Package: volleysim (via r-universe)

August 21, 2024

Title Volleyball Match Simulation Tools

Version 0.3.7

Description Functions for simulating volleyball match outcomes.

Depends R (>= 3.3.0)

URL https://volleysim.openvolley.org,
https://github.com/openvolley/volleysim

BugReports https://github.com/openvolley/volleysim/issues

Imports assertthat, dplyr, magrittr, markovchain, methods, rlang

Suggests datavolley, testthat

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Remotes openvolley/datavolley

Repository https://openvolley.r-universe.dev

RemoteUrl https://github.com/openvolley/volleysim

RemoteRef HEAD

RemoteSha 211ee9e20af5bb5f7c1756c44d5a15b02c1ffeec

Contents

volleysim **volleysim**

Description

Functions for simulating volleyball match outcomes.

vs_estimate_rates *Estimate parameters required for simulation*

Description

Estimate parameters required for simulation

Usage

```
vs_estimate_rates(
  x,
  target_team,
  by = "none",
  moderate = TRUE,
  process_model = "all"
)
```

Arguments

x	datavolleyplays: the plays component of a datavolley object as returned by datavolley::dv_read()
target_team	string: the team name to calculate rates for. If missing or NULL, rates will be calculated across the entire data.frame x. If target_team is "each", rates will be calculated for each team separately
by	string: grouping to calculate rates by. Either "none" (calculate whole-data set rates), "match" (by match), or "set" (by match and set)
moderate	logical: if TRUE, apply some checks to attempt to ensure that the estimated rates are reasonable. Currently these checks include: • setting error rates are limited to a maximum of 5%. Some scouts do not include setting actions, except where they are errors or otherwise exceptional, which can lead to unrealistic estimates of setting error rates
process_model	string: estimate the rates required for which process model? Either "sideout", "phase", or "all"

Value

A tibble, with columns match_id (if by is "match"), set_number (if by is "set"), and (if `target_team` is "each") "team". The remaining columns depend on the `process_model`: for "sideout", column sideout. If `process_model` is "phase" then additionally the columns serve_ace, serve_error, rec_loss_other, rec_att_error, rec_att_kill, rec_att_replayed, rec_no_att, rec_block, trans_loss_other, trans_att_error, trans_att_kill, trans_att_replayed, trans_no_att, and trans_block

See Also

`vs_simulate_set()`

Examples

```
## Not run:
  library(datavolley)
  x <- dv_read(dv_example_file())
  rates <- vs_estimate_rates(x, target_team = "each")

  vs_simulate_set(rates) ## simulate a single set
  vs_simulate_match(rates) ## simulate a match
  ## so given the performances of the two teams during that match, we expect
  ##  that the home team should have won, with 3-0 being the most likely scoreline

  ## compare to the actual match result
  summary(x)

## End(Not run)
```

---

vs_example_file *Example DataVolley files provided as part of the volleysim package*

Description

Example DataVolley files provided as part of the volleysim package

Usage

```
vs_example_file(choice = 1)
```

Arguments

`choice`	numeric: which data file to return? • 1 - the 2020 Austrian Women's Volley Cup played between Hartberg and UVC Graz

Value

path to the file

References

The example data files came from https://www.volleynet.at/dvdownload/information/f-Damen/

See Also

datavolley::dv_read()

Examples

```
## Not run:
  myfile <- vs_example_file()
  x <- datavolley::dv_read(myfile)
  summary(x)

## End(Not run)
```

---

vs_match_win_probability *Create a win probability graph for a match*

Description

Create a win probability graph for a match

Usage

```
vs_match_win_probability(
  pbp,
  so,
  go_to = 25,
  go_to_tiebreak = 15,
  max_sets = 5,
  show_plot = TRUE,
  home_color = "blue",
  visiting_color = "darkred"
)
```

Arguments

pbp	data frame: a data frame containing the set number, home team, visiting team, serving team, point-winning team, home team score, and visiting team score at the end of each point, easiest to obtain by subsetting the plays component of a datavolley object as returned by datavolley::dv_read() to only include rows where point == TRUE
so	integer: a two-element vector of sideout rates for the home team and visiting team, easiest to obtain using vs_estimate_rates()

go_to	integer: the minimum score that must be reached to end a non-tiebreaker set (typically 25 for indoor volleyball in sets 1 to 4, or 21 in beach volleyball)
go_to_tiebreak	integer: the minimum score that must be reached to end a tiebreaker set (typically 15)
max_sets	integer: the maximum number of sets that can be played, either 3 or 5
show_plot	logical: if TRUE, produce a graph showing the home team's win probability at each point in the match
home_color	string: the color used to indicate points in the match where the home team is favored to win
visiting_color	string: the same as home_color, but for the visiting team

Value

A data frame containing the home team's probability of winning the set (set_probs) and match (match_probs) at each point in the set. The first row of the data frame refers to the start of the match (0-0, Set 1).

Examples

```
## Not run:
  library(datavolley)
  x <- dv_read(vs_example_file())
  sideout_rates <- vs_estimate_rates(x, target_team = "each")$sideout
  play_by_play <- subset(plays(x), point)

 vs_match_win_probability(play_by_play, sideout_rates)  ## data frame is not printed to console
  ## but can be stored in a variable
  match_win_probs <- vs_match_win_probability(play_by_play, sideout_rates)

## End(Not run)
```

---

vs_simulate_match *Simulate a volleyball match*

Description

Simulate a volleyball match using either best-of-5 or best-of-3 scoring

Usage

```
vs_simulate_match(
  rates,
  process_model = "phase",
  serving = NA,
  serving5 = NA,
  max_sets = 5,
  go_to = 25,
```

```
  go_to5 = 15,
  point_margin = 2L,
  point_margin5 = 2L,
  n = 2000,
  simple = TRUE,
  method = "theoretical"
)

vs_simulate_match_mc(...)

vs_simulate_match_theor(...)

vs_simulate_match_beach(...)
```

Arguments

rates	list: A two-element list, each element of which is a set of rates as returned by vs_estimate_rates
process_model	string: either "sideout" or "phase". See vs_estimate_rates()
serving	logical: if TRUE, team 1 will serve first in the match. If NA, the team serving first will be chosen at random
serving5	logical: if TRUE, team 1 will serve first in the tiebreaking set (if the match gets that far). If NA, the team serving first in that set will be chosen at random
max_sets	integer: the maximum number of sets to be played (either 3 or 5)
go_to	integer: the minimum score that must be reached to end the set (typically 25 for indoor volleyball in sets 1 to 4, 15 in set 5, or 21 in beach volleyball)
go_to5	integer: the minimum score that must be reached to end the tiebreaker set (typically 15 for indoor volleyball)
point_margin	integer: the minimum score difference in order to win the set. Only applicable to method "monte carlo". For method "theoretical" a two-point margin is always used
point_margin5	integer: the minimum score difference in order to win the tiebreaker set. Only applicable to method "monte carlo". For method "theoretical" a two-point margin is always used
n	integer: the number of simulations to run. Only applicable to method "monte carlo"
simple	logical: if TRUE, just return the probability of team winning and the probabilities of each possible set score. If FALSE, return extra details in a named list. The details will differ between method = "monte carlo" and method = "theoretical"
method	string: the simulation method to use. Either "monte carlo" or "theoretical". Details TBD
...	parameters as for vs_simulate_match. vs_simulate_match_theor and vs_simulate_match_mc are convenience functions for vs_simulate_match(..., method = "theoretical") and vs_simulate_match(..., method = "monte carlo") respectively. vs_simulate_match_beach is a convenience function for vs_simulate_match(..., max_sets = 3, go_to = 21, go_to5 = 21) (typical beach volleyball settings).

See Also

vs_estimate_rates() vs_simulate_set()

Examples

```
## Not run:
  library(datavolley)
  x <- dv_read(dv_example_file())
  rates <- vs_estimate_rates(x, target_team = "each")

  vs_simulate_set(rates) ## simulate a single set
  vs_simulate_match(rates) ## simulate a match
  ## so given the performances of the two teams during that match, we expect
  ##  that the home team should have won, with 3-0 being the most likely scoreline

  ## compare to the actual match result
  summary(x)

## End(Not run)
```

---

vs_simulate_set *Simulate a set of volleyball*

Description

vs_simulate_set_theor and vs_simulate_set_mc are convenience functions for vs_simulate_set(..., method = "theoretical") and vs_simulate_set(..., method = "monte carlo") respectively.

Usage

```
vs_simulate_set(
  rates,
  process_model = "phase",
  serving = NA,
  go_to = 25,
  point_margin = 2,
  simple = FALSE,
  id = NULL,
  method = "theoretical"
)

vs_simulate_set_mc(...)

vs_simulate_set_theor(...)
```

Arguments

rates	list: A two-element list, each element of which is a set of rates as returned by vs_estimate_rates. Experimental: for process_model "sideout", the sideout rate component can be a function. This function will be called at each step of the simulation with the parameters: • team_1_score - the score of team 1 at each point in the set so far • team_2_score - the score of team 2 at each point in the set so far • team_1_rotation - the rotation of team 1 at each point in the set so far (rotations are counted relative to the team's starting rotation, which is 1) • team_2_rotation - the rotation of team 2 at each point in the set so far (rotations are counted relative to the team's starting rotation, which is 1) • serving - the serving team 1 or 2 at each point in the set so far • point_won_by - which team won each point in the set so far (this will be NA for the last entry, because that's the current point that hasn't been simulated yet) • outcome - the outcome of each point in the set so far, either "Sideout" or "Breakpoint" if process_model is "sideout", or details TBD if process_model is "phase" The function should return the sideout rate of the receiving team.
process_model	string: either "sideout" or "phase". The "sideout" model uses the estimated sideout rates (in the rates object) directly. The "phase" model breaks play down into different phases (serve, serve receive, etc) and uses the rates associated with those separate phases
serving	logical: if TRUE, team 1 will serve first. If NA, the team serving first will be chosen at random
go_to	integer: the minimum score that must be reached to end the set (typically 25 for indoor volleyball in sets 1 to 4, 15 in set 5, or 21 in beach volleyball)
point_margin	integer: the minimum score difference in order to win the set. Only applicable to method "monte carlo". For method "theoretical" a two-point margin is always used
simple	logical: if TRUE, return simplified output. Only applicable to method "monte carlo". If simple = TRUE, return the team (1 or 2) that won the set. If simple = FALSE, return extra details in a data.frame
id	: an optional value that (if non-NULL) will be returned in the id column of the returned data frame, if simple is FALSE
method	string: the simulation method to use. Either "monte carlo" or "theoretical". Details TBD
...	: parameters as for vs_simulate_set. vs_simulate_set_theor and vs_simulate_set_mc are convenience functions for vs_simulate_set(..., method = "theoretical") and vs_simulate_set(..., method = "monte carlo") respectively

Value

Integer (1 or 2) or a data frame, depending on the value of simple

See Also

vs_estimate_rates() vs_simulate_match()

Examples

```
## Not run:
  library(datavolley)
  x <- dv_read(dv_example_file())
  rates <- vs_estimate_rates(x, target_team = "each")

  vs_simulate_set(rates) ## simulate a single set
  vs_simulate_match(rates) ## simulate a match
  ## so given the performances of the two teams during that match, we expect
  ##  that the home team should have won, with 3-0 being the most likely scoreline

  ## compare to the actual match result
  summary(x)

## End(Not run)

## sideout rates as a function for team 2
sofun2 <- function(serving, point_won_by, ...) {
    ## if team 2 won their previous sideout opportunity, their sideout rate is 0.6
    ## otherwise it's 0.5
    prevso <- tail(na.omit(point_won_by[serving == 1]), 1)
    if (length(prevso) < 1 || prevso == 1) {
        ## first sideout opportunity or lost the last one
        0.5
    } else {
        0.6
    }
}

rates <- list(list(sideout = 0.55), ## first team has constant 55% sideout rate
              list(sideout = sofun2)) ## function for team 2's sideout rate

## need to use method = "monte carlo" for this
vs_simulate_set(rates = rates, process_model = "sideout", method = "monte carlo")
```

---

vs_theoretical_sideout_rates

Estimate theoretical sideout rates given 'phase' parameters

Description

The vs_estimate_rates() function returns a team's performance rates across a range of aspects of play, including serve ace rate, serve error rate, and so on. Using vs_theoretical_sideout_rates() We can estimate the theoretical sideout rate that we would expect to see, given those parameters. This can be compared to the actual sideout rate achieved by the team.

Usage

```
vs_theoretical_sideout_rates(rates, process_model = "phase")
```

Arguments

rates	list: rates as returned by vs_estimate_rates()
process_model	string: currently only "phase". See vs_estimate_rates()

Value

The theoretical sideout rates of the two teams

See Also

vs_estimate_rates()

Examples

```
## Not run:
  library(datavolley)
  x <- dv_read(dv_example_file())
  rates <- list(vs_estimate_rates(x, target_team = home_team(x)),
                vs_estimate_rates(x, target_team = visiting_team(x)))

  ## the theoretical sideout rates
  vs_theoretical_sideout_rates(rates)

  ## compare to their actual sideout rates
  c(rates[[1]]$sideout, rates[[2]]$sideout)

## End(Not run)
```

Index