Package ‘SwimmeR’

July 11, 2020

Title Data Import, Cleaning, and Conversions for Swimming Results

Version 0.3.1

Description There are two goals for 'SwimmeR' as presently constructed. The first is reading in swimming results from html or pdf sources and returning tidy dataframes. The second is working with the resulting data. To this end 'SwimmeR' converts swimming times (performances) between the computationally useful
format of seconds, reported to the 100ths place (e.g. 95.37), and the conventional reporting format (1:35.37) used in the swimming community, as well as providing tools for assigning team names etc.
Additionally 'SwimmeR' has functions for drawing single-elimination brackets and also converts times between the various pool sizes used in competitive swimming, namely 50m length (LCM), 25m length (SCM)
and 25y length (SCY).

License MIT + file LICENSE

Imports purrr, dplyr, stringr, tibble, utils, rvest, pdftools, ggplot2, scales, magrittr, xml2

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Suggests testthat (>= 2.1.0), knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Greg Pilgrim [aut, cre] (<https://orcid.org/0000-0001-7831-442X>),
Caitlin Baldwin [ctb]

Maintainer Greg Pilgrim <gpilgrim2670@gmail.com>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2020-07-11 07:00:02 UTC

R topics documented:

`course_convert` *Swimming Course Convertor*

Description

Used to convert times between Long Course Meters, Short Course Meters and Short Course Yards

Usage

```
course_convert(time, event, course, course_to)
```

Arguments

`time`	A time, or vector of times to convert. Can be in either seconds (numeric, `95.97`) format or swim (character, `"1:35.97"`) format
`event`	The event swum as `"100 Fly"`, `"200 IM"`, `"400 Free"`, `"50 Back"`, `"200 Breast"` etc.
`course`	The course in which the time was swum as `"LCM"`, `"SCM"` or `"SCY"`
`course_to`	The course to convert the time to as `"LCM"`, `"SCM"` or `"SCY"`

Value

returns the `time` for a specified `event` and `course` converted to a time for the specified `course_to` in swimming format

Note

Relays are not presently supported.

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

References

Uses the USA swimming age group method described here: https://support.teamunify.com/en/articles/260

Examples

```
course_convert(time = "1:35.93", event = "200 Free", course = "SCY", course_to = "LCM")
course_convert(time = 95.93, event = "200 Free", course = "scy", course_to = "lcm")
course_convert(time = 53.89, event = "100 Fly", course = "scm", course_to = "scy")
```

---

course_convert_DF *Course converter, returns dataframe*

Description

Used to convert times between Long Course Meters, Short Course Meters and Short Course Yards, returns dataframe

Usage

```
course_convert_DF(time, event, course, course_to)
```

Arguments

`time`	A time, or vector of times to convert. Can be in either seconds (numeric, `95.97`) format or swim (character, `"1:35.97"`) format
`event`	The event swum as `"100 Fly"`, `"200 IM"`, `"400 Free"`, `"50 Back"`, `"200 Breast"` etc.
`course`	The course in which the time was swum as `"LCM"`, `"SCM"` or `"SCY"`
`course_to`	The course to convert the time to as `"LCM"`, `"SCM"` or `"SCY"`

Value

This function returns a `data.frame` including columns:

- time
- course
- course_to
- event
- Time_Converted_sec
- Time_Converted_mmss

Note

Relays are not presently supported.

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

References

Uses the USA swimming age group method described here https://support.teamunify.com/en/articles/260

Examples

```
course_convert_DF(time = "1:35.93", event = "200 Free", course = "SCY", course_to = "LCM")
course_convert_DF(time = 95.93, event = "200 Free", course = "scy", course_to = "lcm")
course_convert_DF(time = 53.89, event = "100 Fly", course = "scm", course_to = "scy")
```

---

draw_bracket *Creates a bracket for tournaments involving 5 to 64 teams, single elimination*

Description

Will draw a single elimination bracket for the appropriate number of teams, inserting first round byes for higher seeds as needed

Usage

```
draw_bracket(
  teams,
  title = "Championship Bracket",
  text_size = 0.7,
  round_two = NULL,
  round_three = NULL,
  round_four = NULL,
  round_five = NULL,
  round_six = NULL,
  champion = NULL
)
```

Arguments

teams	a list of teams, ordered by desired seed, to place in bracket. Must be between 5 and 64 inclusive. Teams must have unique names
title	bracket title

`text_size`	number passed to `cex` in plotting
`round_two`	a list of teams advancing to the second round (need not be in order)
`round_three`	a list of teams advancing to the third round (need not be in order)
`round_four`	a list of teams advancing to the forth round (need not be in order)
`round_five`	a list of teams advancing to the fifth round (need not be in order)
`round_six`	a list of teams advancing to the fifth round (need not be in order)
`champion`	the name of the overall champion team (there can be only one)

Value

a plot of a bracket for the teams, with results and titles as specified

Author(s)

Greg Pilgrim <`gpilgrim2670@gmail.com`>

References

based on `draw.bracket` from the seemingly now defunct `mRchmadness` package by Eli Shayer and Saber Powers and used per the terms of that package's GPL-2 license

Examples

```
## Not run:
teams <- c("red", "orange", "yellow", "green", "blue", "indigo", "violet")
round_two <- c("red", "yellow", "blue", "indigo")
round_three <- c("red", "blue")
champion <- "red"
draw_bracket(teams = teams,
             round_two = round_two,
             round_three = round_three,
             champion = champion)

## End(Not run)
```

`fold` *Fold a vector onto itself*

Description

Fold a vector onto itself

Usage

```
fold(x, block.size = 1)
```

Arguments

`x`	a vector
`block.size`	the size of groups in which to block the data

Value

a new vector in the following order: first block, last block, second block, second-to-last block, ...

Author(s)

sspowers

References

from the seemingly now defunct `mRchmadness` package by Eli Shayer and Saber Powers and used per the terms of that package's GPL-2 license

---

`get_mode` *Find the mode (most commonly occurring) element of a list*

Description

Determines which element of list appears most frequently. Based on `base::which.max`, so if multiple values appear with the same frequency will return the first one. Ignores `NA` values. In the context of swimming data is often used to clean team names, as in the Lilly King example below.

Usage

```
get_mode(x, type = "first")
```

Arguments

`x`	A list. `NA` elements will be ignored.
`type`	a character string of either `"first"` or `"all"` which determines behavior for ties. Setting `type = "first"` (the default) will return the element that appears most often and appears first in list `x`. Setting `type = "all"` will return all elements that appear most frequently.

Value

the element of `x` which appears most frequently. Ties go to the lowest index, so the element which appears first.

Author(s)

Greg Pilgrim <`gpilgrim2670@gmail.com`>

Examples

```
a <- c("a", "a", "b", "c")
get_mode(a)
ab <- c("a", "a", "b", "b", "c") # returns "a", not "b"
get_mode(ab)
#' ab <- c("a", "a", "b", "b", "c") # returns "a" and "b"
get_mode(ab, type = "all")
a_na <- c("a", "a", NA, NA, "c")
get_mode(a_na)
numbs <- c(1, 1, 1, 2, 2, 2, 3, NA)
get_mode(numbs, type = "all")

Name <- c(rep("Lilly King", 5))
Team <- c(rep("IU", 2), "Indiana", "IUWSD", "Indiana University")
df <- data.frame(Name, Team, stringsAsFactors = FALSE)
df$Team <- get_mode(df$Team)
```

`King200Breast` *Results for Lilly King's 200 Breaststrokes*

Description

Lilly King's 200 Breaststroke swims from her NCAA career

Usage

```
data(King200Breast)
```

Format

An object of class `"data.frame"`

Source

NCAA Times Database

`mmss_format` *Formatting seconds as mm:ss.hh*

Description

Takes a numeric item or list of numeric items representing seconds (e.g. 95.37) and converts to a character string or list of strings in swimming format ("1:35.37").

Usage

```
mmss_format(x)
```

Arguments

x	A number of seconds to be converted to swimming format

Value

the number of seconds x converted to conventional swimming format mm:ss.hh

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

sec_format mmss_format is the reverse of sec_format

Examples

```
mmss_format(95.37)
mmss_format(200.95)
mmss_format(59.47)
mmss_format(c(95.37, 200.95, 59.47, NA))
```

Read_Results *Reads swimming and diving results into a list of strings in preparation for parsing with* Swim_Parse

---

Description

Outputs list of strings to be processed by Swim_Parse

Usage

```
Read_Results(file, node = NULL)

read_results(file, node = NULL)
```

Arguments

file	a .pdf or .html file (could be a url) where containing swimming results. Must be formatted in a "normal" fashion - see vignette
node	a CSS node where html results are stored. Required for html results.

Value

returns a list of strings containing the information from `x`. Should then be parsed with `swim_parse`

Author(s)

Greg Pilgrim <`gpilgrim2670@gmail.com`>

See Also

`read_results` is meant to be followed by `swim_parse`

Examples

```
## Not run: read_results("http://www.nyhsswim.com/Results/Boys/2008/NYS/Single.htm", node = "pre")
```

`sec_format` *Formatting mm:ss.tt times as seconds*

Description

Takes a character string (or list) representing time in swimming format (e.g. 1:35.37) and converts it to a numeric value (95.37) or a list of values representing seconds.

Usage

```
sec_format(x)
```

Arguments

`x`	A character vector of time(s) in swimming format (e.g. 1:35.93) to be converted to seconds (95.93)

Value

returns the value of the string `x` which represents a time in swimming format (mm:ss.hh) and converts it to seconds

Author(s)

Greg Pilgrim <`gpilgrim2670@gmail.com`>

See Also

`mmss_format` `sec_format` is the reverse of `mmss_format`

Examples

```
sec_format("1:35.93")
sec_format("16:45.19")
sec_format("25.43")
sec_format(c("1:35.93", "16:45.19", "25.43"))
sec_format(c("1:35.93", "16:45.19", NA, "25.43"))
```

`sec_format_helper` *Helper function for formatting mm:ss.hh times as seconds*

Description

Helper function for formatting mm:ss.hh times as seconds

Usage

```
sec_format_helper(x)
```

Arguments

`x`	A character vector of time(s) in swimming format (e.g. 1:35.93) to be converted to seconds (95.93)

Author(s)

Greg Pilgrim <`gpilgrim2670@gmail.com`>

`SwimmeR` *SwimmeR: A package for working with swimming times*

Description

There are two goals for 'SwimmeR' as presently constructed. The first is reading in swimming results from html or pdf sources and returning tidy dataframes. The second is working with the resulting data. To this end 'SwimmeR' converts swimming times (performances) between the computationally useful format of seconds, reported to the 100ths place (e.g. 95.37), and the conventional reporting format (1:35.37) used in the swimming community, as well as providing tools for assigning team names etc. Additionally 'SwimmeR' has functions for drawing single-elimination brackets and also converts times between the various pool sizes used in competitive swimming, namely 50m length (LCM), 25m length (SCM) and 25y length (SCY).

Author(s)

Greg Pilgrim <`gpilgrim2670@gmail.com`>

`Swim_Parse`	*Formats swimming and diving data read in with Read_Results into dataframe*

Description

Takes the output of read_results and cleans it, yielding a dataframe of swimming results

Usage

```
Swim_Parse(
  file,
  avoid = avoid_default,
  typo = typo_default,
  replacement = replacement_default
)

swim_parse(
  file,
  avoid = avoid_default,
  typo = typo_default,
  replacement = replacement_default
)
```

Arguments

`file`	output from `read_results`
`avoid`	a list of strings. Rows in `x` containing these strings will not be included. For example "Pool:", often used to label pool records, could be passed to `avoid`. The default is `avoid_default`, which contains many strings similar to "Pool:", such as "STATE:" and "Qual:". Users can supply their own lists to `avoid`.
`typo`	a list of strings that are typos in the original results. `swim_parse` is particularly sensitive to accidental double spaces, so "Central High School", with two spaces between "Central" and "High" is a problem, which can be fixed. Pass "Central High School" to `typo`. Unexpected commas as also an issue, for example "Texas, University of" should be fixed using `typo` and `replacement`
`replacement`	a list of fixes for the strings in `typo`. Here one could pass "Central High School" (one space between "Central" and "High") and "Texas" to `replacement` fix the issues described in `typo`

Value

returns a dataframe with columns `Name`, `Place`, `Grade`, `School`, `Prelims_Time`, `Finals_Time`, `Points`, & `Event`. Note all swims will have a `Finals_Time`, even if that time was actually swam in the prelims (i.e. a swimmer did not qualify for finals). This is so that final results for an event can be generated from just one column.

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

swim_parse must be run on the output of read_results

Examples

```
## Not run:
swim_parse(read_results("http://www.nyhsswim.com/Results/Boys/2008/NYS/Single.htm", node = "pre"),
 typo = c("-1NORTH ROCKL"), replacement = c("1-NORTH ROCKL"))

## End(Not run)
## Not run:
swim_parse(read_results("inst/extdata/Texas-Florida-Indiana.pdf"),
 typo =  c("Indiana  University", ", University of"), replacement = c("Indiana University", ""))

## End(Not run)
```

Index