Package: gtreeWebPlay (via r-universe)

August 31, 2024

Type Package

Title Create shiny apps to play gtree games

Version 0.0.1

Date 2019-06-14

Author Sebastian Kranz

Maintainer Sebastian Kranz <sebastian.kranz@uni-ulm.de>

Description Create shiny web apps that allow single users to play game-theoretic gtree games against the computer who can e.g. follow equilibrium strategies, or average behavior of the population of all players so far.

License GPL >= 2.0

URL https://github.com/skranz/gtreeWebPlay

Depends gtree, shinyEvents, shiny, rmdtools

Suggests knitr,rmarkdown

VignetteBuilder knitr

RoxygenNote 6.0.1

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

RemoteUrl https://github.com/skranz/gtreeWebPlay

RemoteRef master

RemoteSha d2ff09b5aed9297d65f05b4a5f50f7c4cbe7d892

Contents

bot_eq *Bot that plays according to a specified equilibrium*

Description

Bot that plays according to a specified equilibrium

Usage

```
bot_eq(game, player, eq = game$eq.li[[1]], eq.tables = gtree::eq_tables(game
  = game, eq.li = list(eq)), name = "eq_bot", ...)
```

Arguments

game	the game object
player	the player number of this bot
eq	an equilibrium, typically an element of game$eq.li

See Also

Other Bots: bot_mixture, bot_pop, bot_random, bot_tables, make_bots, play_bot_vs_bot

`bot_mixture` *Bot that mixes between different bots*

Description

The first time the bot is called for a particular player in a play He picks a child bot randomly. Then it continues with that child bot for this player the whole play.

Usage

```
bot_mixture(game, player, child_bots, prob = NULL, ...)
```

Arguments

`game`	the game object
`player`	the player number of this bot
`child_bots`	A list of child pots
`prob`	A vector of weights for each child bot. If `NULL` (default) all are equally likely.

Details

If you use `make_bots` or call repeatedly `bot_mixture` to generate mixture bots for each player, the child bots will be independently drawn for each player.

Instead, if bot1 is a mixture bot for player 1 and you create a mixture bot for player 2 by `bot2 = bot1\nbot2$player = 2` then bot2 will select in every play the same child bot than bot1.

See Also

Other Bots: `bot_eq`, `bot_pop`, `bot_random`, `bot_tables`, `make_bots`, `play_bot_vs_bot`

`bot_pop` *Bot who mimics the average player population*

Description

Draws actions from previous actions stored in population play summary (pps) object

Usage

```
bot_pop(game, player, pps, alt.bot = NULL, alt.bot.count = 5,
  name = "pop_bot", alt.bot.fun = bot_random, ...)
```

Arguments

`game`	the game object
`player`	the player number of this bot
`pps`	a pps object, can be extended during play.
`alt.bot`	a bot who will be aksed if there are too few observations in the popoulation
`alt.bot.count`	we assume that we already have this many observations for alt.bot. This determines the probability to draw from the alt.bot instead of the population

See Also

Other Bots: `bot_eq`, `bot_mixture`, `bot_random`, `bot_tables`, `make_bots`, `play_bot_vs_bot`

Other population play functions: `new_pps`, `pps_add_play_actions`, `pps_rearrange`

`bot_random` *Bot that chooses all actions randomly*

Description

Always picks each possible move with equal probability

Usage

```
bot_random(game, player, ...)
```

Arguments

`game`	the game object
`player`	the player number of this bot

See Also

Other Bots: `bot_eq`, `bot_mixture`, `bot_pop`, `bot_tables`, `make_bots`, `play_bot_vs_bot`

`bot_tables` *Bot whose actions are determined by key-action tables*

Description

Bot whose actions are determined by key-action tables

Usage

```
bot_tables(game, player, tables, name = "table_bot", ...)
```

Arguments

`game`	the game object
`player`	the player number of this bot
`tables`	a list of tables for each action. The result of `eq_tables` is

See Also

Other Bots: `bot_eq`, `bot_mixture`, `bot_pop`, `bot_random`, `make_bots`, `play_bot_vs_bot`

`deploy_webplay_example` *Deploys an example app to local directory*

Description

Deploys an example app to local directory

Usage

```
deploy_webplay_example(example = c("UltimatumGame", "KuhnPoker")[1],
  dest.dir = file.path(getwd(), example))
```

Arguments

`example`	Name of the example. Current options are \n\t1. `"UltimatumGame"` a simple introductionary app and \n\t2.`"KuhnPoker"` a more complex app that implements a `bot_pop` to play against the population of earlier players.
`dest.dir`	The destination directory

get_wp — *Gets the web play object of the current app instance*

Description

Gets the web play object of the current app instance

Usage

```
get_wp(app = getApp())
```

See Also

set_wp_for_app

Other Web Play: new_wp, set_wp_for_app, wpDevelApp, wp_copy, wp_developer_ui, wp_reset, wp_set_to_play

make_bots — *Convenience function to create a list of bots for all players*

Description

Every player gets the same bot type

Usage

```
make_bots(game, bot_fun, ..., players = game$players)
```

Arguments

game	the game object
bot_fun	a bot function like e.g. bot_eq
...	additional arguments passed to bot_fun

See Also

Other Bots: bot_eq, bot_mixture, bot_pop, bot_random, bot_tables, play_bot_vs_bot

new_pps — *Create a new empty population play summary*

Description

Create a new empty population play summary

Usage

```
new_pps(...)
```

See Also

Other population play functions: bot_pop, pps_add_play_actions, pps_rearrange

new_wp — *Create a new web play object*

Description

Create a new web play object

Usage

```
new_wp(game, bots, human = draw_human_pos(human_draw_method =
  human_draw_method, numPlayers = game$vg$params$numPlayers, human = 0),
  human_draw_method = c("cycle", "random", "fixed")[1], wpUI = "wpUI",
  verbose = FALSE, pages.dir = file.path(getwd(), "pages"),
  custom = list(), pre.page.handler = NULL, post.page.handler = NULL,
  finish.handler = wp.default.finish.handler,
  comp.pages = as.environment(list()), page.ui.fun = NULL, ...)
```

Arguments

game	A gtree game generated with new_game.
bots	A list with one bots for every player. Also add a bot for the human player. You can call make_bots to conveniently create the bots.
human	index of the player that is played by a human in the first play of the web app.
human_draw_method	Method how the index of the human player is determined by default if a new play is started. The default "cycle" lets the human cycle through all players. "random" picks a random player, and "fixed" keeps the current player.
wpUI	the id of the uiOutput element in the app ui where the web play will be shown.
verbose	shall information about state of play be printed to the standard output?

`pages.dir`	the directory in which the Rmd files for the stage pages can be found. By default `"./pages"`.
`custom`	A list of custom parameters that will be passed to handlers.
`pre.page.handler`	a function that is called before a page is shown to a human. It should return a list of values that can be accessed in the whiskers of the page Rmd file.
`post.page.handler`	a function that is called after a human made input in a stage. Can for example be used to update a population play summary. (See the KuhnPoker example)
`finish.handler`	is called when the final results page of a play is left. The default handler simply starts a new play.
`page.ui.fun`	optionally a function that returns for each page a shiny tag that will be shown. If NULL (default) we specify the page ui via Rmd files in the pages subfolder.

See Also

Other Web Play: `get_wp`, `set_wp_for_app`, `wpDevelApp`, `wp_copy`, `wp_developer_ui`, `wp_reset`, `wp_set_to_play`

`play_bot_vs_bot` *Simulate one play of the game*

Description

Simulate one play of the game

Usage

```
play_bot_vs_bot(game, bots, return.play.object = FALSE)
```

Arguments

`game`	the game object
`bots`	a list containing one bot per player
`return.play.object`	By default only the outcome of the play as a one-row data frame is returned. If you set `return.play.object` an internal `play` object will be returned with more detailed information about the simulation run

See Also

Other Bots: `bot_eq`, `bot_mixture`, `bot_pop`, `bot_random`, `bot_tables`, `make_bots`

`pps_add_play_actions` *Call this function in the post.page.handler to update the population play summary*

Description

Call this function in the post.page.handler to update the population play summary

Usage

```
pps_add_play_actions(pps, play, stage.num = play$human.stage.finished)
```

See Also

Other population play functions: `bot_pop`, `new_pps`, `pps_rearrange`

`pps_rearrange` *Order pps columns naturally*

Description

If the pps is dynamically created during plays the column order may change

Usage

```
pps_rearrange(pps)
```

See Also

Other population play functions: `bot_pop`, `new_pps`, `pps_add_play_actions`

`set_wp_for_app` *This function should be called in the appInitHandler of your shinyEvents app*

Description

Assigns a copy of a global web play object to the current instance of the shiny app. This means every user has her own instance. Note that it is not possible to have two or more web plays active in the same app.

Usage

```
set_wp_for_app(wp, app = getApp(), copy = TRUE)
```

Details

The function get_wp() returns the web play object of the current app instance.

See Also

Other Web Play: `get_wp`, `new_wp`, `wpDevelApp`, `wp_copy`, `wp_developer_ui`, `wp_reset`, `wp_set_to_play`

`wpDevelApp` *Create a simple app for testing and developing a gtree web play*

Description

Returns a shinyEvents app. You can view the app in RStudio using `[shinyEvents]viewApp`

Usage

```
wpDevelApp(wp, title = paste0("Playing ", wp$play$game$gameId))
```

Arguments

`wp`	a web play object generated with `new_wp`
`title`	A title string

See Also

Other Web Play: `get_wp`, `new_wp`, `set_wp_for_app`, `wp_copy`, `wp_developer_ui`, `wp_reset`, `wp_set_to_play`

`wp_copy` *Copy a web play object*

Description

Copy a web play object

Usage

```
wp_copy(wp)
```

See Also

Other Web Play: `get_wp`, `new_wp`, `set_wp_for_app`, `wpDevelApp`, `wp_developer_ui`, `wp_reset`, `wp_set_to_play`

wp_developer_ui *A developer toolbar to your web play app*

Description

Returns a shiny tag list that you can add to your app$ui definition. It contains buttons to restart the experiment, edit the page rmd file in RStudio and to refresh an edited page. Button handlers are automatically added.

Usage

```
wp_developer_ui()
```

See Also

Other Web Play: `get_wp`, `new_wp`, `set_wp_for_app`, `wpDevelApp`, `wp_copy`, `wp_reset`, `wp_set_to_play`

wp_reset *Reset a web play to the start of a new play*

Description

If you immediately want to start the new play. Call `wp_continue` afterwards.

Usage

```
wp_reset(wp = get_wp(), bots = wp$play$bots, human = draw_human_pos(wp))
```

Arguments

`wp`	A web play object
`bots`	You can provide new bots. By default the current bots are used again.
`human`	You can define a new index of the human player. By default the current human is used.

See Also

Other Web Play: `get_wp`, `new_wp`, `set_wp_for_app`, `wpDevelApp`, `wp_copy`, `wp_developer_ui`, `wp_set_to_play` wp_set_to_play *Sets the state of a web play to a play object*

Description

Can for example be used to continue with a human after bots played some earlier rounds

Usage

```
wp_set_to_play(wp, play, human = play$human)
```

See Also

Other Web Play: get_wp, new_wp, set_wp_for_app, wpDevelApp, wp_copy, wp_developer_ui, wp_reset

Index