Package 'runexp'

October 14, 2022

Type Package

Title Softball Run Expectancy using Markov Chains and Simulation

Version 0.2.1

Depends R (>= 3.6)

Description Implements two methods of estimating runs scored in a softball scenario: (1) theoretical expectation using discrete Markov chains and (2) empirical distribution using multinomial random simulation. Scores are based on player-specific input probabilities (out, single, double, triple, walk, and homerun). Optional inputs include probability of attempting a steal, probability of succeeding in an attempted steal, and an indicator of whether a player is ``fast'' (e.g. the player could stretch home). These probabilities may be calculated from common player statistics that are publicly available on team's webpages. Scores are evaluated based on a nine-player lineup and may be used to compare lineups, evaluate base scenarios, and compare the offensive potential of individual players. Manuscript forthcoming. See Bukiet & Harold (1997) <doi:10.1287/opre.45.1.14> for implementation of discrete Markov chains.

License LGPL

LazyData true

Encoding UTF-8

Suggests xml2, rvest

Imports doParallel, foreach, parallel

RoxygenNote 7.1.1

NeedsCompilation no

Author Annie Sauer [aut, cre],
Sierra Merkes [aut]

Maintainer Annie Sauer <anniees@vt.edu>

Repository CRAN

Date/Publication 2021-03-22 05:10:02 UTC

R topics documented:

`runexp-package` *Package runexp*

Description

Implements two methods of estimating runs scored in a softball scenario: (1) theoretical expectation using discrete Markov chains and (2) empirical distribution using multinomial random simulation. Scores are based on player-specific input probabilities (out, single, double, triple, walk, and home-run). Optional inputs include probability of attempting a steal, probability of succeeding in an attempted steal, and an indicator of whether a player is "fast" (e.g. the player could stretch home). These probabilities may be calculated from common player statistics that are publicly available on team's webpages. Scores are evaluated based on a nine-player lineup and may be used to compare lineups, evaluate base scenarios, and compare the offensive potential of individual players. Manuscript forthcoming. See Bukiet & Harold (1997) <doi:10.1287/opre.45.1.14> for implementation of discrete Markov chains.

Important Functions

- `chain`: calculates run expectancy using discrete Markov chains
- `sim`: estimates run expectancy using multinomial simulation
- `plot.chain`: S3 method for plotting `chain` output objects
- `prob_calc`: calculates player probabilities from commonly available stats
- `scrape`: scrapes player statistics from a given URL

Data Files

- `wku_stats`: player statistics for the 2013 Western Kentucky University softball team
- `wku_probs`: calculated player probabilities for the 2013 Western Kentucky University softball team

Author(s)

Annie Sauer <anniees@vt.edu>

Sierra Merkes <smerkes@vt.edu>

References

B. Bukiet, E. R. Harold, and J. L. Palacios, "A Markov Chain Approach to Baseball," Operations Research 45, 14–23 (1997).

Examples

```
# see "?scrape", "?prob_calc", "?chain" and "?sim" for relevant examples
```

`chain` *Softball run expectancy using discrete Markov chains*

Description

Uses discrete Markov chains to calculate softball run expectancy for a single (half) inning. Calculations depend on specified player probabilities (see details) and a nine-player lineup. Optionally incorporates attempted steals and "fast" players who are able to strech bases.

Usage

```
chain(lineup, stats, cycle = FALSE, max_at_bats = 18)
```

Arguments

`lineup`	either character vector of player names or numeric vector of player numbers. Must be of length 1 or 9. If lineup is of length 1, the single player will be "copied" nine times to form a complete lineup.
`stats`	data frame of player statistics (see details)
`cycle`	logical indicating whether to calculate run expectancy for each of the 9 possible lead-off batters. Preserves the order of the lineup. As a default, only the first player in `lineup` is used as lead-off. Cycling is not relevant when the lineup is made up of a single player.
`max_at_bats`	maximum number of at bats (corresponding to matrix powers) used in calculation. Must be sufficiently large to achieve convergence. Convergence may be checked using `plot` with `type = 1`.

Details

The typical state space for softball involves 25 states defined by the base situation (runners on base) and number of outs. The standard base situations are: (1) bases empty, (2) runner on first, (3) runner on second, (4) runner on third, (5) runners on first and second, (6) runners on second and third, (7) runners on first and third, and (8) bases loaded. These 8 states are crossed with each of three out states (0 outs, 1 out, or 2 outs) to form 24 states. The final 25th state is the 3 outs that marks the end of an inning.

We expand these 25 states to incorporate "fast" players. We make the following assumptions concerning fast players:

- If a fast player is on first and the batter hits a single, the fast player will stretch to third base (leaving the batter on first).
- If a fast player is on second and the batter hits a single, the fast player will stretch home (leaving the batter on first and a single run scored).

- If a fast player is on first and the batter hits a double, the fast player will stretch home (leaving the batter on second base and a single run scored).
- A typical player (not fast) who successfully steals a base will become a fast player for the remainder of that inning (meaning that a player who successfully steals second base will stretch home on a single).

Based on these assumptions, we add base situations that designate runners on first and second base as either typical runners (R) or fast runners (F). The entirety of these base situations can be viewed using `plot.chain` with `fast = TRUE`. Aside from these fast player assumptions, runners advance bases as expected (a single advances each runner one base, a double advances each runner two bases, etc.).

Each at bat results in a change to the base situation and/or the number of outs. The outcomes of an at-bat are limited to:

- batter out (O): base state does not change, outs increase by one
- single (S): runners advance accordingly, score may increase, outs do not change
- double (D): runners advance accordingly, score may increase, outs do not change
- triple (TR): runners advance accordingly, score may increase, outs do not change
- homerun (HR): bases cleared, score increases accordingly, outs do not change
- walk (W): runners advance accordingly, score may increase, outs do not change

The transitions resulting from these outcomes are stored in "transition matrices." We utilize separate transition matrices for typical batters and fast batters (in order to keep fast runners designated separately). We additionally incorporate stolen bases. Steals are handled separately than the six at-bat outcomes because they do not result in changes to the batter. Following softball norms, we only entertain steals of second base. Steals are considered in cases when there is a runner on first and no runner on second. In this situation, steal possibilities are limited to:

- no steal attempt: base situation and outs do not change
- successful steal: runner advances to second base
- caught steal: runner is removed, outs increase by one

Steal possibilities are implemented in separate transition matrices. All transition matrices are stored as internal RData files.

The `stats` input must be a data frame containing player probabilities. It must contain columns "O", "S", "D", "TR", "HR", and "W" whose entries are probabilities summing to one, corresponding to the probability of a player's at-bat resulting in each outcome. The data frame must contain either a "NAME" or "NUMBER" column to identify players (these must correspond to the `lineup`). Extra rows for players not in the lineup will be ignored. This data frame may be generated from player statistics using `prob_calc`.

The `stats` data frame may optionally include an "SBA" (stolen base attempt) column that provides the probability a given player will attempt a steal (provided they are on first base with no runner on second). If "SBA" is specified, the data frame must also include a "SB" (stolen base) column that provides the probability of a given player successfully stealing a base (conditional on them attempting a steal). If these probabilities are not specified, calculations will not involve any steals.

The stats data frame may also include a logical "FAST" column that indicates whether a player is fast. If this column is not specified, the "FAST" designation will be assigned based on each player's "SBA" probability. Generally, players who are more likely to attempt steals are the fast players.

The cycle parameter is a useful tool for evaluating an entire lineup. Through the course of a game, any of the nine players may lead-off an inning. A weighted or un-weighted average of these nine expected scores provides a more holistic representation of the lineup than the expected score based on a single lead-off.

Value

A list of the S3 class "chain" with the following elements:

- lineup: copy of input lineup
- stats: copy of input stats
- score_full: list of matrices containing expected score by each base/out state and the number of at-bats (created by matrix powers). List index corresponds to lead-off batter. Rows of matrix correspond to base/out states. Each column represents an additional matrix power. Used to assess convergence of the chain (through convergence of each row).
- score_state: matrix of expected score at the completion of an inning based on starting base/out state. Rows correspond to initial state; columns correspond to lead-off batter. Equal to the final column of score_full.
- score: vector of expected score for an entire inning (starting from zero runners and zero outs). Index corresponds to lead-off batter. Equal to the first row of score_state.
- time: computation time in seconds

References

B. Bukiet, E. R. Harold, and J. L. Palacios, "A Markov Chain Approach to Baseball," Operations Research 45, 14–23 (1997).

Examples

```
# Expected score for single batter (termed "offensive potential")
chain1 <- chain("B", wku_probs)
plot(chain1)

# Expected score without cycling
lineup <- wku_probs$name[1:9]
chain2 <- chain(lineup, wku_probs)
plot(chain2)

# Expected score with cycling
chain3 <- chain(lineup, wku_probs, cycle = TRUE)
plot(chain3, type = 1:3)


# GAME SITUATION COMPARISON OF CHAIN AND SIMULATOR

# Select lineup made up of the nine "starters"
```

```
lineup <- sample(wku_probs$name[1:9], 9)

# Average chain across lead-off batters
chain_avg <- mean(chain(lineup, wku_probs, cycle = TRUE)$score)

# Simulate full 7 inning game (recommended to increase cores)
sim_score <- sim(lineup, wku_probs, inn = 7, reps = 50000, cores = 1)

# Split into bins in order to plot averages
sim_grouped <- split(sim_score$score, rep(1:100, times = 50000 / 100))

# Plot results
boxplot(sapply(sim_grouped, mean), ylab = 'Expected Score for Game')
points(1, sim_score$score_avg_game, pch = 16, cex = 2, col = 2)
points(1, chain_avg * 7, pch = 18, cex = 2, col = 3)
```

`plot.chain` *Plots an object of S3 class "chain"*

Description

Acts on a "chain" object output from the "chain" function. Plots convergence of chain, expected score by state, and expected score by lead-off batter (if applicable).

Usage

```
## S3 method for class 'chain'
plot(x, type = 1:2, lead_off = 1, fast = FALSE, ...)
```

Arguments

x	object of class "chain"
type	denotes which type of plot to generate - 1, 2, 3, or any combination of these. See details for plot descriptions.
lead_off	an integer 1-9. Denotes which lead-off batter to plot in type 1 and type 2 plots. Lead-off batters 2-9 are only available if chain was calculated with cycle = TRUE.
fast	logical indicating whether to plot additional fast player states in type 1 and type 2 plots.
...	NA

Details

This function generates three types of plots:

- Type 1: Plots chain convergence. Each line corresponds to the expected score from a specific initial base/out state as at-bats are accumulated. If the chain has reached convergence, each line should level off.
- Type 2: Plots expected score by initial base/out state. This plot can be used to compare different states (e.g. is it better to have a runner on second and one out or a runner on first and no outs?).
- Type 3: Plots expected score for an inning based on lead-off batter. Requires a `chain` object that was created with `cycle = TRUE`. The average across all lead-off batters is the most holistic metric for comparing different lineups.

Both type 1 and type 2 plots rely on the specification of a lead-off batter. In states with runners on base and/or outs, the lead-off batter refers to the first batter to come up to to the plate starting in that situation, not the first batter to start the inning. The "true" lead-off batter at the start of the inning corresponds to the R0 (no runners) 0 out case.

Value

No return value, called to generate plots.

Examples

```
# Expected score for single batter (termed "offensive potential")
chain1 <- chain("B", wku_probs)
plot(chain1)

# Expected score without cycling
lineup <- wku_probs$name[1:9]
chain2 <- chain(lineup, wku_probs)
plot(chain2)

# Expected score with cycling
chain3 <- chain(lineup, wku_probs, cycle = TRUE)
plot(chain3, type = 1:3)
```

`prob_calc` *Calculates player probabilities given players' game statistics.*

Description

Uses player statistics to calculate the probability of six possible at bat outcomes (walk, single, double, triple, homerun, or out). Also estimates the probability of a player attempting a steal (SBA) and succeeding in an attempted steal (SB). Player game statistics are commonly available on team's public webpages.

Usage

```
prob_calc(playerData)
```

Arguments

playerData	data frame of the players statistics (details below)

Details

The playerData data frame must contain the following columns of player statistics:

- Name: player name
- Number: player number
- AB: at bats
- BB: walks
- HBP: hit by pitch
- H: hits
- 2B: doubles
- 3B: triples
- HR: homeruns
- ATT: attempted steals
- SB: successfully stolen bases

Plate appearances (PA) are calculated as AB + BB + HBP. The player probabilities are calculated as:

- Walk probability: W = (BB + HBP) / (PA)
- Single probability: S = (H - (2B + 3B + HR)) / (PA)
- Double probability: D = 2B / PA
- Triple probability: TR = 3B / PA
- Home Run probability: HR = HR / PA
- Out probability: O = (PA - (H + BB + HBP)) / PA

Probabilities calculated from limited at bats will not be very useful. Note, this function does not assign TRUE/FALSE values for fast players. These may be manually assigned or will be assigned based on SBA probability when chain or sim functions are called.

SBA (Stolen Base Attempt) is the probability a player will attempt to steal given they are on first base and there is no runner on second. As a default, we estimate a player's SBA probability using a rough thresholding rule based on the team's overall SB probability and the player's SB probability. Essentially, we group the players into three categories:

- Almost Always Attempt to Steal Group: These players receive the SBA probability of the team's overall SB probability which is calculated as (Team's total # of SB)/(Team's total # of SBA)
- 50/50 Attempt to Steal Group: These players receive SBA probability of 0.50

- Never Attempt to Steal Group: These players receive SBA probability of 0.0

We recommend reviewing these default probabilities before proceeding with run expectancy calculations.

Value

a dataframe of the players' probabilities for W, S, D, TR, HR, O, SBA, and SB

Examples

```
probs <- prob_calc(wku_stats)   # probs corresponds to wku_probs
```

---

scrape *Softball Webscraper*

Description

Scrapes the player statistics from a given URL.

Usage

```
scrape(url)
```

Arguments

url	the web address of the teams' data to scrape

Value

A dataframe consisting of each player's Number, NAME, AVG, OPS, AB, R, H, 2B, 3B, HR, RBI, TB, SLG SB, ATT, GP, and GS.

Examples

```
url <-"https://wmubroncos.com/sports/softball/stats/2019"
test <- scrape(url)
test_probs <- prob_calc(test)
```

`sim` *Softball run expectancy using multinomial random trial simulation*

Description

Utilizes a multinomial simulation to simulate a softball game scenario with a specified number of innings (inn) per game over a specified number of games (reps). Calculations depend on specified player probabilities (see details) and a nine-player lineup. Optionally incorporates attempted steals and "fast" players who are able to stretch bases. Optionally utilizes SNOW parallelization.

Usage

```
sim(
  lineup,
  stats,
  inn = 7,
  reps = 100,
  graphic = FALSE,
  waitTime = 2,
  cores = NULL
)
```

Arguments

`lineup`	either character vector of player names or numeric vector of player numbers. Must be of length 1 or 9. If lineup is of length 1, the single player will be "duplicated" nine times to form a complete lineup.
`stats`	data frame of player statistics (see details)
`inn`	number of innings per rep (the default of 7 represents a typical softball game)
`reps`	number of times to repeat the softball game simulation. Can be thought of as number of games.
`graphic`	logical indicating on whether to plot the player base movement. Requires `reps < 4`. Forces `cores = 1`.
`waitTime`	the amount of time to pause before making next plot for play. Only relevant when `graphic = TRUE`.
`cores`	number of cores to utilize in parallel. Defaults to one less than maximum available nodes.

Details

In each simulation, we determine each batter's hit results through a multinomial random trial where the probability of walk (W), single (S), double (D), triple (TR), home run (HR), and batter out (O) are assigned per input player statistics. We incorporate the impact of "fast" players through the following assumptions:

- If a fast player is on first and the batter hits a single, the fast player will stretch to third base (leaving the batter on first).
- If a fast player is on second and the batter hits a single, the fast player will stretch home (leaving the batter on first and a single run scored).
- If a fast player is on first and the batter hits a double, the fast player will stretch home (leaving the batter on second base and a single run scored).
- A typical player (not fast) who successfully steals a base will become a fast player for the remainder of that inning (meaning that a player who successfully steals second base will stretch home on a single).

Aside from these fast player assumptions, runners advance bases as expected (a single advances each runner one base, a double advances each runner two bases, etc.).

Following softball norms, we only entertain steals of second base. Steals are considered in cases when there is a runner on first and no runner on second. In these situations, we use a bernoulli coin flip (based on the runner's SBA probability) to determine whether the runner on first will attempt a steal. In practice, these decisions are commonly left up to coaches. If it is decided that the player will attempt a steal, a second bernoulli coin flip (based on the runner's SB probability) determines whether the steal was successful or whether the player was caught stealing.

The `stats` input must be a data frame containing player probabilities. It must contain columns "O", "S", "D", "TR", "HR", and "W" whose entries are probabilities summing to one, corresponding to the probability of a player's at-bat resulting in each outcome. The data frame must contain either a "NAME" or "NUMBER" column to identify players (these must correspond to the `lineup`). Extra rows for players not in the lineup will be ignored. This data frame may be generated from player statistics using `prob_calc`.

The `stats` data frame may optionally include an "SBA" (stolen base attempt) column that provides the probability a given player will attempt a steal (provided they are on first base with no runner on second). If "SBA" is specified, the data frame must also include a "SB" (stolen base) column that provides the probability of a given player successfully stealing a base (conditional on them attempting a steal). If these probabilities are not specified, calculations will not involve any steals.

The `stats` data frame may also include a logical "FAST" column that indicates whether a player is fast. If this column is not specified, the "FAST" designation will be assigned based on each player's "SBA" probability. Players who are more likely to attempt steals are likely the fast players.

As a default, simulations will be processed in parallel over all but one of the maximum available cores. Parallelization is recommended to reduce computation time. Interactive plotting (`graphic = TRUE`) requires no parallelization and will override specified cores with `cores = 1`.

Value

A list of the S3 class "`sim`" with the following elements:

- `lineup`: copy of input lineup
- `stats`: copy of input stats
- `inn`: copy of input innings
- `score`: a vector containing the scores per each rep (game)
- `score_avg_game`: the average expected score per rep (game). That is, `mean(score)`.

- score_avg_inn: the average expected score per rep (game) per inning. That is, mean(score)/inn. If inn = 1, then score_avg_game = score_avg_inn.
- time: computation time in seconds

Examples

```
# Short simulation (designed to run in less than 5 seconds)
sim1 <- sim("B", wku_probs, inn = 1, reps = 100, cores = 1)

# Simulation with interactive graphic
lineup <- wku_probs$name[1:9]
sim2 <- sim(lineup, wku_probs, inn = 7, reps = 1, graphic = TRUE)

# Simulation for entire game (recommended to increase cores)
sim3 <- sim(lineup, wku_probs, cores = 1)
boxplot(sim3$score)
points(1, sim3$score_avg_game)

# GAME SITUATION COMPARISON OF CHAIN AND SIMULATOR

# Select lineup made up of the nine "starters"
lineup <- sample(wku_probs$name[1:9], 9)

# Average chain across lead-off batters
chain_avg <- mean(chain(lineup, wku_probs, cycle = TRUE)$score)

# Simulate full 7 inning game (recommended to increase cores)
sim_score <- sim(lineup, wku_probs, inn = 7, reps = 50000, cores = 1)

# Split into bins in order to plot averages
sim_grouped <- split(sim_score$score, rep(1:100, times = 50000 / 100))

boxplot(sapply(sim_grouped, mean), ylab = 'Expected Score for Game')
points(1, sim_score$score_avg_game, pch = 16, cex = 2, col = 2)
points(1, chain_avg * 7, pch = 18, cex = 2, col = 3)
```

---

wku_stats *Player statistics and probabilities for WKU softball*

Description

Statistics and calculated probabilities for each player on the 2013 Western Kentucky University softball team. Data is stored in two data frames: wku_stats contains the game statistics for each player and wku_probs contains the calculated probabilities for each player. Player names and numbers have been replaced with random letters/numbers to preserve anonymity.

Usage

```
wku_stats

wku_probs
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 18 rows and 23 columns.

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 18 rows and 11 columns.

Details

`wku_stats` contains the raw player data taken from the 2013 WKU softball team's webpage. `wku_probs` contains the player probabilities calculated using `prob_calc`. Together, they provide an example of calculating probabilities from player statistics. See `?prob_calc` for more details on the columns of the two data frames.

`wku_probs` is designed for use with `chain` and `sim`. It additionally contains a `fast` column that indicates whether each player is considered fast. This column is not necessary for running `chain` and `sim`, since it is equivalent to the default assignments.

Source

https://wkusports.com/sports/softball/stats/

Examples

```
probs <- prob_calc(wku_stats)   # probs corresponds to wku_probs
```

Index