Package 'pitchRx'

December 9, 2015

Title Tools for Harnessing 'MLBAM' 'Gameday' Data and Visualizing 'pitchfx'

Version 1.8.2

Author Carson Sievert <cpsievert1@gmail.com>

Maintainer Carson Sievert <cpsievert1@gmail.com>

Description With 'pitchRx', one can easily obtain Major League Baseball Advanced Media's 'Gameday' data (as well as store it in a remote database). The 'Gameday' website hosts a wealth of data in XML format, but perhaps most interesting is 'pitchfx'. Among other things, 'pitchfx' data can be used to recreate a baseball's flight path from a pitcher's hand to home plate. With pitchRx, one can easily create animations and interactive 3D 'scatterplots' of the baseball's flight path. 'pitchfx' data is also commonly used to generate a static plot of baseball locations at the moment they cross home plate. These plots, sometimes called strike-zone plots, can also refer to a plot of event probabilities over the same region. 'pitchRx' provides an easy and robust way to generate strike-zone plots using the 'ggplot2' package.

License MIT + file LICENSE

Depends R (>= 2.15.1), ggplot2 (>= 0.9.3)

Imports XML2R (>= 0.0.6), plyr, MASS, hexbin, mgcv

Suggests DBI, dplyr, RSQLite (>= 1.0.0), parallel, knitr, animation, shiny, testthat, ggsubplot, rgl

LazyData true

URL http://cpsievert.github.com/pitchRx

BugReports http://github.com/cpsievert/pitchRx/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2015-12-09 12:57:25

R topics documented:

`animateFX` *Animate PITCHf/x*

Description

Pitch trajectories animated on a two-dimensional plot.

Usage

```
animateFX(data, color = "pitch_types", avg.by, point.alpha = 1/3,
  limitz = c(-3.5, 3.5, 0, 7), flag = FALSE, interval = 0.01,
  layer = list(), parent = FALSE, ...)
```

Arguments

`data`	data frame with appropriately named PITCHf/x variables
`color`	variable used to control coloring scheme.
`avg.by`	variable used as an index for averaging over PITCHf/x parameters
`point.alpha`	ggplot2 alpha parameter
`limitz`	limits for horizontal and vertical axes.
`flag`	indicate whether or not batter has decided to swing.
`interval`	time (in seconds) between plotting the pitch locations.
`layer`	list of ggplot2 layer modifications.
`parent`	is the function being called from a higher-level function? (experimental)
`...`	extra options passed onto geom commands

Details

animateFX plots a series of "snapshots" that represent pitch trajectories from the point of release until all of them reach home plate. The graphic takes on the viewpoint of the umpire; that is, the pitches are getting closer to the viewer with time. This is relected with the increase in size of the "balls" as the animation progresses.

Value

Returns a series of objects of the class used by package ggplot2 to represent plots.

Examples

```
data(pitches)
#generate animation and prompt default web browser to view the sequence of plots
## Not run:
animation::saveHTML({ animateFX(pitches, layer = facet_grid(pitcher_name~stand)) })
animation::saveHTML({ animateFX(pitches, avg.by="pitch_types",
                          layer = facet_grid(pitcher_name~stand))
                  })

## End(Not run)
```

---

export *Export (append) a data.frame to a remote table in a database.*

Description

This function is convenient if you plan on repeatedly appending to a table in a database. All that is required is a database connection and a data.frame you want to export to that database. If you want to initiate a table with more columns use the template argument. Note that if the table already exists, the template argument will be ignored.

Usage

```
export(connect, value, name, template, ...)
```

Arguments

connect	database connection.
value	local data frame.
name	name of the remote table.
template	a named character vector. The names of the vector should contain the names of value. The values of this vector should contain the relevant field types.
...	arguments passed onto DBI::dbWriteTable

Examples

```
## Not run:
library(dplyr)
my_db <- src_sqlite("DB.sqlite3")
data(pitches, package="pitchRx")
# Creates the 'pitches' table in the database
export(connect=my_db$con, value=pitches, name="pitches")
# Appends to the 'pitches' tables, but with the first column missing
export(connect=my_db$con, value=pitches[,-1], name="pitches")
tail(data.frame(collect(tbl(my_db, "pitches")))) #verify it appends correctly
# This data frame has a column that doesn't exist in the pitches table --
# so a new table is created.
export(connect=my_db$con, value=cbind(pitches, test="works"), name="pitches")

## End(Not run)
```

---

fields *Master list of tables and fields returned by* scrape

Description

This data object is as a template for fields and fields types for each table. Since it's much easier to write to a table with more fields (than vice versa), this object contains every possible field for each table.

Usage

```
fields
```

Format

A list of character vectors.

---

getSnapshots *Produce time sequenced pitch locations from PITCHf/x parameters*

Description

This function generates the x, y and z locations used in animateFX and interactiveFX.

Usage

```
getSnapshots(data, interval = 0.01)
```

Arguments

`data`	The nine PITCHf/x parameters used to determine the location of a pitch at a given time.
`interval`	the amount of time between 'snapshots'

Value

Return a three dimensional array. The third dimension corresponds to different 'snapshots' of locations.

References

http://baseball.physics.illinois.edu/KaganPitchfx.pdf

`gids` *All MLB Gameday IDs from 2008-2013*

Description

A character vector with every "gameday" attribute in the "game" element taken from scoreboard files like this one: http://gd2.mlb.com/components/game/mlb/year_2011/month_04/day_04/gid_2011_04_04_minmlb_nyamlb_1/miniscoreboard.xml Note they are ordered from oldest game to newest game.

Usage

```
gids
```

Format

A character vector

`interactiveFX` *Use rgl to visualize PITCHf/x*

Description

Three-dimensional plot of pitch trajectories.

Usage

```
interactiveFX(data, spheres = TRUE, color = "pitch_types", avg.by,
  interval = 0.01, alpha = 1, show.legend = TRUE, ...)
```

Arguments

data	data.frame with appropriately named PITCHf/x variables
spheres	Use rgl::spheres3d or rgl::plot3d?
color	variable used to control coloring scheme.
avg.by	variable used as an index for averaging over PITCHf/x parameters
interval	the amount of time between 'snapshots'
alpha	color transparency
show.legend	print coloring legend in R console?
...	other param passed onto rgl::spheres3d or rgl::plot3d

Value

rgl object is returned.

Examples

```
data(pitches)
Rivera <- subset(pitches, pitcher_name=="Mariano Rivera")
## Not run:
 interactiveFX(Rivera, interval=.05)
 interactiveFX(Rivera, avg.by="pitch_types")

## End(Not run)
```

---

makeUrls *Construct Gameday urls based on some parameters.*

Description

This is a convenience function (used by scrape) which constructs urls with the common Gameday root http://gd2.mlb.com/components/game/mlb/.

Usage

```
makeUrls(start, end, gids = "infer")
```

Arguments

start	date "yyyy-mm-dd" to commence scraping.
end	date "yyyy-mm-dd" to terminate scraping.
gids	The default value "infer" suggests gameday_links should be derived and appended appropriately (based on values of start and end). Otherwise, a character vector with gameday_links can be supplied.

Value

Returns a character vector.

Examples

```
# XML file names with pitch-by-pitch level data
prefix <- makeUrls(start="2011-04-04", end="2011-04-04")
paste0(prefix, "/inning/inning_all.xml")
# XML file names with hit location data
paste0(prefix, "/inning/inning_hit.xml")
# XML file names with game-by-game level data
paste0(makeUrls(start="2011-04-04", end="2011-04-04", gids=""), "/miniscoreboard.xml")
# Use gids option instead
data(gids)
identical(prefix, makeUrls(gids=gids[grep("2011_04_04", gids)]))
```

---

nonMLBgids *All non-MLB Gameday IDs from 2008-2013*

Description

A character vector with every "gameday" attribute in the "game" element taken from scoreboard files like this one: http://gd2.mlb.com/components/game/aaa/year_2013/month_06/day_08/gid_2013_06_08_freaaa_slcaaa_1/miniscoreboard.xml Note they are ordered from oldest game to newest game.

Usage

```
nonMLBgids
```

Format

A character vector

---

pitches *Sample PITCHf/x Data Set*

Description

Every four-seam and cutting fastball thrown by Mariano Rivera and Phil Hughes during the 2011 season.

Usage

```
pitches
```

Format

A data frame with variables from the 'atbat' and 'pitch' tables.

See Also

http://fastballs.wordpress.com/2007/08/02/glossary-of-the-gameday-pitch-fields/

Examples

```
#This can reproduce data(pitches), but it takes a while...
## Not run:
data <- scrape(start="2011-01-01", end="2011-12-31")
names <- c("Mariano Rivera", "Phil Hughes")
atbats <- subset(data$atbat, pitcher_name %in% names)
pitchFX <- plyr::join(atbats, data$pitch, by=c("num", "url"), type="inner")
pitches <- subset(pitchFX, pitch_type %in% c("FF", "FC"))

## End(Not run)
```

---

pitchRx *PITCHf/x package*

Description

PITCHf/x package

Author(s)

Carson Sievert

See Also

http://cpsievert.github.com/pitchRx

---

players *All MLB and MiLB players from 2008 to date*

Description

A data frame with the full name and corresponding ID for every player. This data is used during `scrape` to append a name to the atbat table so we can reference data by batter_name and pitcher_name without any extra hassle. This was constructed by taking the unique name and ID combinations from every players.xml file.

Usage

```
players
```

Format

A data frame with 2 variables: `ID` and `full_name`

`scrape` *Scrape Major League Baseball's Gameday Data*

Description

Function for obtaining PITCHf/x and other related Gameday Data. `scrape` currently has support for files ending with: inning/inning_all.xml, inning/inning_hit.xml, players.xml, or miniscoreboard.xml. It's worth noting that PITCHf/x is contained in files ending with "inning/inning_all.xml", but the other files can complement this data depending on the goal for analysis. Any collection of file names may be passed to the `suffix` argument, and `scrape` will retrieve data from a (possibly large number) of files based on either a window of dates or a set of `game.ids`. If collecting data in bulk, it is strongly recommended that one establishes a database connection and supplies the connection to the `connect` argument. See the examples section for a simple example of how to do so.

Usage

```
scrape(start, end, game.ids, suffix = "inning/inning_all.xml", connect, ...)
```

Arguments

`start`	character string specifying a date "yyyy-mm-dd" to commence scraping.
`end`	character string specifying a date "yyyy-mm-dd" to terminate scraping.
`game.ids`	character vector of gameday_links. If this option is used, `start` and `end` are ignored. See `data(gids, package="pitchRx")` for examples.
`suffix`	character vector with suffix of the XML files to be parsed. Currently supported options are: 'players.xml', 'miniscoreboard.xml', 'inning/inning_all.xml', 'inning/inning_hit.xml'.
`connect`	A database connection object. The class of the object should be "MySQLConnection" or "SQLiteConnection". If a valid connection is supplied, tables will be copied to the database, which will result in better memory management. If a connection is supplied, but the connection fails for some reason, csv files will be written to the working directory.
`...`	arguments passed onto `XML2R::XML2Obs`. Among other things, this can be used to switch on asynchronous downloads.

Value

Returns a list of data frames (or nothing if writing to a database).

Note

This function was adapted from `scrapeFX` which is deprecated as of version 1.0

See Also

If you want to add support for more file types, the XML2R package is a good place to start.

Examples

```
## Not run:
# Collect PITCHf/x (and other data from inning_all.xml files) from
# all games played on August 1st, 2013 (using asynchronous downloads)
dat <- scrape(start = "2013-08-01", end = "2013-08-01")
#As of XML2R 0.0.5, asyncronous downloads can be performed
dat <- scrape(start = "2013-08-01", end = "2013-08-01", async = TRUE)

# Scrape PITCHf/x from Minnesota Twins 2011 season
data(gids, package = "pitchRx")
twins11 <- gids[grepl("min", gids) & grepl("2011", gids)]
dat <- scrape(game.ids = twins11[1]) #scrapes 1st game only

data(nonMLBgids, package = "pitchRx")
# Grab IDs for triple A games on June 1st, 2011
# This post explains more about obtaining game IDs with regular expressions --
# http://baseballwithr.wordpress.com/2014/06/30/pitchrx-meet-openwar-4/
aaa <- nonMLBgids[grepl("2011_06_01_[a-z]{3}aaa_[a-z]{3}aaa", nonMLBgids)]
dat <- scrape(game.ids = aaa)

# Create SQLite database, then collect and store data in that database
library(dplyr)
my_db <- src_sqlite("Gameday.sqlite3")
scrape(start = "2013-08-01", end = "2013-08-01", connect = my_db$con)

# Collect other data complementary to PITCHf/x and store in database
files <- c("inning/inning_hit.xml", "miniscoreboard.xml", "players.xml")
scrape(start = "2013-08-01", end = "2013-08-01", connect=my_db$con, suffix = files)

# Simple example to demonstrate database query using dplyr
# Note that 'num' and 'gameday_link' together make a key that allows us to join these tables
locations <- select(tbl(my_db, "pitch"), px, pz, des, num, gameday_link)
names <- select(tbl(my_db, "atbat"), pitcher_name, batter_name, num, gameday_link)
que <- inner_join(locations, filter(names, batter_name == "Paul Goldschmidt"),
                  by = c("num", "gameday_link"))
que$query #refine sql query if you'd like
pitchfx <- collect(que) #submit query and bring data into R


## End(Not run)
```

---

scrapeFX *Scrape Major League Baseball's PITCHf/x Data*

Description

This function is deprecated as of version 1.0

Usage

```
scrapeFX(start, end, tables = list())
```

Arguments

start	date "yyyy-mm-dd" to commence scraping of pitch F/X data
end	date "yyyy-mm-dd" to terminate scraping pitch F/X data
tables	XML nodes to be parsed into a data frame

See Also

scrape

---

strikeFX *Visualize PITCHf/x strikezones*

Description

A suite of bivariate plots with "px" on the horizontal axis and "pz" on the vertical axis.

Usage

```
strikeFX(data, geom = "point", contour = FALSE, point.size = 3,
  point.alpha = 1/3, color = "pitch_types", fill = "des",
  layer = list(), model, model.save = TRUE, density1 = list(),
  density2 = list(), limitz = c(-2, 2, 0.5, 4.5), adjust = FALSE,
  draw_zones = TRUE, parent = FALSE, ...)
```

Arguments

data	PITCHf/x data to be visualized.
geom	plotting geometry. Current choices are: "point", "hex", "bin", "tile" and "raster"
contour	logical. Should contour lines be included?
point.size	Size of points (when geom="point")
point.alpha	plotting transparency parameter (when geom="point").
color	variable used to define coloring scheme.
fill	variable used to define subplot scheme (when geom="subplot2d").
layer	list of other ggplot2 (layered) modifications.

model	Either a gamObject or a call to fit a model via gam or bam. Note that the horizontal and vertical location of the pitch MUST be included as covariates named "px" and "pz", respectively. Relevant factor variables must also be included as covariates in order to produce facetted or differenced plot(s). If this option is used, the geometry must be either "hex", "tile" or "bin". If a non-valid geometry is used, the geometry will be forced to "tile".
model.save	logical. Save the fitted model? If TRUE, the relevant model object is assigned to the global environment
density1	List of length one. The name should correspond to a variable in data. The value should correspond to an (observed) value of that variable.
density2	Similar to density1. If density1 != density2, the relevant estimates are automatically differenced.
limitz	limits for horizontal and vertical axes.
adjust	logical. Should vertical locations be adjusted according to batter height?
draw_zones	logical. Should strikezones be included?
parent	is the function being called from a higher-level function? (experimental)
...	extra options passed onto geom commands

Value

Returns an object of the class used by package ggplot2 to represent plots.

Examples

```
data(pitches)

strikeFX(pitches)
## Not run:
strikeFX(pitches, layer=facet_grid(.~stand))
 #silly example on how to modify default settings and add layers
 strikeFX(pitches, color="", layer=facet_grid(s~stand))+
 geom_point(aes(x=px, y=pz, shape=pitch_types))+ #you could add color here
 geom_text(aes(x=px+0.5, y=pz, label=b))

 p <- strikeFX(pitches, geom="tile", layer=facet_grid(.~stand))
 p+theme(aspect.ratio=1)

strikeFX(pitches, geom="hex", density1=list(des="Called Strike"), density2=list(des="Ball"),
         draw_zones=FALSE, contour=TRUE, layer=facet_grid(.~stand))

noswing <- subset(pitches, des %in% c("Ball", "Called Strike"))
noswing$strike <- as.numeric(noswing$des %in% "Called Strike")
library(mgcv)
m1 <- bam(strike ~ s(px, pz, by=factor(stand)) +
               factor(stand), data=noswing, family = binomial(link='logit'))
# geom will automatically be set to 'raster'
strikeFX(noswing, model=m1, layer=facet_grid(.~stand))

m2 <- bam(strike ~ s(px, pz, by=factor(stand)) + s(px, pz, by=factor(inning_side)) +
```

```
        factor(stand) + factor(inning_side), data=noswing, family = binomial(link='logit'))
strikeFX(noswing, model=m2, density1=list(inning_side="top"),
        density2=list(inning_side="bottom"), layer=facet_grid(.~stand))

## End(Not run)
```

---

update_db *Update an existing PITCHf/x database*

Description

Data from games played starting the day after the most recent date in the database are appended to the appropriate tables.

Usage

```
update_db(connect, end = Sys.Date() - 1, ...)
```

Arguments

connect	Either an SQLite or MySQL database connection
end	date to stop data collection. The default value of 'yesterday' is recommended to ensure the update performs properly.
...	arguments passed onto scrape

Details

Using this function requires the DBI package

See Also

http://baseballwithr.wordpress.com/2014/04/13/modifying-and-querying-a-pitchfx-database-with-dplyr/

Examples

```
## Not run:
 library(dplyr)
 db <- src_sqlite("pitchRx.sqlite3")
 update_db(db$con)

## End(Not run)
```

`urlsToDataFrame` *Parse XML files into data frame(s)*

Description

This function is deprecated as of version 1.0

Usage

```
urlsToDataFrame(urls, tables = list(), add.children = FALSE,
  use.values = FALSE)
```

Arguments

`urls`	set of urls for parsing
`tables`	list of character vectors with appropriate names. The list names should correspond to XML nodes of interest within the XML files.
`add.children`	logical parameter specifying whether to scrape the XML children of the node(s) specified in `tables`.
`use.values`	logical parameter specifying whether to extract XML attributes or values of the node(s).

Index