first non- na value between original data frame (x) and new data frame (y)

select(-Perm\_ID.x, -Perm\_ID.y) # don't need these columns any more

# demonstration table Postal %>%

filter(Year == 1998) %>%

select(Place, Name, Year, USMS\_ID, Perm\_ID) %>% head(5) %>%

flextable\_style()

|  |  |  |  |
| --- | --- | --- | --- |
| Place | Name | Year USMS\_ID | Perm\_ID |
| 1 | Becky Crowe | 1998 | XZPOKS |
| 2 | Johanna Hardin | 1998 | EOFXZG |
| 3 | Sarah Anderson | 1998 | YSAWZT |

Place Name Year USMS\_ID Perm\_ID

1. Sarah Baker 1998 09RB9
2. Jane Kelsey 1998 DHNSNX

## Club Sizes

USMS defines categories of club size for the ePostal as small, medium, large and extra large based on the number of athletes representing that club. Here we’ll count the number of athletes for each club with n and categorize appropriately with case\_when. It’s also interesting to look at participation by gender, so we’ll count the male and female athletes for each club using sum.

We’ll get new columns of the form Club\_Count (number of athletes in a club) and Club\_Size

(a factor with levels S, M, L, XL).

# total club size Postal <- Postal %>%

group\_by(Club, Year) %>% # working with clubs now, by year mutate(Club\_Count = n()) %>% # number of athletes in each club for

a given year

mutate(Club\_Size\_Combined = case\_when( # code in club sizes based on number of athletes

Club\_Count < 26 ~ "S", Club\_Count < 50 ~ "M", Club\_Count <= 100 ~ "L", TRUE ~ "XL"

)) %>%

mutate(Club\_Size\_Combined = factor(Club\_Size\_Combined, levels = c("S", "M", "L", "XL")))

# male club size Postal <- Postal %>%

group\_by(Club, Year) %>%

mutate(Club\_Count\_Male = sum(Gender == "M", na.rm = TRUE)) %>% # only want to count men this time

mutate(Club\_Size\_Male = case\_when( Club\_Count\_Male < 26 ~ "S", Club\_Count\_Male < 50 ~ "M", Club\_Count\_Male <= 100 ~ "L", TRUE ~ "XL"

)) %>%

mutate(Club\_Size\_Male = factor(Club\_Size\_Male, levels = c("S", "M", "L", "XL")))

# female club size Postal <- Postal %>%

group\_by(Club, Year) %>%

mutate(Club\_Count\_Female = sum(Gender == "W", na.rm = TRUE)) %>% # only want to count women this time

mutate(Club\_Size\_Female = case\_when(

Club\_Count\_Female < 26 ~ "S", Club\_Count\_Female < 50 ~ "M", Club\_Count\_Female <= 100 ~ "L", TRUE ~ "XL"

)) %>%

mutate(Club\_Size\_Female = factor(Club\_Size\_Female, levels = c("S", "M", "L", "XL")))

# demonstration table Postal %>%

select(Club, Year, Club\_Count, Club\_Size\_Combined, Club\_Size\_Male)

%>%

head(5) %>% flextable\_style()

Club Year Club\_Count Club\_Size\_Combined Club\_Size\_Male

|  |  |  |  |
| --- | --- | --- | --- |
| BSMT | 2021 46 | M | S |
| SKY | 2021 15 | S | S |
| SKY | 2021 15 | S | S |
| GS | 2021 3 | S | S |
| SKY | 2021 15 | S | S |

## Club Summary Stats

In addition to making comparisons between athletes we can also make comparisons between clubs.

**Club Total and Average Distance**

Clubs are scored based on the total distance swam by their memberships. Here we can

group\_by club can year, then add up the total distance each club swam with sum.

# total club stats Postal <- Postal %>%

mutate(Distance = na\_if(Distance, 0)) %>% # shouldn't be any distance NA values, but convert to zero if there are

group\_by(Club, Year) %>%

mutate(Total\_Distance\_Combined = sum(Distance, na.rm = TRUE)) %>% # add up distance for each club/year

mutate(Avg\_Distance\_Combined = Total\_Distance\_Combined / Club\_Count)

%>% # calculate avg distance per athlete mutate(Avg\_Distance\_Combined = round(Avg\_Distance\_Combined, 0))#

don't want decimal places

# male stats

Postal <- Postal %>% group\_by(Club, Year) %>%

mutate(Total\_Distance\_Male = sum(Distance[Gender == "M"], na.rm = TRUE)) %>%

mutate(Avg\_Distance\_Male = Total\_Distance\_Male / Club\_Count\_Male) %>% mutate(Avg\_Distance\_Male = round(Avg\_Distance\_Male, 0))

# female stats Postal <- Postal %>%

group\_by(Club, Year) %>%

mutate(Total\_Distance\_Female = sum(Distance[Gender == "W"], na.rm = TRUE)) %>%

mutate(Avg\_Distance\_Female = Total\_Distance\_Female / Club\_Count\_Female) %>%

mutate(Avg\_Distance\_Female = round(Avg\_Distance\_Female, 0))

# demonstration table Postal %>%

filter(Gender == "W") %>%

select(Club, Year, Total\_Distance\_Female, Avg\_Distance\_Female) %>% head(5) %>%

flextable\_style()

Club Year Total\_Distance\_Female Avg\_Distance\_Female

IM 2021 45420 3785

GS 2021 5850 2925

|  |  |
| --- | --- |
| UC08 2021 3400 | 3400 |
| CRUZ 2021 18475 | 4619 |
| 1776 2021 38340 | 4260 |
| **Club Rank** |  |

Clubs can be ranked within their size/gender categories by total distance swam in much the same way we ranked swimmers within their age/gender categories. Since their are only three categories (combined, male, female) I’m not going to bother about including category size like with did with individual athletes although it’s certainly possible to do so. Here we’ll just group\_by the appropriate Club\_size column and the year and then use dense\_rank to get the ranking for each club. dense\_rank is one of the the six dplyr ranking functions. It ranks values in a vector giving tied values a minimum rank *and* does not skip places as a result of ties. Here’s a demonstration, because some of you might not be familiar with dense\_rank.

Assume four clubs A-D, each of which swam some total distance.

distances <- c(10000, 15000, 15000, 20000)

names(distances) <- c("A", "B", "C", "D") distances

## A B C D ## 10000 15000 15000 20000

What we’d like to say is that club D is first, clubs B and C are tied for second and club A is third. Club B and C both getting second (rather than say, 2.5th, or randomly giving one 2nd and the other 3rd) is what I mean by “giving tied values a minimum rank”. Club A getting third, rather than forth (as the forth club on the list) is what I mean by “not skip[ping] places as a result of ties”. We also need to use desc because we want the clubs with the largest distance values to get the low...