Package 'SwimmeR'

August 28, 2020

Title Data Import, Cleaning, and Conversions for Swimming Results

Version 0.4.1

Description The goal for of the 'SwimmeR' package is to provide means of acquiring, and then analyzing, data from swimming (and diving) competitions. To that end 'SwimmeR' allows results to be read in from .html sources, like 'Hy-Tek' real time results pages, '.pdf' files, and now (on a development basis) '.hy3' files. Once read in, 'SwimmeR' can convert 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. 'SwimmeR' can also score meets in a variety of formats with user defined point values, convert times between courses ('LCM', 'SCM', 'SCY') and draw single elimination brackets, as well as providing a suite of tools for working cleaning swimming data. This is a developmental package, not yet mature.

License MIT + file LICENSE

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

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

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 2020-08-28 19:50:08 UTC

R topics documented:

`add_row_numbers` *Add row numbers to raw results*

Description

Takes the output of `read_results` and adds row numbers to it

Usage

```
add_row_numbers(text)
```

Arguments

`text`	output from `read_results`

Value

returns a dataframe with event names and row numbers to eventually be recombined with swimming results inside `swim_parse`

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

add_row_numbers is a helper function inside swim_parse

---

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")
```

---

dive_place *Adds places to diving results*

Description

Places are awarded on the basis of score, with highest score winning. Ties are placed as ties (both athletes get 2nd etc.)

Usage

```
dive_place(df, max_place)
```

Arguments

df	a dataframe with results from swim_parse, including only diving results (not swimming)
max_place	highest place value that scores

Value

df modified so that places have been appended based on diving score

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

dive_place is a helper function used inside of results_score

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)
```

`event_parse` *Pulls out event labels from text*

Description

Locates event labels in text of results output from `read_results` and their associated row numbers. The resulting dataframe is joined back into results to include event names

Usage

```
event_parse(text)
```

Arguments

`text`	output from `read_results`

Value

returns a dataframe with event names and row numbers to eventually be recombined with swimming results inside `swim_parse`

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

`event_parse` is a helper function inside `swim_parse`

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)
```

---

hy3_places *Helper for reading prelims and finals places from Hy-Tek .hy3 files*

Description

Used to pull prelims and finals places from .hy3 files as part of parsing them.

Usage

```
hy3_places(
  file,
  type = c("prelims", "relay_prelims", "finals", "relay_finals")
)
```

Arguments

file	an output of read_results, from an .hy3 file
type	type of times, either "prelims", "relay_prelims", "finals" or "relay_finals"

Value

a dataframe where column 1 is times and column 2 is row number

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

hy3_places is run inside of parse_hy3

---

hy3_times *Helper for reading prelims and finals times from Hy-Tek .hy3 files*

Description

Used to pull prelims and finals times from .hy3 files as part of parsing them.

Usage

```
hy3_times(file, type = c("prelims", "relay_prelims", "finals", "relay_finals"))
```

Arguments

file	an output of read_results, from an .hy3 file
type	type of times, either "prelims", "relay_prelims", "finals" or "relay_finals"

Value

a dataframe where column 1 is times and column 2 is row number

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

hy3_times is run inside of parse_hy3

interleave_results *Helper for reading interleaving prelims and finals results*

Description

Interleaves times or places based on row number ranges.

Usage

```
interleave_results(entries, results, type = c("individual", "relay"))
```

Arguments

entries	a dataframe containing columns for minimum and maximum row number (usually 'Row_Min' and 'Row_Max'). Times or places will be interleaved into this dataframe.
results	a dataframe containing times (or places) in column 1 (or other values to be interleaved) and row numbers in column 2 (usually 'Row_Numb').
type	either "individual" or "relay"

Value

a modified version of 'entries' with values from 'results' interleaved on the basis of row number

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

interleave_results is a helper function used in parse_hy3

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))
```

parse_hy3 *Parses Hy-Tek .hy3 files*

Description

Helper function used inside ‘swim_parse‘ for dealing with Hy-Tek .hy3 files. Can have more columns than other ‘swim_parse‘ outputs, because .hy3 files can contain more data

Usage

```
parse_hy3(
  file,
  avoid = avoid_minimal,
  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, & Event. May also contain Seed_Time, USA_ID, and/or Birthdate. 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

parse_hy3 must be run on the output of read_results

parse_hy3 runs inside of swim_parse

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 file. 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")
```

`results_score` *Scores a swim meet*

Description

Used to add a `Points` column with point values for each place. Can either score "timed finals" type meets where any athlete can get any place, or "prelims-finals", type meets, where placing is restricted by prelim performance.

Usage

```
results_score(
  results,
  events,
  meet_type = c("timed_finals", "prelims_finals"),
  lanes = c(4, 6, 8, 10),
  scoring_heats = c(1, 2, 3),
  point_values
)
```

Arguments

`results`	an output from `swim_parse`
`events`	list of events
`meet_type`	how to score based on `timed_finals`, where any place is possible, or `prelims_finals` where athletes are locked into heats for scoring purposes
`lanes`	number of lanes in to the pool, for purposes of heat
`scoring_heats`	number of heats which score (if 1 only A final scores, if 2 A and B final score etc.)
`point_values`	a list of point values for each scoring place

Value

results with point values in a column called `Points`

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

Examples

```
## Not run:
file <-
system.file("extdata", "BigTen_WSWIM_2018.pdf", package = "SwimmeR")
BigTenRaw <- read_results(file)

BigTen <- swim_parse(
```

```
  BigTenRaw,
  typo = c(
    "^\\s{1,}\\*",
    "^\\s{1,}(\\d{1,2})\\s{2,}",
    ",\\s{1,}University\\s{1,}of",
    "University\\s{1,}of\\s{1,}",
    "\\s{1,}University",
    "SR\\s{2,}",
    "JR\\s{2,}",
    "SO\\s{2,}",
    "FR\\s{2,}"
  ),
  replacement = c(" ",
                  "  \\1 ",
                  "", "", "",
                  "SR ",
                  "JR ",
                  "SO ",
                  "FR "),
  avoid = c("B1G", "Pool")
)

BigTen <- BigTen %>%
  dplyr::filter(
    stringr::str_detect(Event, "Time Trial") == FALSE,
    stringr::str_detect(Event, "Swim-off") == FALSE
  ) %>%
  dplyr::mutate(School = dplyr::case_when(School == "Wisconsin, Madi" ~ "Wisconsin",
                                          TRUE ~ School))

# begin results_score portion
df <- BigTen %>%
  results_score(
    events = unique(BigTen$Event),
    meet_type = "prelims_finals",
    lanes = 8,
    scoring_heats = 3,
    point_values = c(
     32, 28, 27, 26, 25, 24, 23, 22, 20, 17, 16, 15, 14, 13, 12, 11, 9, 7, 6, 5, 4, 3, 2, 1)
  )

## End(Not run)
```

`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

The goal for of the 'SwimmeR' package is to provide means of acquiring, and then analyzing, data from swimming (and diving) competitions. To that end 'SwimmeR' allows results to be read in from .html sources, like 'Hy-Tek' real time results pages, '.pdf' files, and now (on a development basis) '.hy3' files. Once read in, 'SwimmeR' can convert 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. 'SwimmeR' can also score meets in a variety of formats with user defined point values, convert times between courses ('LCM', 'SCM', 'SCY') and draw single elimination brackets, as well as providing a suite of tools for working cleaning swimming data. This is a developmental package, not yet mature.

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

Swim_Parse *Formats swimming and diving data read with* read_results *into a dataframe*

Description

Takes the output of read_results and cleans it, yielding a dataframe of swimming (and diving) 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 & DQ. 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)
```

swim_place *Adds places to swimming results*

Description

Places are awarded on the basis of time, with fastest (lowest) time winning. Ties are placed as ties (both athletes get 2nd etc.)

Usage

```
swim_place(df, max_place)
```

Arguments

df	a dataframe with results from swim_parse, including only swimming results (not diving)
max_place	highest place value that scores

Value

df modified so that places have been appended based on swimming time

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

swim_place is a helper function used inside of results_score

tie_rescore *Rescore to account for ties*

Description

Rescoring to average point values for ties. Ties are placed as ties (both athletes get 2nd etc.)

Usage

```
tie_rescore(df, point_values, lanes)
```

Arguments

df	a dataframe with results from swim_parse, with places from swim_place and/or dive_place
point_values	a named list of point values for each scoring place
lanes	number of scoring lanes in the pool

Value

df modified so that places have been appended based on swimming time

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

See Also

`tie_rescore` is a helper function used inside of `results_score`

`%notin%` *"Not in" function*

Description

The opposite of ‘

Usage

```
x %notin% y

x %!in% y
```

Arguments

x	a value
y	a list of values

Value

a ‘TRUE‘ or ‘FALSE‘

Author(s)

Greg Pilgrim <gpilgrim2670@gmail.com>

Examples

```
"a" %!in% c("a", "b", "c")
"a" %notin% c("b", "c")
```

Index