If you’re only going to read one line of this post let it be this one: v0.6.0 of SwimmeR is now available from CRAN and it’s better now than ever before. But don’t just stop at one line though, grab that update and come along on this journey, deep into the heart of SwimmeR.

library(SwimmeR) library(dplyr) library(stringr) 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()

}

The rest of this post, which I promise will be super interesting and even have jokes, is about the tidy framework, inconsistencies in source material, subcategories and my failure to get all those things to play nicely together in the context of SwimmeR. I realize calling it a failure makes it sound like I was lying a few lines ago when I said that SwimmeR v0.6.0 is better than ever before, but I wasn’t.

Don’t jump to conclusions.

# Breaking Change

Before going any further though there’s one important, breaking, change. The output of swim\_parse has been modified, such that the Grade column is renamed to Age and the School column is renamed to Team. These are frankly overdue changes, which reflect the broader applicability of SwimmeR outside the American high school swimming arena I first developed it for. My apologies and please adjust your work flows.

# Tidy Data

Tidy is a way of organizing data. One of those philosophies is tidy data.

He also has a distinctive first name, so you’ll often see him referred to simply as Hadley, like he’s Kobe or Madonna or something, but that’s neither here nor there.

Tidy data has several benefits.

1. It’s easy to explain. Tidy data is simply data where each row is a distinct observation, or event (like a swim), and each column contains one and only one variable, like a name, or a time.
2. It’s easy to represent in a table, or even a spreadsheet.
3. It’s super easy to work with in R. This is arguably because of tidy data’s relative simplicity of

structure, but it’s also in some part because Hadley Wickham/RStudio release tons of free, excellent, packages that use (and enforce the norms of) tidy data.

In spite of what I see as tidy data’s real utility, bemoaning the influence of Hadley Wickham/RStudio is something of a sport in the R blog-o-sphere. I’ve got my hands way too full dealing with getting SwimmeR releases out, the International Swim League changing their results formats for no reason, and breaststrokers trying to cheat all the time to be complaining about someone releasing free tools. I need all the help I can get.

Back to SwimmeR v0.6.0 and its many improvements…

# Splits

SwimmeR can now read split times. Both the swim\_parse and swim\_parse\_ISL now have another argument, called splits. Setting splits = TRUE will direct each function to attempt to read in splits. The default is FALSE so splits won’t be collected unless you, the user, decide they should be.

In swim\_parse there’s another argument, splits\_length. This is the length of pool at which the splits are recorded. It’s usually 50 (the default).

file\_50 <- read\_results("<http://www.section1swim.com/Results/BoysHS/2020>

/Sec1/Single.htm")

df\_50 <- swim\_parse(file\_50, splits = TRUE) df\_50 %>%

filter(Event == "Men 100 Yard Butterfly") %>% select(Place, Name, Finals\_Time, Split\_50:Split\_100) %>% head(3) %>%

flextable\_style()

Place Name Finals\_Time Split\_50 Split\_100

1 Chung, Hudson H 51.98 24.82 27.16

2 Laidlaw, John F 52.15 24.91 27.24

3 Sakharuk, Nikita V 52.71 24.73 27.98

Sometimes though splits\_length should be set to 25, because splits are taken every 25 meters or yards.

file\_25 <- read\_results("https://github.com/gpilgrim2670/Pilgrim\_Data/raw/master

/SwimmeR%20Demo%20Files/7th-snsc-scm-2017-full-results.pdf")

df\_25 <- swim\_parse(file\_25, avoid = c("MR\\:", "NR\\:"), splits = TRUE, split\_length = 25)

df\_25 %>%

filter(Event == "Women 100 SC Meter IM") %>% select(Place, Name, Finals\_Time, Split\_25:Split\_100) %>% head(3) %>%

flextable\_style()

Place Name Finals\_Time Split\_25 Split\_50 Split\_75 Split\_100

1 \*Ling, Jessica 1:05.58 13.45 17.21 19.16 15.76

2 HO, Hui Ting Natalie 1:07.31 13.32 16.90 21.06 16.03

3 Cheong, Chloe 1:07.76 13.74 17.01 21.27 15.74

This is new, exciting, and undoubtedly the most requested feature, so there you go people – enjoy!

# Relay Swimmers

SwimmeR can also now capture relay swimmers, by setting the argument relay\_swimmers = TRUE inside either swim\_parse or swim\_parse\_ISL. Like splits, relay\_swimmers defaults to FALSE – using it is optional.

df\_relay <- swim\_parse(file\_50, relay\_swimmers = TRUE)

df\_relay %>%

filter(Event == "Men 400 Yard Freestyle Relay") %>%

select(Place, Team, Finals\_Time, Relay\_Swimmer\_1:Relay\_Swimmer\_4) %>% head(3) %>%

flextable\_style()

Place Team Finals\_Time Relay\_Swimmer\_1 Relay\_Swimmer\_2 Relay\_Swimmer\_3 Relay\_Swimmer\_4

Ardsley-

1 Hast-Edge- 3:12.34 Lee, Christian Dobbs-Irv

Andrews, Samuel T

Vincent, Connor J Pierce, Adrien T

Horace

2

Greeley

3:16.06 Chung, Hudson H Sakharuk, Nikita V McHugh, Luke P Laidlaw, John F

3 Wappingers 3:17.31 McGregor, Kyle T

McGregor, Matthew J

Holan, Steve

Smith, Sebastian L

# The Troubles

## Splits

“All that sounds great”, I hear you saying, “and your examples worked flawlessly!” Wrong, wrong,

wrong. I picked those examples as traps.



Not where you want a split

Here’s the problem. SwimmeR reads in data from sources, but it also imposes a structure on that data, a tidy structure in fact. In versions prior to v0.6.0 there was no issue with imposing a tidy structure, it didn’t chafe at all, because the source data SwimmeR was reading in was fundamentally tidy. Each row was a swim, each column the corresponding event, or athlete name, or final time.

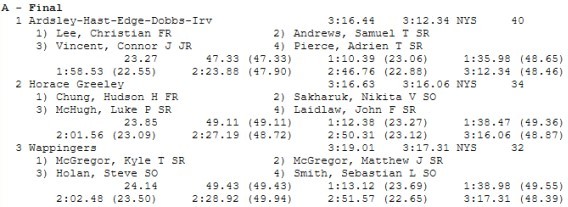
Splits are different though.

There are two general types of splits, cumulative and non-cumulative. Cumulative splits work like this: The first 50 meters of your race takes you 29.00 seconds. The second 50 meters takes you

31.00 seconds. Your cumulative split for the 50 is 29.00, and for the 100 is 61.00 seconds (1:01.00). You keep going, your third 50 takes you 31.50 seconds, so your cumulative split for the 150 is 29.00 plus 31.00 plus 31.50, for a total of 92.50 seconds (1:32.50). That’s all fine.

Non cumulative is the same, just without the adding. Your split for the 50 is still 29.00, for the next 50 (not the 100), it’s 31.00, for the third 50 (not the 150) it’s 31.50. This is fine too. SwimmeR prefers non-cumulative splits due to tidy principles – each split is it’s own variable, rather than the splits being collections (sums, or accumulations) of other splits/variables but if only cumulative splits are available SwimmeR will take them.

Either cumulative or non-cumulative splits wouldn’t be a problem, for SwimmeR or otherwise, and indeed both are often used together in swimming results, with non-cumulative printed inside parenthesis and cumulative outside. What about when the non-cumulative isn’t quite non- cumulative? Sounds daft right, but that’s the convention during relays. Take a look:



In the 1st place relay Christian Lee swims his first 50 in 23.27, then his next 50 in what? Not 47.33 – that’s his total time, despite it being wrapped in parenthesis. Then Samuel Andrews dives in and he swims the third 50 in 23.06, for a total time 1:10.39. He swims his second 50 in – again, not 48.65, that’s his cumulative 100. You see the problem. This problem is reproduced within the results from swim\_parse.

df\_50 %>%

filter(Event == "Men 400 Yard Freestyle Relay") %>% select(Place, Team, Finals\_Time, Split\_50:Split\_400) %>% head(3) %>%

flextable\_style()

Place Team Finals\_Time Split\_50 Split\_100 Split\_150 Split\_200 Split\_250 Split\_300 Split\_350 Split\_400

Ardsley-

1 Hast-Edge- Dobbs-Irv

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3:12.34 | 23.27 | 47.33 | 23.06 | 48.65 | 22.55 | 47.90 | 22.88 | 48.46 |
| 3:16.06 | 23.85 | 49.11 | 23.27 | 49.36 | 23.09 | 48.72 | 23.12 | 48.87 |

Horace

2

Greeley

3 Wappingers 3:17.31 24.14 49.43 23.69 49.55 23.50 49.94 22.65 48.39

I could of course convert the cumulative splits to non-cumulative.

df\_50 %>% rowwise() %>% mutate(

Split\_100 = case\_when(

str\_detect(Event, "Men 400 Yard Freestyle Relay") ~ as.character(sec\_format(Split\_100) - sec\_format(Split\_50)),

TRUE ~ Split\_100

),

Split\_200 = case\_when(

str\_detect(Event, "Men 400 Yard Freestyle Relay") ~ as.character(sec\_format(Split\_200) - sec\_format(Split\_150)),

TRUE ~ Split\_200

),

Split\_300 = case\_when(

str\_detect(Event, "Men 400 Yard Freestyle Relay") ~ as.character(sec\_format(Split\_300) - sec\_format(Split\_250)),

TRUE ~ Split\_300

),

Split\_400 = case\_when(

str\_detect(Event, "Men 400 Yard Freestyle Relay") ~ as.character(sec\_format(Split\_400) - sec\_format(Split\_350)),

TRUE ~ Split\_400

)

) %>%

filter(Event == "Men 400 Yard Freestyle Relay") %>% select(Place, Team, Finals\_Time, Split\_50:Split\_400) %>% head(3) %>%

flextable\_style()

Place Team Finals\_Time Split\_50 Split\_100 Split\_150 Split\_200 Split\_250 Split\_300 Split\_350 Split\_400

Ardsley-

1 Hast-Edge- Dobbs-Irv

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3:12.34 | 23.27 | 24.06 | 23.06 | 25.59 | 22.55 | 25.35 | 22.88 | 25.58 |
| 3:16.06 | 23.85 | 25.26 | 23.27 | 26.09 | 23.09 | 25.63 | 23.12 | 25.75 |

Horace

2

Greeley

3 Wappingers 3:17.31 24.14 25.29 23.69 25.86 23.50 26.44 22.65 25.74

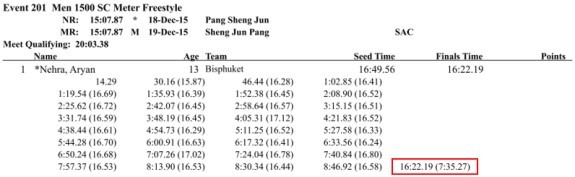
I can do the conversion here, and I could have built the conversion into swim\_parse and swim\_parse\_ISL, but there’s a tension, between the extremely useful tidy framework on one hand, and fidelity to the source material on the other. SwimmeR thus far gives users the choice. You can choose to read in splits (and remember, the default is not to), and then you can choose how you treat splits if you decide to read them in. You can choose between tidiness and fidelity.

I may bundle the above code, for converting split types, into a future version of SwimmeR but if I do its use will also be at the discretion of the user.

Even deeper is the fact that (for example), the third 50 of a 200 Freestyle and the third 50 of a 200 Freestyle relay are both recorded in the same column (Split\_150) but they’re fundamentally different, because the third 50 on a relay begins with a relay pickup (a start from the blocks), and involves a fresh swimmer, where the third 50 in the individual event starts from a turn and involves the same swimmer who swam the first two 50s. One column contains two variables. That’s a no-no. I could put relay splits in their own columns, but I’d have to break them up by relay type too. The 2nd 50 of a 200 Freestyle Relay is almost always done with the front crawl stroke (freestyle), but the 2nd 50 of a 200 Medley Relay is required to be swam breaststroke – different animals all together.

Differentiating them means a lot lot lot of columns/variables.

And what if the length at which splits are taken varies within a meet, or even within a race? Have a look at this:



The splits are by 25 for the first 800m of this 1500m freestyle, but for the last 700 there’s only one split, for the whole 700m. swim\_parse faithfully reproduces this, except that it doesn’t know that suddenly the split length has changed mid-race. The result is a column labeled Split\_825 that isn’t actually the 825m split.

df\_25 %>%

filter(Event == "Men 1500 SC Meter Freestyle") %>% select(Place, Name, Finals\_Time, Split\_700:Split\_825) %>% head(3) %>%

flextable\_style()

Place Name Finals\_Time Split\_700 Split\_725 Split\_750 Split\_775 Split\_800 Split\_825

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 \*Nehra, Aryan | 16:22.19 | 16.80 | 16.53 | 16.53 | 16.44 | 16.58 | 7:35.27 |
| 2 Cheng, Jimmy | 16:49.64 | 17.35 | 16.63 | 16.76 | 17.21 | 17.17 | 7:53.87 |
| 3 Abraham, Levente | 17:06.60 | 17.64 | 17.51 | 17.63 | 17.48 | 17.65 | 8:03.36 |

There’s nothing I can do here – I can’t recreate splits that aren’t included in some form in the source data. You’ll just have to know what’s going on in your results.

For further discussion of potential issues with splits please see vignette("SwimmeR").

## Relay Swimmers

Relay swimmers aren’t anywhere near as troublesome as splits, but they’re still in tension with tidy principles. Let’s say you were interested in what races a particular athlete had participated in. It’s easy to filter or in some other way subset the data based on Name. With relay\_swimmers = TRUE though that athlete’s name might be in Name or it might be in one of the relay swimmer columns. Having one variable (athlete name) in multiple columns is not tidy.

I could put relay swimmers in the names columns, but then what to do about their times? They don’t have a Finals\_Time in the normal sense, only their splits, and as you’ve just read that’s a whole other mess.

Again my approach here has been reasonable fidelity to the results, and letting users decide if they want to opt into tidy troubles (by affirmatively setting relay\_swimmers = TRUE).