Package ‘bootwar’

October 1, 2023

Title  Nonparametric Bootstrap Test with Pooled Resampling Card Game

Version  0.2.1

Description  The card game War is simple in its rules but can be lengthy. In another domain, the nonparametric bootstrap test with pooled resampling (nbpr) methods, as outlined in Dwivedi, Mallawaarachchi, and Alvarado (2017) <doi:10.1002/sim.7263>, is optimal for comparing paired or unpaired means in non-normal data, especially for small sample size studies. However, many researchers are unfamiliar with these methods. The 'bootwar' package bridges this gap by enabling users to grasp the concepts of nbpr via Boot War, a variation of the card game War designed for small samples. The package provides functions like score_keeper() and play_round() to streamline gameplay and scoring. Once a predetermined number of rounds concludes, users can employ the analyze_game() function to derive game results. This function leverages the 'npboottprm' package's nonparboot() to report nbpr results and, for comparative analysis, also reports results from the 'stats' package's t.test() function. Additionally, 'bootwar' features an interactive 'shiny' web application, bootwar(). This offers a user-centric interface to experience Boot War, enhancing understanding of nbpr methods across various distributions, sample sizes, number of bootstrap resamples, and confidence intervals.

License  MIT + file LICENSE

Encoding  UTF-8

RoxygenNote  7.2.3

URL  https://github.com/mightymetrika/bootwar

BugReports  https://github.com/mightymetrika/bootwar/issues

Imports  ggplot2, mmcards, npboottprm, shiny, shinyjs, shinythemes

Depends  R (>= 2.10)

LazyData  true

Suggests  knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition  3

VignetteBuilder  knitr
analyze_game

Description
This function analyzes the results of the game using both nonparametric bootstrap with pooled resampling and classical t-tests. It then determines the winner based on the bootstrap results and effect size.

Usage
analyze_game(plyr_vv, comp_vv, mode = "t", conf.level = 0.95, ...)

Arguments
- plyr_vv: A numeric vector storing the values of the cards dealt to the player.
- comp_vv: A numeric vector storing the values of the cards dealt to the computer.
- mode: A character string indicating the type of test. Valid options are "t" for independent t-test and "pt" for paired t-test. Default is "t".
- conf.level: A confidence level for \texttt{npbootprm::nonparboot}, \texttt{stats::t.test}. The confidence level is also used to set the alpha level to $\alpha = 1 - \text{conf.level}$
- ...: Additional arguments passed to the \texttt{npbootprm::nonparboot} function.

Value
A list containing:
- bootstrap_results: A list containing results from the bootstrap test.
- classical_results: A list containing results from the classical t-test.
- winner: A character string indicating the winner ("Player Wins", "Computer Wins", or "Draw").
Examples

# Analyze a sample game
plyr_values <- c(4, 3, 2, 1)
comp_values <- c(1, 2, 3, 4)
game_results <- analyze_game(plyr_values, comp_values, nboot = 1000,
                           mode = "t", seed = 150)

Description

Launches a Shiny application for the Bootwar card game. The app allows users to play a card game where they can analyze the game results using nonparametric bootstrap test with pooled resampling methods.

Usage

bootwar()

Details

The Bootwar card game is a bootstrap variation of the card game War. The Bootwar application has options to select different modes ('t' for independent t-test and 'pt' for paired t-test) and decks. Players can use a standard 52 card deck and they can also input a custom anonymous function to generate a deck. The app will let users deal cards, play the game, and then score and analyze results using nonparametric bootstrap test with pooled resampling methods. The game is designed to help users gain greater intuition on nonparametric bootstrap test with pooled resampling methods; as such, players are encouraged to experiment with different confidence levels, number of rounds, number of bootstrap resamples, and custom decks.

Value

A Shiny application object. Running this function will launch the Shiny app in the user's default web browser.

Examples

if(interactive()){
  bootwar()
}
deck

Description
A 52 card deck of playing cards with suit ranking.

Usage
deck

Format
deck:
A data frame with 52 rows and 4 columns:

rank  A factor representing card rank taking values 2 - A
suit  A card suit with ranked order Club (C), Diamond (D), Heart (H), and Spade (S)
card  A card
value  A card value ranging from 2.00 (2C) to 14.75 (AS)

Source
Standard Deck of Playing Cards

play_round

Description
This function simulates a single round of the card game, where both the computer and the player are dealt a card. The function returns the updated state of the game after the round.

Usage
play_round(
cdeck,
plyr_cv,
plyr_vv,
plyr_ic = NULL,
comp_cv,
comp_vv,
comp_ic = NULL
)
Arguments

cdeck A dataframe representing the current deck of cards.
plyr_cv A character vector storing the cards dealt to the player so far.
plyr_vv A numeric vector storing the values of the cards dealt to the player so far.
plyr_ic A character vector storing the image cards dealt to the player. Default is NULL.
comp_cv A character vector storing the cards dealt to the computer so far.
comp_vv A numeric vector storing the values of the cards dealt to the computer so far.
comp_ic A character vector storing the image cards dealt to the computer. Default is NULL.

Value

A list containing:

- updated_deck: A dataframe representing the updated deck of cards after the round.
- plyr_cv: Updated character vector of cards dealt to the player.
- plyr_vv: Updated numeric vector of values of cards dealt to the player.
- plyr_ic: Updated character vector of image cards dealt to the player.
- comp_cv: Updated character vector of cards dealt to the computer.
- comp_vv: Updated numeric vector of values of cards dealt to the computer.
- comp_ic: Updated character vector of image cards dealt to the computer.

Examples

# Simulate a round of the game with a sample deck
deck <- mmcards::shuffle_deck()
plyr_cards <- character(0)
plyr_values <- numeric(0)
comp_cards <- character(0)
comp_values <- numeric(0)
round_result <- play_round(deck, plyr_cv = plyr_cards, plyr_vv = plyr_values,
                           comp_cv = comp_cards, comp_vv = comp_values)

score_keeper Calculate Scores and Effect Size

Description

This function computes the sum and mean of the player’s and computer’s values and calculates the effect size based on the given mode (t or pt).

Usage

score_keeper(player_values, comp_values, mode)
Arguments

player_values  A numeric vector representing the values of the player’s cards.
comp_values   A numeric vector representing the values of the computer’s cards.
mode          A character string representing the mode of the game, either ‘t’ for independent
              t-test or ‘pt’ for paired t-test.

Value

A list containing:

- player_sum: Sum of player’s values.
- player_mean: Mean of player’s values.
- comp_sum: Sum of computer’s values.
- comp_mean: Mean of computer’s values.
- effect_size: Calculated effect size based on the given mode.

Examples

# Calculate scores for a simple game
player_vals <- c(2.5, 3.0, 4.5)
comp_vals <- c(3.5, 2.0, 4.0)
scores <- score_keeper(player_vals, comp_vals, mode = "t")
Index

* datasets
  deck, 4

analyze_game, 2

bootwar, 3

deck, 4

play_round, 4

score_keeper, 5