Package: vscoututils (via r-universe)

August 13, 2024

Title Utility Functions for Volleyball Scout Files

Version 0.1.7

Description Common utility functions shared across various volleyball scouting and data access packages. Functions in this package are likely to be mostly of interest to package developers, rather than end users.

Depends R (>= 4.0.0)

License MIT + file LICENSE

Encoding UTF-8

Imports assertthat, base64enc, digest, dplyr, lubridate, methods, stringr

Suggests ovdata, testthat

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Remotes openvolley/ovdata

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

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

RemoteRef HEAD

RemoteSha 2a1c26d04083ee378a2a0995d1f0c1198ab52537

Contents

`dv_create_file_meta` *Create the* `file_meta` *component of a datavolley object*

Description

Create the `file_meta` component of a datavolley object

Usage

```
dv_create_file_meta(
  file_type = "indoor",
  date_format = "%Y/%m/%d",
  generator_day,
  generator_idp,
  generator_prg,
  generator_release,
  generator_version,
  generator_name
)
```

Arguments

file_type	string: "indoor", "beach"
date_format	string: preferred date format
generator_day	datetime: date and time of file creation
generator_idp, generator_prg, generator_release, generator_version, generator_name	string: information about the software that generated the file. If missing, defaults will be used

Value

A tibble

---

dv_create_meta *Create metadata component of a datavolley object*

Description

Create metadata component of a datavolley object

Usage

```
dv_create_meta(
  match,
  more,
  comments,
  result,
  teams,
  players_h,
  players_v,
  video,
  attacks,
  setter_calls,
```

```
  winning_symbols,
  data_type = "indoor",
  style = "default"
)
```

Arguments

match	tibble: as returned by dv_create_meta_match()
more	tibble: as returned by dv_create_meta_more()
comments	tibble: as returned by dv_create_meta_comments()
result	tibble: as returned by dv_create_meta_result()
teams	tibble: as returned by dv_create_meta_teams()
players_h	tibble: home team table as returned by dv_create_meta_players()
players_v	tibble: visiting team table as returned by dv_create_meta_players()
video	tibble: as returned by dv_create_meta_video()
attacks	tibble: as returned by dv_create_meta_attack_combos(). If missing, the default attack combos for data_type and style and simplified = TRUE will be used
setter_calls	tibble: as returned by dv_create_meta_setter_calls(). If missing, the default setter calls for data_type and style will be used
winning_symbols	tibble: as returned by dv_default_winning_symbols(). If missing, the default winning symbols for data_type and style will be used
data_type	string: "indoor", "beach"
style	string: conventions "default", "volleymetrics", "german"

Value

A list

Examples

```
dv_create_meta(match = dv_create_meta_match(zones_or_cones = "Z"), more = dv_create_meta_more(),
               comments = dv_create_meta_comments(),
               teams = dv_create_meta_teams(
                   team_ids = c("ardv", "badg"), teams = c("Aardvarks", "Badgers")))
```

dv_create_meta_attack_combos *Create the* attacks *metadata component of a datavolley object*

Description

The attacks metadata component of a datavolley object describes the attack combination codes

Usage

```
dv_create_meta_attack_combos(
  code,
  start_zone,
  side = NA_character_,
  tempo,
  description,
  colour = NA_character_,
  start_coordinate = NA_integer_,
  target_attacker
)
```

Arguments

code	character: vector of two-character (uppercase) attack codes. Attack codes must start with C, G, I, J, L, P, V, W, X, Y, or Z
start_zone	integer: vector of start zones
side	character: vector of L, C, R
tempo	character: vector of tempo codes F, H, M, N, O, Q, T, U
description	character: vector of attack descriptions
colour	character: vector of colour codes (maybe) used when plotting, in "#RRGGBB" format
start_coordinate	integer: vector of start locations as single-index coordinates (see e.g. datavolley:dv_xy2index)
target_attacker	character: vector of single-character codes giving the target attacker: B (back/right side attacker), C (centre), F (front/left side attacker), P (pipe), S (setter attack)

Value

A tibble

dv_create_meta_comments *Create the* comments *metadata component of a datavolley object*

Description

Create the comments metadata component of a datavolley object

Usage

```
dv_create_meta_comments(
  summary,
  match_description,
  home_coach_comments,
  visiting_coach_comments
)
```

Arguments

summary, match_description, home_coach_comments, visiting_coach_comments
: string: comments

Value

A tibble

dv_create_meta_match *Create the* match *metadata component of a datavolley object*

Description

Create the match metadata component of a datavolley object

Usage

```
dv_create_meta_match(
  date,
  season,
  league,
  phase,
  home_away,
  day_number,
  match_number,
  regulation = "indoor rally point",
  zones_or_cones
)
```

Arguments

`date`	Date or datetime: match date and time
`season`	string: season name
`league`	string: league name
`phase`	string: match phase (e.g. "Playoffs", "Finals")
`home_away`	string: typically "Home" or "Away"
`day_number, match_number`	numeric: day and match number
`regulation`	string: "indoor sideout", "indoor rally point" (default), or "beach rally point"
`zones_or_cones`	string: are attacks being scouted with "zones" or "cones" (or "Z" or "C")

Value

A tibble

`dv_create_meta_match_id` *Create the* `match_id` *metadata component of a datavolley object*

Description

The `match_id` is generated from the `match` and `teams` components of the `mx` object, so ensure that these are fully populated before generating a match_id

Usage

```
dv_create_meta_match_id(mx)
```

Arguments

`mx`	list: the `meta` component of a datavolley object

Value

A string

dv_create_meta_more *Create the* more *metadata component of a datavolley object*

Description

Create the more metadata component of a datavolley object

Usage

```
dv_create_meta_more(referees, spectators, receipts, city, arena, scout)
```

Arguments

referees, spectators, receipts, city, arena	string: match descriptors
scout	string: name of the scout

Value

A tibble

dv_create_meta_players *Create the* players_h *or* players_v *metadata component of a datavolley object*

Description

Create the players_h or players_v metadata component of a datavolley object

Usage

```
dv_create_meta_players(players)
```

Arguments

players	data.frame: the team roster, as a data.frame. players must contain the columns lastname, firstname, and number. Columns player_id, role, nickname, special_role, foreign, and starting_position_set1 - starting_position_set5 are optional

Value

A tibble

dv_create_meta_result *Create the* result *metadata component of a datavolley object*

Description

Note that the returned object has some NA values. It is expected that the user will call dv_update_meta() on the full datavolley object, which will update entries in this metadata component.

Usage

```
dv_create_meta_result(
  home_team_scores = NA_integer_,
  visiting_team_scores = NA_integer_,
  durations = NA_integer_
)
```

Arguments

home_team_scores	integer: vector of home team scores in each set
visiting_team_scores	integer: vector of visiting team scores in each set
durations	integer: vector of set durations (in minutes). If not provided, these will be calculated from video times by dv_update_meta()

Value

A tibble

dv_create_meta_setter_calls *Create the* sets *(setter calls) metadata component of a datavolley object*

Description

Create the sets (setter calls) metadata component of a datavolley object

Usage

```
dv_create_meta_setter_calls(
  code,
  description,
  colour = NA_character_,
  start_coordinate = NA_integer_,
  mid_coordinate = NA_integer_,
  end_coordinate = NA_integer_,
  path = NA_character_,
  path_colour = NA_character_
)
```

Arguments

code	character: vector of two-character (uppercase) setter call codes. Setter call codes must start with K
description	character: vector of setter call descriptions
colour	character: vector of colour codes (maybe) used when plotting, in "#RRGGBB" format
start_coordinate, mid_coordinate, end_coordinate	integer: vector of start/mid/end locations as single-index coordinates (see e.g. datavolley:dv_xy2index)
path	character: vector of paths, each a comma-separated list of single-index coordinates
path_colour	character: vector of colour codes in "#RRGGBB" format

Value

A tibble

---

dv_create_meta_teams *Create the* teams *metadata component of a datavolley object*

Description

dv_create_meta_teams2 provides an alternative parameterization with home and visiting team information specified separately.

Usage

```
dv_create_meta_teams(
  team_ids,
  teams,
  sets_won,
  coaches,
  assistants,
```

```
  shirt_colours
)

dv_create_meta_teams2(
  home_team_id,
  home_team,
  home_coach,
  home_assistant,
  home_shirt_colour,
  visiting_team_id,
  visiting_team,
  visiting_coach,
  visiting_assistant,
  visiting_shirt_colour,
  sets_won
)
```

Arguments

`team_ids`	character: (required) 2-element vector of home and visiting team IDs
`teams`	character: (required) 2-element vector of home and visiting team names
`sets_won`	integer: 2-element vector of the number of sets won by the home team and visiting team
`coaches`	character: 2-element vector of home and visiting team coach names
`assistants`	character: 2-element vector of home and visiting team assistant coach names
`shirt_colours`	character: 2-element vector of home and visiting team shirt colours in "#RRGGBB" format
`home_team_id`, `visiting_team_id`	string: home and visiting team IDs
`home_team`, `visiting_team`	string: home and visiting team names
`home_coach`, `visiting_coach`	string: home and visiting coach name
`home_assistant`, `visiting_assistant`	string: home and visiting assistant coach name
`home_shirt_colour`, `visiting_shirt_colour`	string: home and visiting shirt colours in "#RRGGBB" format

Value

A tibble

dv_create_meta_video *Create the* video *metadata component of a datavolley object*

Description

Create the video metadata component of a datavolley object

Usage

```
dv_create_meta_video(video_file)
```

Arguments

video_file	character: path to video file. More than one is allowed, but anything other than the first might be ignored by some functions

Value

A data.frame

dv_decode_evaluation *Decode evaluation codes*

Description

Decode evaluation codes

Usage

```
dv_decode_evaluation(
  skill,
  evaluation_code,
  table,
  data_type = "indoor",
  style = "default"
)
```

Arguments

skill	character: full skill names (Serve, Reception, etc)
evaluation_code	character: evaluation codes (#, +, etc)
table	data.frame: optional table with columns skill, evaluation_code, evaluation. If not provided, the default table for data_type and style will be used
data_type	string: "indoor" or "beach"
style	string: "default", "volleymetrics"

Value

A character vector of evaluations, with a "dvmessages" attribute if any issues were found

Examples

```
dv_decode_evaluation("Serve", "#")
dv_decode_evaluation("Attack", "!")
```

`dv_decode_num_players` *Decode number of players*

Description

Decode number of players

Usage

```
dv_decode_num_players(
  skill,
  num_players_code,
  table,
  data_type = "indoor",
  style = "default"
)
```

Arguments

`skill`	character: full skill names (Serve, Reception, etc)
`num_players_code`	character: number of players codes (0, 1, etc)
`table`	data.frame: optional table with columns `skill`, `num_players_code`, `num_players`. If not provided, the default table for `data_type` and `style` will be used
`data_type`	string: "indoor" or "beach"
`style`	string: "default", "volleymetrics"

Value

A character vector of number of players, with a "dvmessages" attribute if any issues were found

Examples

```
dv_decode_num_players("Reception", 7)
dv_decode_num_players("Attack", 3, data_type = "beach")
```

`dv_decode_skill_subtype` *Decode skill subtypes*

Description

Decode skill subtypes

Usage

```
dv_decode_skill_subtype(
  skill,
  skill_subtype_code,
  evaluation,
  table,
  data_type = "indoor",
  style = "default"
)
```

Arguments

`skill`	character: full skill names (Serve, Reception, etc)
`skill_subtype_code`	character: subtype codes (H, P, etc)
`evaluation`	character: skill evaluations (Ace, Perfect pass, etc). Only used to make small adjustments for certain scouting styles, which will be skipped if `evaluation` is not provided
`table`	data.frame: optional table with columns `skill`, `skill_subtype_code`, `skill_subtype`. If not provided, the default table for `data_type` and `style` will be used
`data_type`	string: "indoor" or "beach"
`style`	string: "default", "volleymetrics"

Value

A character vector of skill subtypes, with a "dvmessages" attribute if any issues were found

Examples

```
dv_decode_skill_subtype("Reception", "M")
dv_decode_skill_subtype("Attack", "P")
dv_decode_skill_subtype("Dig", "Q")
```

dv_decode_skill_type *Decode skill type (tempo) codes*

Description

Decode skill type (tempo) codes

Usage

```
dv_decode_skill_type(
  skill,
  skill_type_code,
  table,
  data_type = "indoor",
  style = "default"
)
```

Arguments

skill	character: full skill names (Serve, Reception, etc)
skill_type_code	character: skill type (tempo) codes (Q, M, H, etc)
table	data.frame: optional table with columns skill, skill_type_code, skill_type. If not provided, the default table for data_type and style will be used
data_type	string: "indoor" or "beach"
style	string: "default", "volleymetrics"

Value

A character vector of skill types, with a "dvmessages" attribute if any issues were found

Examples

```
dv_decode_skill_type("Serve", "Q")
dv_decode_skill_type("Attack", "Z")
```

dv_decode_special_code *Decode special codes*

Description

Decode special codes

Usage

```
dv_decode_special_code(
  skill,
  special_code,
  evaluation,
  table,
  data_type = "indoor",
  style = "default"
)
```

Arguments

skill	character: full skill names (Serve, Reception, etc)
special_code	character: special codes (Z, U, etc)
evaluation	character: skill evaluations (Ace, Perfect pass, etc)
table	data.frame: optional table with columns skill, special_code, special. If not provided, the default table for data_type and style will be used
data_type	string: "indoor" or "beach"
style	string: "default", "volleymetrics"

Value

A character vector of special code interpretations, with a "dvmessages" attribute if any issues were found

Examples

```
dv_decode_special_code("Reception", "P", NA_character_) ## evaluation is irrelevant here
dv_decode_special_code("Attack", "N", "Error")
dv_decode_special_code("Attack", "N", "Winning attack") ## different interpretation if not an error
dv_decode_special_code("Attack", "C", "Winning attack") ## "C" is not valid for winning attack
dv_decode_special_code("Attack", "C", "Positive, good attack") ## but is for attacks in play
dv_decode_special_code("Attack", "Q", "Positive, good attack") ## this is invalid everywhere
```

`dv_default_attack_combos` *Default attack combination codes table*

Description

Default attack combination codes table

Usage

```
dv_default_attack_combos(
  data_type = "indoor",
  style = "default",
  simplified = TRUE
)
```

Arguments

`data_type`	string: "indoor", "beach"
`style`	string: conventions "default", "volleymetrics", "german"
`simplified`	logical: if `TRUE`, just the most common ones (indoor only)

Value

A tibble

`dv_default_num_players` *Default number of players table*

Description

Default number of players table

Usage

```
dv_default_num_players(data_type = "indoor", style = "default")
```

Arguments

`data_type`	string: "indoor", "beach"
`style`	string: conventions "default", "volleymetrics", "german"

Value

A tibble

`dv_default_scouting_table` *Default scouting (type and evaluation for each skill) table*

Description

Default scouting (type and evaluation for each skill) table

Usage

```
dv_default_scouting_table(data_type = "indoor", style = "default")
```

Arguments

`data_type`	string: "indoor", "beach"
`style`	string: conventions "default", "volleymetrics", "german"

Value

A tibble

`dv_default_setter_calls` *Default setter calls table*

Description

Default setter calls table

Usage

```
dv_default_setter_calls(data_type = "indoor", style = "default")
```

Arguments

`data_type`	string: "indoor", "beach"
`style`	string: conventions "default", "volleymetrics", "german"

Value

A tibble

dv_default_skill_evaluations

Default skill evaluation table

Description

Default skill evaluation table

Usage

```
dv_default_skill_evaluations(data_type = "indoor", style = "default")
```

Arguments

data_type	string: "indoor", "beach"
style	string: conventions "default", "volleymetrics", "german"

Value

A tibble

dv_default_skill_subtypes

Default skill_subtype (type of hit) table

Description

Default skill_subtype (type of hit) table

Usage

```
dv_default_skill_subtypes(data_type = "indoor", style = "default")
```

Arguments

data_type	string: "indoor", "beach"
style	string: conventions "default", "volleymetrics", "german"

Value

A tibble

`dv_default_skill_types`
Default skill_type (tempo) table

Description

Default skill_type (tempo) table

Usage

```
dv_default_skill_types(data_type = "indoor", style = "default")
```

Arguments

`data_type`	string: "indoor", "beach"
`style`	string: conventions "default", "volleymetrics", "german"

Value

A tibble

`dv_default_special_codes`
Default special codes table

Description

Default special codes table

Usage

```
dv_default_special_codes(data_type = "indoor", style = "default")
```

Arguments

`data_type`	string: "indoor", "beach"
`style`	string: conventions "default", "volleymetrics", "german"

Value

A tibble

dv_default_winning_symbols *Default winning symbols table*

Description

Default winning symbols table

Usage

```
dv_default_winning_symbols(data_type = "indoor", style = "default")
```

Arguments

data_type	string: "indoor", "beach"
style	string: conventions "default", "volleymetrics", "german"

Value

A tibble

dv_expand_rally_codes *Expand scouted rally codes*

Description

- insert dummy rally rows for score corrections. Note that this will not adjust the setter position in the dummy rows nor insert intermediate setter positions (but will insert one setter position code per team at the start of the returned data frame, if needed)
- for a substitution, insert the setter assignment code (*PXX or aPXX) if it was the on-court setter that was substituted
- insert the setter position codes at the start of the rally. These should not be inserted on the first point of a set, in that situation they should be in the >LUp codes - so for the first point, make sure that the `last_home_setter_position` and `last_visiting_setter_position` are passed as their starting values)

Usage

```
dv_expand_rally_codes(
  rx,
  last_home_setter_position,
  last_home_setter,
  last_visiting_setter_position,
  last_visiting_setter,
  last_home_team_score,
```

```
  last_visiting_team_score,
  keepcols,
  meta,
  rebuild_codes = TRUE,
  do_warn = TRUE
)
```

Arguments

`rx`	data.frame: with at least the columns: • point_id, code, team ("*" or "a"), point, substitution, timeout, home_setter_position, visiting_setter_position • home_team_score, visiting_team_score (scores at the end of the rally) • player_out, player_in (substitions) and optional columns: • player_number, skill (one-character code), skill_type_code, evaluation_code (used to rebuild the codes before calling `dv_green_codes()`
`last_home_setter_position`, `last_visiting_setter_position`	integer: home and visiting setter positions in the previous rally
`last_home_setter`, `last_visiting_setter`	integer: home and visiting setter jersey numbers in the previous rally
`last_home_team_score`, `last_visiting_team_score`	integer: home and visiting team scores at the end of the previous rally
`keepcols`	character: names of the columns in `rx` to keep, when inserting new rows. Values in these columns will be copied from an adjacent row. If `keepcols` is not provided, a guess will be made
`meta`	list: meta component from a datavolley object
`rebuild_codes`	logical: if `TRUE`, reconstruct the scout code before calling `dv_green_codes()`
`do_warn`	logical: if `FALSE`, suppress warnings issued directly by this function (but warnings generated by downstream code will still occur)

Value

`rx` potentially with additional rows inserted. Note that the added rows might have fractional `point_id` values, in which case you will need to renumber all `point_ids` in the match to integers afterwards

---

dv_file_type *Detect scout file type*

Description

Returns the general file type (format), but does not differentiate e.g. indoor from beach. Type is based on the file extension and a minimal check of the contents, but without fully parsing the file. For `dv_file_data_type` the type is based only on a minimal check of the contents (provided in `x`)

Usage

```
dv_file_type(filename, error_on_unknown = FALSE)

dv_file_data_type(x, error_on_unknown = FALSE)
```

Arguments

filename	character: paths to files. Files that do not exist will be classified as "unknown" type
error_on_unknown	logical: if TRUE an error will be thrown if an unknown file type is encountered
x	character: a character vector with the contents of a scout file

Value

Character vector containing:

- "dvw"
- "vsm"
- "psvb"
- "hxml"
- "unknown"

Examples

```
## Not run:
  library(ovdata)
  dv_file_type(ovdata_example("190301_kats_beds")) ## dvw
  x <- readLines(ovdata_example("190301_kats_beds"), warn = FALSE)
  dv_file_data_type(x)

  dv_file_type(ovdata_example("clickscout")) ## dvw
  dv_file_data_type(readLines(ovdata_example("clickscout"), warn = FALSE))

  dv_file_type(ovdata_example("2017_AVL_mens_HEAT_vs_UTSSU")) ## psvb
 dv_file_data_type(readLines(ovdata_example("2017_AVL_mens_HEAT_vs_UTSSU"), warn = FALSE))

  dv_file_type(tempfile(fileext = ".dvw")) ## unknown (contents not as expected for dvw)
  dv_file_data_type(readLines(tempfile(fileext = ".dvw"))) ## "unknown", with warning


## End(Not run)
```

dv_get_player_meta *Extract player metadata by team and jersey number*

Description

Extract player metadata by team and jersey number

Usage

```
dv_get_player_meta(team, number, meta)
```

Arguments

team	character: vector of "*" or "a"
number	integer: vector of player jersey numbers
meta	list: the meta component of a datavolley object, or a datavolley object

Value

A data frame with columns "team" and plus those in meta, sorted to match the inputs team and number

dv_green_codes *Add green codes to the scouted codes from a rally*

Description

Add green codes to the scouted codes from a rally

Usage

```
dv_green_codes(code, meta, do_warn = TRUE)
```

Arguments

code	character: a character vector of scouted codes for a rally. The codes must be non-compound but need only be the first 6 characters (i.e. up to and including the evaluation_code). The code vector must have a valid point code at the end (the *p or ap code)
meta	list: meta component from a datavolley object
do_warn	logical: if FALSE, suppress warnings issued directly by this function (but warnings generated by downstream code will still occur)

Value

A character vector of codes. This will be the same as the input code vector, but potentially with additional green code entries (*$$&H#, *$$&H=, a$$&H#, a$$&H=) inserted before the last entry

`dv_insert_sets` *Insert setting actions for attacks that have not been scouted with sets*

Description

This function will insert setting actions prior to attacks, where those attacks do not already have a set scouted. The sets are assigned to the setter on court for that team. It is therefore possible to scout a match, only manually scouting the setting actions where they were made by a player other than the designated setter. The remaining setting actions (made by the designated setter) can be filled in using this function, making the live scouting a little more efficient. Note, however, that automatically-inserted sets do not have the full information that can be included when scouting manually, including setter calls (but see the note on the `set_call_table` parameter) and location of the set.

Usage

```
dv_insert_sets(
  x,
  no_set_attacks = c("PP", "P2", "PR"),
  phase_select = "Reception",
  default_set_evaluation = "+",
  attack_rows,
  set_call_table
)
```

Arguments

`x`	datavolley: datavolley object as returned by `datavolley::dv_read()`
`no_set_attacks`	character: vector of attack codes for which we will not automatically insert sets (e.g. setter tips, overpass attacks)
`phase_select`	character: play phase(s) of attacks to consider. One or more of "Reception", "Transition"
`default_set_evaluation`	string: the default evaluation code for a set (used unless the attack was against 0 or 1 blockers, in which case it gets "#")
`attack_rows`	integer: a vector of row numbers of attacks for which sets should be inserted. Automatically calculated if not provided
`set_call_table`	data.frame: a data.frame with columns `attack_code` and `set_call`. Setter calls will be added to sets associated with attack codes in this list. Note that setter calls from this table will NOT be inserted on sets where the setter did not set the middle hitter (e.g. if the middle ran X1 but the setter set someone else, no "K1" call can be inserted because there is no way of knowing what the middle was running). This gives a biased set of setter call entries that are unlikely to be useful for analysis purposes. It is therefore recommended that you provide `set_call_table` to this function ONLY if you are then going to manually insert setter calls on the remaining rows

Value

A modified copy of x

Examples

```
## Not run:
  x <- dv_read(dv_example_file())
  sum(plays(x)$skill == "Set", na.rm = TRUE)
  x <- dv_insert_sets(x)
  sum(plays(x)$skill == "Set", na.rm = TRUE)

## End(Not run)
```

---

dv_insert_sets_check *Find attacks for which we could insert missing setting actions*

Description

Find attacks for which we could insert missing setting actions

Usage

```
dv_insert_sets_check(
  x,
  no_set_attacks = c("PP", "P2", "PR"),
  phase_select = "Reception"
)
```

Arguments

x	datavolley: datavolley object as returned by datavolley::dv_read()
no_set_attacks	character: vector of attack codes for which we will not automatically insert sets (e.g. setter tips, overpass attacks)
phase_select	character: play phase(s) of attacks to consider. One or more of "Reception", "Transition"

Value

The row numbers of attacks in the plays component of x, for which sets could be inserted

dv_insert_technical_timeouts *Insert technical timeouts*

Description

Insert technical timeouts

Usage

```
dv_insert_technical_timeouts(x, at, data_type)
```

Arguments

x	datavolleyplays: the plays component of a datavolley object as returned by dv_read()
at	list: (optional) a two-element list can be supplied, giving the scores at which technical timeouts will be inserted for sets 1–4, and set 5 or golden sets. If not provided, technical timeouts are inserted at points 8 and 16 of sets 1–4 (for indoor files) or when the team scores sum to 21 in sets 1–2 (beach)
data_type	string: "indoor" or "beach". If not provided, a guess will be made as to whether x is beach or indoor data

Value

A modified copy of x

dv_update_meta *Update the metadata component of a datavolley object*

Description

These elements will be updated:

- the played, duration, and score-related columns in x$meta$result
- the sets_won and won_match columns in x$meta$teams
- the starting positions and substitutions in x$meta$players_h and x$meta$players_v

Usage

```
dv_update_meta(x)
```

Arguments

x	datavolley: as returned by e.g. datavolley::dv_read()

Value

A modified copy of x

---

pv_default_errortypes *Define errortype interpretations*

Description

Define the interpretation of errortypes associated with events.

Usage

```
pv_default_errortypes(data_type)
```

Arguments

data_type	string: "indoor" or "beach"

Value

a tibble with columns "skill", "errortype", and "evaluation"

Examples

```
pv_default_errortypes("beach")
```

---

pv_default_eventgrades *Define eventgrade interpretations*

Description

Define the interpretation of eventgrades associated with events.

Usage

```
pv_default_eventgrades(data_type)
```

Arguments

data_type	string: "indoor" or "beach"

Value

a tibble with columns "skill", "eventgrade", "evaluation_code" (the equivalent DataVolley code, if there is one), "evaluation", and "win_loss"

Examples

```
pv_default_eventgrades("beach")
```

---

pv_default_eventtypes *Lookup tables for Perana Sports database codes*

Description

Lookup tables for Perana Sports database codes

Usage

```
pv_default_eventtypes(data_type)
```

Arguments

data_type	string: "indoor" or "beach"

Value

A tibble

---

pv_default_subevents *Define subevent interpretations*

Description

Define the interpretation of subevents associated with events

Usage

```
pv_default_subevents(data_type)
```

Arguments

data_type	string: "indoor" or "beach"

Value

a tibble with columns "skill", "subevent", and "interpretation"

Examples

```
pv_default_subevents("beach")
```

pv_default_subevents2 *Define subevent2 interpretations*

Description

Define the interpretation of subevents2 associated with events

Usage

```
pv_default_subevents2(data_type)
```

Arguments

data_type	string: "indoor" or "beach"

Value

a tibble with columns "skill", "subevent2", and "interpretation"

Examples

```
pv_default_subevents("beach")
```

pv_parse_ballstring *Parse a ballstring found in Perana Sports files*

Description

Parse a ballstring found in Perana Sports files

Usage

```
pv_parse_ballstring(z, which = "start")
```

Arguments

z	character: vector of ballstring entries
which	string: "start", "mid", or "end". Will be used to name the columns of the returned data.frame

Value

A data.frame with columns `start_coordinate_x` and `start_coordinate_y` (or `mid_*` or `end_*`, depending on `which`)

Examples

```
pv_parse_ballstring(c("51.25, 143.33"))
```

---

vscoututils **vscoututils**

Description

Utility Functions for Volleyball Scout Files

Author(s)

Maintainer: Ben Raymond <ben@untan.gl>

Authors:

- Adrien Ickowicz

Index