Package ‘footBayes’

January 9, 2025

Type Package

Title Fitting Bayesian and MLE Football Models

Version 1.0.0

Date 2025-01-09

Maintainer Leonardo Egidi <legidi@units.it>

License GPL-2

Description This is the first package allowing for the estimation,
visualization and prediction of the most well-known
football models: double Poisson, bivariate Poisson,
Skellam, student_t, diagonal-inflated bivariate Poisson, and
zero-inflated Skellam. The package allows Hamiltonian
Monte Carlo (HMC) estimation through the underlying Stan
environment and Maximum Likelihood estimation (MLE, for
'static' models only). The model construction relies on
the most well-known football references, such as
Dixon and Coles (1997) <doi:10.1111/1467-9876.00065>,
Karlis and Ntzoufras (2003) <doi:10.1111/1467-9884.00366> and
Egidi, Pauli and Torelli (2018) <doi:10.1177/1471082X18798414>.

URL https://github.com/leoegidi/footbayes

Encoding UTF-8

SystemRequirements pandoc (>= 1.12.3), pandoc-citeproc

Depends R (>= 3.1.0)

Imports rstan (>= 2.18.1), arm, reshape2, ggplot2, ggridges,
bayesplot, matrixStats, extraDistr, parallel, metRology, dplyr,
tidyr, numDeriv, magrittr, rlang

Suggests testthat, knitr (>= 1.37), rmarkdown (>= 2.10), loo

RoxygenNote 7.3.2

LazyData true

BuildManual yes

NeedsCompilation no

Author Leonardo Egidi [aut, cre],
Roberto Macrì Demartino [aut],
Vasilis Palaskas. [aut]

Repository CRAN

Date/Publication 2025-01-09 15:20:02 UTC

Contents

btd_foot 2
compare_foot 5
england 7
foot_abilities 8
foot_prob 9
foot_rank 11
foot_round_robin 12
italy 13
mle_foot 14
plot_btdPosterior 15
plot_logStrength 18
pp_foot 19
print.btdFoot 21
print.compareFoot 22
print.stanFoot 22
priors 23
stan_foot 25

Index 31

btd_foot *Bayesian Bradley-Terry-Davidson Model*

Description

Fits a Bayesian Bradley-Terry-Davidson model using Stan. Supports both static and dynamic ranking models, allowing for the estimation of team strengths over time.

Usage

```
btd_foot(
  data,
  dynamic_rank = FALSE,
  home_effect = FALSE,
 prior_par = list(logStrength = normal(0, 3), logTie = normal(0, 0.3), home = normal(0,
    5)),
  rank_measure = "median",
  ...
)
```

Arguments

`data` A data frame containing the observations with columns:

- `periods`: Time point of each observation (integer >= 1).
- `home_team`: Home team's name (character string).
- `away_team`: Away team's name (character string).
- `match_outcome`: Outcome (1 if home team beats away team, 2 for tie, and 3 if away team beats home team).

The data frame must not contain missing values.

`dynamic_rank` A logical value indicating whether a dynamic ranking model is used (default is `FALSE`).

`home_effect` A logical value indicating the inclusion of a home effect in the model. (default is `FALSE`).

`prior_par` A list specifying the prior distributions for the parameters of interest, using the `normal` function:

- `logStrength`: Prior for the team log-strengths. Default is `normal(0, 3)`.
- `logTie`: Prior for the tie parameter. Default is `normal(0, 0.3)`.
- `home`: Prior for the home effect (`home`). Applicable only if `home_effect = TRUE`. Default is `normal(0, 5)`.

Only normal priors are allowed for this model.

`rank_measure` A character string specifying the method used to summarize the posterior distributions of the team strengths. Options are:

- `"median"`: Uses the median of the posterior samples (default).
- `"mean"`: Uses the mean of the posterior samples.
- `"map"`: Uses the Maximum A Posteriori estimate, calculated as the mode of the posterior distribution.

`...` Additional arguments passed to `stan` (e.g., `iter`, `chains`, `control`).

Value

A list of class `"btdFoot"` containing:

- `fit`: The fitted `stanfit` object returned by `stan`.
- `rank`: A data frame with the rankings, including columns:
  - `periods`: The time period.
  - `team`: The team name.
  - `rank_points`: The estimated strength of the team based on the chosen `rank_measure`.
- `data`: The input data.
- `stan_data`: The data list prepared for Stan.
- `stan_code`: The path to the Stan model code used.
- `stan_args`: The optional parameters passed to (`...`).
- `rank_measure`: The method used to compute the rankings.

Author(s)

Roberto Macrì Demartino <roberto.macridemartino@phd.unipd.it>.

Examples

```
## Not run:

library(dplyr)

data("italy")

italy_2020_2021 <- italy %>%
  dplyr::select(Season, home, visitor, hgoal, vgoal) %>%
  dplyr::filter(Season == "2020" | Season == "2021") %>%
  dplyr::mutate(match_outcome = dplyr::case_when(
    hgoal > vgoal ~ 1,        # Home team wins
    hgoal == vgoal ~ 2,       # Draw
    hgoal < vgoal ~ 3         # Away team wins
  )) %>%
  dplyr::mutate(periods = dplyr::case_when(
    dplyr::row_number() <= 190 ~ 1,
    dplyr::row_number() <= 380 ~ 2,
    dplyr::row_number() <= 570 ~ 3,
    TRUE ~ 4
  )) %>%  # Assign periods based on match number
  dplyr::select(periods, home_team = home,
   away_team = visitor, match_outcome)

# Dynamic Ranking Example with Median Rank Measure
fit_result_dyn <- btd_foot(
  data = italy_2020_2021,
  dynamic_rank = TRUE,
  home_effect = TRUE,
  prior_par = list(
    logStrength = normal(0, 10),
    logTie = normal(0, 5),
    home = normal(0, 5)
  ),
  rank_measure = "median",
  iter = 1000,
  cores = 2,
  chains = 2
)

print(fit_result_dyn)

print(fit_result_dyn, pars = c("logStrength", "home"), teams = c("AC Milan", "AS Roma"))

# Static Ranking Example with MAP Rank Measure
fit_result_stat <- btd_foot(
  data = italy_2020_2021,
  dynamic_rank = FALSE,
  prior_par = list(
    logStrength = normal(0, 10),
    logTie = normal(0, 5),
    home = normal(0, 5)
  ),
  rank_measure = "map",
  iter = 1000,
  chains = 2
)

print(fit_result_stat)

## End(Not run)
```

compare_foot *Compare Football Models using Various Metrics*

Description

Compares multiple football models or directly provided probability matrices based on specified metrics (accuracy, Brier score, ranked probability score, Pseudo R^2, average coverage probability), using a test dataset. Additionally, computes the confusion matrices. The function returns an object of class compareFoot.

Usage

```
compare_foot(
  source,
  test_data,
  metric = c("accuracy", "brier", "ACP", "pseudoR2", "RPS"),
  conf_matrix = FALSE
)
```

Arguments

source A named list containing either:

- Fitted model objects (of class stanFoot or stanfit), each representing a football model.
- Matrices where each matrix contains the estimated probabilities for "Home Win," "Draw," and "Away Win" in its columns.

test_data A data frame containing the test dataset, with columns:

- home_team: Home team's name (character string).
- away_team: Away team's name (character string).
- home_goals: Goals scored by the home team (integer >= 0).
- away_goals: Goals scored by the away team (integer >= 0).

metric A character vector specifying the metrics to use for comparison. Options are:

- "accuracy": Computes the accuracy of each model.
- "brier": Computes the Brier score of each model.
- "RPS": Computes the ranked probability score (RPS) for each model.
- "ACP": Computes the average coverage probability (ACP) for each model.
- "pseudoR2": Computes the Pseudo R^2, defined as the geometric mean of the probabilities assigned to the actual results.

Default is c("accuracy", "brier", "ACP", "pseudoR2", "RPS"), computing the specified metrics.

conf_matrix A logical value indicating whether to generate a confusion matrix comparing predicted outcomes against actual outcomes for each model or probability matrix. Default is FALSE.

Details

The function extracts predictions from each model or directly uses the provided probability matrices and computes the chosen metrics on the test dataset. It also possible to compute confusion matrices.

Value

An object of class compare_foot_output, which is a list containing:

- metrics: A data frame containing the metric values for each model or probability matrix.
- confusion_matrix: Confusion matrices for each model or probability matrix.

Author(s)

Roberto Macrì Demartino <roberto.macridemartino@phd.unipd.it>

Examples

```
## Not run:
library(dplyr)

data("italy")
italy_2000 <- italy %>%
  dplyr::select(Season, home, visitor, hgoal, vgoal) %>%
  dplyr::filter(Season == "2000")

colnames(italy_2000) <- c("periods", "home_team", "away_team", "home_goals", "away_goals")

# Example with fitted models
fit_1 <- stan_foot(data = italy_2000,
                   model = "double_pois", predict = 18)  # Double Poisson model
fit_2 <- stan_foot(data = italy_2000,
                   model = "biv_pois", predict = 18)     # Bivariate Poisson model

italy_2000_test <- italy_2000[289:306, ]


compare_results_models <- compare_foot(
  source = list(double_poisson = fit_1,
                bivariate_poisson = fit_2),
  test_data = italy_2000_test,
  metric = c("accuracy", "brier", "ACP", "pseudoR2", "RPS"),
  conf_matrix = TRUE
)

print(compare_results_models)

## End(Not run)
```

`england` *English league results 1888-2022*

Description

All results for English soccer games in the top 4 tiers from 1888/89 season to 2021/22 season.

Usage

```
england
```

Format

A data frame with 203956 rows and 12 variables:

Date Date of match

Season Season of match - refers to starting year

home Home team

visitor Visiting team

FT Full-time result

hgoal Goals scored by home team

vgoal Goals scored by visiting team

division Division: 1,2,3,4 or 3N (Old 3-North) or 3S (Old 3-South)

tier Tier of football pyramid: 1,2,3,4

totgoal Total goals in game

goaldif Goal difference in game home goals - visitor goals

result Result: H-Home Win, A-Away Win, D-Draw

foot_abilities *Plot football abilities from Stan and MLE models*

Description

Depicts teams' abilities either from the Stan models fitted via the `stan_foot` function or from MLE models fitted via the `mle_foot` function.

Usage

```
foot_abilities(
  object,
  data,
  type = c("attack", "defense", "both"),
  teams = NULL,
  ...
)
```

Arguments

`object` An object either of class `stanfit` or `stanFoot` as given by `stan_foot` function, or class `list` containing the Maximum Likelihood Estimates (MLE) for the model parameters fitted with `mle_foot`.

`data` A data frame containing match data with columns:

- `periods`: Time point of each observation (integer >= 1).
- `home_team`: Home team's name (character string).
- `away_team`: Away team's name (character string).
- `home_goals`: Goals scored by the home team (integer >= 0).
- `away_goals`: Goals scored by the away team (integer >= 0).

`type` Type of ability in Poisson models: one among `"defense"`, `"attack"` or `"both"`.

`teams` An optional character vector specifying team names to include. If `NULL`, all teams are included.

`...` Optional graphical parameters.

Value

Abilities plots for the selected teams: for Poisson models only, red denotes the attack, blue the defense.

Author(s)

Leonardo Egidi `<legidi@units.it>`

Examples

```
## Not run:
library(dplyr)

data("italy")
italy <- as_tibble(italy)

### no dynamics, no prediction

italy_2000_2002 <- italy %>%
 dplyr::select(Season, home, visitor, hgoal, vgoal) %>%
 dplyr::filter(Season=="2000" |  Season=="2001" | Season =="2002")

colnames(italy_2000_2002) <- c("periods", "home_team", "away_team", "home_goals", "away_goals")

fit1 <- stan_foot(data = italy_2000_2002,
                  model="double_pois") # double poisson

fit2 <- stan_foot(data = italy_2000_2002,
                  model="biv_pois")    # bivariate poisson

fit3 <- stan_foot(data = italy_2000_2002,
                  model="skellam")     # skellam

fit4 <- stan_foot(data = italy_2000_2002,
                  model="student_t")   # student_t

foot_abilities(fit1, italy_2000_2002)
foot_abilities(fit2, italy_2000_2002)
foot_abilities(fit3, italy_2000_2002)
foot_abilities(fit4, italy_2000_2002)

### seasonal dynamics, predict the last season

fit5 <-stan_foot(data = italy_2000_2002,
                 model = "biv_pois",
                 predict = 180,
                 dynamic_type = "seasonal")   # bivariate poisson
foot_abilities(fit5, italy_2000_2002)


## End(Not run)
```

foot_prob *Plot football matches probabilities for out-of-sample football matches.*

Description

The function provides a table containing the home win, draw and away win probabilities for a bunch of out-of-sample matches as specified by `stan_foot` or `mle_foot`.

Usage

```
foot_prob(object, data, home_team, away_team)
```

Arguments

`object` An object either of class `stanfit` and `stanFoot` as given by `stan_foot` function or `list` as given by `mle_foot`.

`data` A data frame containing match data with columns:

- `periods`: Time point of each observation (integer >= 1).
- `home_team`: Home team's name (character string).
- `away_team`: Away team's name (character string).
- `home_goals`: Goals scored by the home team (integer >= 0).
- `away_goals`: Goals scored by the away team (integer >= 0).

`home_team` The home team(s) for the predicted matches.

`away_team` The away team(s) for the predicted matches.

Details

For Bayesian models fitted via `stan_foot` the results probabilities are computed according to the simulation from the posterior predictive distribution of future (out-of-sample) matches. For MLE models fitted via the `mle_foot` the probabilities are computed by simulating from the MLE estimates.

Value

A `data.frame` containing the number of out-of-sample matches specified through the argument `predict` passed either in the `mle_foot` or in the `stan_foot` function. For Bayesian Poisson models the function returns also the most likely outcome (mlo) and a posterior probability plot for the exact results, where matches are sorted by the degree of favoritism. Specifically, matches are ordered from those in which the favorite team has the highest posterior probability of winning to those where the underdog is more likely to win.

Author(s)

Leonardo Egidi <legidi@units.it>

Examples

```
## Not run:
library(tidyverse)
library(dplyr)

data("italy")
italy_2000 <- italy %>%
 dplyr::select(Season, home, visitor, hgoal,vgoal) %>%
 dplyr::filter(Season=="2000")

colnames(italy_2000) <- c("periods", "home_team", "away_team", "home_goals", "away_goals")
```

```
fit <- stan_foot(data = italy_2000,
                 model="double_pois",
                 predict = 18)  # double pois

foot_prob(fit, italy_2000, "Inter",
          "Bologna FC")

foot_prob(fit, italy_2000) # all the out-of-sample matches

## End(Not run)
```

foot_rank *Rank and points predictions*

Description

Posterior predictive plots and final rank table for football seasons.

Usage

```
foot_rank(
  object,
  data,
  teams = NULL,
  visualize = c("aggregated", "individual")
)
```

Arguments

object	An object of class stanfit or stanFoot as given by stan_foot function.
data	A data frame containing match data with columns: • periods: Time point of each observation (integer >= 1). • home_team: Home team's name (character string). • away_team: Away team's name (character string). • home_goals: Goals scored by the home team (integer >= 0). • away_goals: Goals scored by the away team (integer >= 0).
teams	An optional character vector specifying team names to include. If NULL, all teams are included.
visualize	Type of plot, default is "aggregated".

Details

For Bayesian models fitted via stan_foot the final rank tables are computed according to the simulation from the posterior predictive distribution of future (out-of-sample) matches. The dataset should refer to one or more seasons from a given national football league (Premier League, Serie A, La Liga, etc.).

Value

Final rank tables and plots with the predicted points for the selected teams as given by the models fitted via the `stan_foot` function.

Author(s)

Leonardo Egidi <legidi@units.it>

Examples

```
## Not run:
library(dplyr)

data("italy")
italy_1999_2000<- italy %>%
dplyr::select(Season, home, visitor, hgoal,vgoal) %>%
dplyr::filter(Season == "1999"|Season=="2000")

colnames(italy_1999_2000) <- c("periods", "home_team", "away_team", "home_goals", "away_goals")

fit <- stan_foot(italy_1999_2000, "double_pois", iter = 200)
foot_rank(fit, italy_1999_2000)
foot_rank(fit, italy_1999_2000, visualize =  "individual")

## End(Not run)
```

`foot_round_robin` *Round-robin for football leagues*

Description

Posterior predictive probabilities for a football season in a round-robin format

Usage

```
foot_round_robin(object, data, teams = NULL)
```

Arguments

`object` An object of class `stanfit` or `stanFoot` as given by `stan_foot` function.

`data` A data frame containing match data with columns:

- `periods`: Time point of each observation (integer >= 1).
- `home_team`: Home team's name (character string).
- `away_team`: Away team's name (character string).
- `home_goals`: Goals scored by the home team (integer >= 0).
- `away_goals`: Goals scored by the away team (integer >= 0).

`teams` An optional character vector specifying team names to include. If `NULL`, all teams are included.

Details

For Bayesian models fitted via `stan_foot` the round-robin table is computed according to the simulation from the posterior predictive distribution of future (out-of-sample) matches. The dataset should refer to one or more seasons from a given national football league (Premier League, Serie A, La Liga, etc.).

Value

Round-robin plot with the home-win posterior probabilities computed from the ppd of the fitted model via the `stan_foot` function.

Author(s)

Leonardo Egidi <legidi@units.it>

Examples

```
## Not run:
library(dplyr)

data("italy")
italy_1999_2000<- italy %>%
dplyr::select(Season, home, visitor, hgoal,vgoal) %>%
dplyr::filter(Season == "1999"|Season=="2000")

colnames(italy_1999_2000) <- c("periods", "home_team", "away_team", "home_goals", "away_goals")

fit <- stan_foot(italy_1999_2000, "double_pois", predict = 45, iter = 200)

foot_round_robin(fit, italy_1999_2000)
foot_round_robin(fit, italy_1999_2000, c("Parma AC", "AS Roma"))


## End(Not run)
```

`italy` *Italy league results 1934-2022*

Description

All results for Italian soccer games in the top tier from 1934/35 season to 2021/22 season.

Usage

```
italy
```

Format

A data frame with 27684 rows and 8 variables:

Date Date of match

Season Season of match - refers to starting year

home Home team

visitor Visiting team

FT Full-time result

hgoal Goals scored by home team

vgoal Goals scored by visiting team

tier Tier of football pyramid: 1

mle_foot *Fit football models with Maximum Likelihood*

Description

ML football modelling for the most famous models: double Poisson, bivariate Poisson, Skellam and student t.

Usage

```
mle_foot(data, model, predict, ...)
```

Arguments

`data`	A data frame, or a matrix containing the following mandatory items: season, home team, away team, home goals, away goals.
`model`	The type of model used to fit the data. One among the following: `"double_pois"`, `"biv_pois"`, `"skellam"`, `"student_t"`.
`predict`	The number of out-of-sample matches. If missing, the function returns the fit for the training set only.
`...`	Optional arguments for MLE fit algorithms.

Details

See documentation of `stan_foot` function for model details. MLE can be obtained only for static models, with no time-dependence. Likelihood optimization is performed via the `BFGS` method of the `optim` function.

Value

MLE and 95% profile likelihood deviance confidence intervals for the model's parameters: attack, defence, home effect and goals' correlation.

Author(s)

Leonardo Egidi <legidi@units.it>

References

Baio, G. and Blangiardo, M. (2010). Bayesian hierarchical model for the prediction of football results. Journal of Applied Statistics 37(2), 253-264.

Egidi, L., Pauli, F., and Torelli, N. (2018). Combining historical data and bookmakers' odds in modelling football scores. Statistical Modelling, 18(5-6), 436-459.

Gelman, A. (2014). Stan goes to the World Cup. From "Statistical Modeling, Causal Inference, and Social Science" blog.

Karlis, D. and Ntzoufras, I. (2003). Analysis of sports data by using bivariate poisson models. Journal of the Royal Statistical Society: Series D (The Statistician) 52(3), 381-393.

Karlis, D. and Ntzoufras,I. (2009). Bayesian modelling of football outcomes: Using the Skellam's distribution for the goal difference. IMA Journal of Management Mathematics 20(2), 133-145.

Owen, A. (2011). Dynamic Bayesian forecasting models of football match outcomes with estimation of the evolution variance parameter. IMA Journal of Management Mathematics, 22(2), 99-113.

Examples

```
## Not run:
require(tidyverse)
require(dplyr)

data("italy")
italy <- as_tibble(italy)
italy_2008<- italy %>%
   dplyr::select(Season, home, visitor, hgoal,vgoal) %>%
   dplyr::filter( Season=="2008")

mle_fit <- mle_foot(data  = italy_2008,
                   model = "double_pois")

## End(Not run)
```

---

plot_btdPosterior *Plot Posterior Distributions for* btdFoot *Objects*

Description

Plots for the posterior distributions of team log-strengths and other parameters with customizable plot types and facets.

Usage

```
plot_btdPosterior(
  x,
  pars = "logStrength",
  plot_type = "boxplot",
  teams = NULL,
  ncol = NULL,
  scales = NULL,
  ...
)
```

Arguments

`x`	An object of class `btdFoot`.
`pars`	A character string specifying the parameter to plot. Choices are `"logStrength"`, `"logTie"`, and `"home"`. Default is `"logStrength"`.
`plot_type`	A character string specifying the type of plot. Choices are `"boxplot"` and `"density"`. Default is `"boxplot"`.
`teams`	An optional character vector specifying team names to include in the posterior boxplots or density plots. If `NULL`, all teams are included.
`ncol`	An optional integer specifying the number of columns in the facet wrap when using a dynamic Bayesian Bradley-Terry-Davidson model. Default is `8`.
`scales`	An optional character string specifying the scales for the facets when using a dynamic Bayesian Bradley-Terry-Davidson model. Options include `"free"`, `"fixed"`, `"free_x"`, and `"free_y"`. Default is `"free_x"`.
`...`	Additional arguments passed to `geom_boxplot()`, `geom_density_ridges()`, or other geoms for customization (e.g., `size`, `alpha`, `color`).

Details

- **Dynamic Ranking**: Faceted boxplots or density plots (including the 95% credible interval) of posterior log-strengths by team and period.
- **Static Ranking**: Boxplots or density plots (including the 95% credible interval) of posterior log-strengths for each team.

Value

A `ggplot` object representing the posterior distributions plot.

Author(s)

Roberto Macrì Demartino <`roberto.macridemartino@phd.unipd.it`>.

Examples

```
## Not run:
library(dplyr)

# Load example data
data("italy")

# Prepare the data
italy_2020_2021_rank <- italy %>%
  select(Season, home, visitor, hgoal, vgoal) %>%
  filter(Season %in% c("2020", "2021")) %>%
  mutate(match_outcome = case_when(
    hgoal > vgoal ~ 1,        # Home team wins
    hgoal == vgoal ~ 2,       # Draw
    hgoal < vgoal ~ 3         # Away team wins
  )) %>%
  mutate(periods = case_when(
    row_number() <= 190 ~ 1,
    row_number() <= 380 ~ 2,
    row_number() <= 570 ~ 3,
    TRUE ~ 4
  )) %>%  # Assign periods based on match number
  select(periods, home_team = home,
         away_team = visitor, match_outcome)

# Fit the Bayesian Bradley-Terry-Davidson model with dynamic ranking
fit_rank_dyn <- btd_foot(
  data = italy_2020_2021_rank,
  dynamic_rank = TRUE,
  rank_measure = "median",
  iter = 1000,
  cores = 2,
  chains = 2
)

# Plot posterior distributions with default settings
plot_btdPosterior(fit_rank_dyn)

# Plot posterior distributions for specific teams with customized facets
plot_btdPosterior(
  fit_rank_dyn,
  teams = c("AC Milan", "AS Roma", "Juventus", "Inter"),
  ncol = 2
)

plot_btdPosterior(
  fit_rank_dyn,
  plot_type = "density",
  teams = c("AC Milan", "AS Roma", "Juventus", "Inter"),
  ncol = 2
)
```

```
## End(Not run)
```

plot_logStrength *Plot Rankings for btdFoot Objects*

Description

Visualizes team rankings based on whether the ranking is dynamic or static.

Usage

```
plot_logStrength(x, teams = NULL, ...)
```

Arguments

x	An object of class btdFoot.
teams	An optional character vector specifying team names to include in the rankings plot. If NULL, all teams are included.
...	Additional arguments passed to geom_line(), geom_point(), and geom_segment() for customization (e.g., size, alpha, color).

Details

- Dynamic Ranking: Plots Rank Points over Periods for each team with lines and points.
- Static Ranking: Plots Rank Points on the x-axis against Team Names on the y-axis with horizontal lines and points.

Value

A ggplot object representing the rankings plot.

Author(s)

Roberto Macrì Demartino <roberto.macridemartino@phd.unipd.it>.

Examples

```
## Not run:
library(dplyr)

data("italy")

italy_2020_2021_rank <- italy %>%
 select(Season, home, visitor, hgoal, vgoal) %>%
 filter(Season == "2020" | Season == "2021") %>%
 mutate(match_outcome = case_when(
   hgoal > vgoal ~ 1,        # Home team wins
```

```
      hgoal == vgoal ~ 2,       # Draw
      hgoal < vgoal ~ 3         # Away team wins
    )) %>%
    mutate(periods = case_when(
      row_number() <= 190 ~ 1,
      row_number() <= 380 ~ 2,
      row_number() <= 570 ~ 3,
      TRUE ~ 4
    )) %>%  # Assign periods based on match number
    select(periods, home_team = home,
                  away_team = visitor, match_outcome)

fit_rank_dyn <- btd_foot(
  data = italy_2020_2021_rank,
  dynamic_rank = TRUE,
  rank_measure = "median",
  iter = 1000,
  cores = 2,
  chains = 2)

plot_logStrength(fit_rank_dyn)

plot_logStrength(fit_rank_dyn, teams = c("AC Milan", "AS Roma", "Juventus", "Inter"))



## End(Not run)
```

pp_foot *Posterior predictive checks for football models*

Description

The function provides posterior predictive plots to check the adequacy of the Bayesian models as returned by the stan_foot function.

Usage

```
pp_foot(object, data, type = c("aggregated", "matches"), coverage = 0.95)
```

Arguments

object	An object of class stanfit or stanFoot as given by stan_foot function.
data	A data frame containing match data with columns: - periods: Time point of each observation (integer >= 1). - home_team: Home team's name (character string). - away_team: Away team's name (character string). - home_goals: Goals scored by the home team (integer >= 0).

- away_goals: Goals scored by the away team (integer >= 0).

type Type of plots, one among "aggregated" or "matches".

coverage Argument to specify the width $1 - \alpha$ of posterior probability intervals. Default is 0.95.

Value

Posterior predictive plots: when "aggregated" (default) is selected, the function returns a frequency plot for some pre-selected goal-difference values, along with their correspondent Bayesian p-values, computed as $Pr(y_rep \geq y)|y)$, where y_rep is a data replication from the posterior predictive distribution (more details in Gelman et al., 2013). Bayesian p-values very close to 0 or 1 could exhibit possible model misfits.

When "matches" is selected an ordered-frequency plot for all the goal-differences in the considered matches is provided, along with the empirical Bayesian coverage at level $1 - \alpha$.

Author(s)

Leonardo Egidi <legidi@units.it>

References

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian data analysis. CRC press.

Examples

```
## Not run:
library(dplyr)

data("italy")
italy_2000 <- italy %>%
 dplyr::select(Season, home, visitor, hgoal,vgoal) %>%
 dplyr::filter(Season=="2000")

colnames(italy_2000) <- c("periods", "home_team", "away_team", "home_goals", "away_goals")

fit <- stan_foot(italy_2000, "double_pois", iter = 200)

pp_foot(fit, italy_2000)


## End(Not run)
```

print.btdFoot *Print Method for btdFoot Objects*

Description

Provides detailed posterior summaries for the Bayesian Bradley-Terry-Davidson model parameters.

Usage

```
## S3 method for class 'btdFoot'
print(
  x,
  pars = NULL,
  teams = NULL,
  digits = 3,
  true_names = TRUE,
  display = c("both", "rankings", "parameters"),
  ...
)
```

Arguments

`x`	An object of class `btdFoot`.
`pars`	Optional character vector specifying parameters to include in the summary (e.g., `"logStrength"`, `"logTie"`, `"home"`, `"log_lik"`, and `"y_rep"`).
`teams`	Optional character vector specifying team names whose `logStrength` parameters should be displayed.
`digits`	Number of digits to use when printing numeric values. Default is `3`.
`true_names`	Logical value indicating whether to display team names in parameter summaries. Default is `TRUE`.
`display`	Character string specifying which parts of the output to display. Options are `"both"`, `"rankings"`, or `"parameters"`. Default is `"both"`.
`...`	Additional arguments passed.

Author(s)

Roberto Macrì Demartino <roberto.macridemartino@phd.unipd.it>

print.compareFoot *Print method for compareFoot objects*

Description

Provides a formatted output when printing objects of class compareFoot, displaying the predictive performance metrics and, if available, the confusion matrices for each model or probability matrix.

Usage

```
## S3 method for class 'compareFoot'
print(x, digits = 3, ...)
```

Arguments

x	An object of class compareFoot returned by compare_foot.
digits	Number of digits to use when printing numeric values for the metrics. Default is 3.
...	Additional arguments passed to print.

Author(s)

Roberto Macrì Demartino <roberto.macridemartino@phd.unipd.it>

print.stanFoot *Print Method for stanFoot Objects*

Description

Provides detailed posterior summaries for the Stan football model parameters.

Usage

```
## S3 method for class 'stanFoot'
print(x, pars = NULL, teams = NULL, digits = 3, true_names = TRUE, ...)
```

Arguments

x	An object of class stanFoot.
pars	Optional character vector specifying parameters to include in the summary. This can be specific parameter names (e.g., "att", "def", "att_raw", "def_raw", "home", "sigma_att", "sigma_def", "rho", and "beta"). If NULL, all parameters are included.
teams	Optional character vector specifying team names whose "att", "def", "att_raw", "def_raw" parameters should be displayed.

`digits`	Number of digits to use when printing numeric values. Default is `3`.
`true_names`	Logical value indicating whether to display team names in parameter summaries. Default is `TRUE`.
`...`	Additional arguments passed.

Author(s)

Roberto Macrì Demartino `<roberto.macridemartino@phd.unipd.it>`

`priors` *Football priors distributions and options*

Description

This prior specification is just a duplicate of some of the priors used by the **rstanarm** package.

These prior distributions can be passed to the `stan_foot` function, through the arguments `prior` and `prior_sd`. See the vignette *Prior Distributions for rstanarm Models* for further details (to view the priors used for an existing model see prior_summary). The default priors used in the **stan_foot** modeling function are intended to be *weakly informative* in that they provide moderate regularlization and help stabilize computation.

You can choose between: `normal`, `cauchy`, `laplace`, `student_t`.

Usage

```
normal(location = 0, scale = NULL, autoscale = TRUE)

student_t(df = 1, location = 0, scale = NULL, autoscale = TRUE)

cauchy(location = 0, scale = NULL, autoscale = TRUE)

laplace(location = 0, scale = NULL, autoscale = TRUE)
```

Arguments

`location`	Prior location. In most cases, this is the prior mean, but for cauchy (which is equivalent to `student_t` with `df=1`), the mean does not exist and `location` is the prior median. The default value is 0.
`scale`	Prior scale. The default depends on the family (see **Details**).
`autoscale`	A logical scalar, defaulting to `TRUE`.
`df`	Prior degrees of freedom. The default is 1 for `student_t`, in which case it is equivalent to `cauchy`.

Details

The details depend on the family of the prior being used:

Student t family: Family members:

- `normal(location, scale)`
- `student_t(df, location, scale)`
- `cauchy(location, scale)`

Each of these functions also takes an argument `autoscale`.

For the prior distribution for the intercept, `location`, `scale`, and `df` should be scalars. For the prior for the other coefficients they can either be vectors of length equal to the number of coefficients (not including the intercept), or they can be scalars, in which case they will be recycled to the appropriate length. As the degrees of freedom approaches infinity, the Student t distribution approaches the normal distribution and if the degrees of freedom are one, then the Student t distribution is the Cauchy distribution.

If `scale` is not specified it will default to 10 for the intercept and 2.5 for the other coefficients.

If the `autoscale` argument is `TRUE` (the default), then the scales will be further adjusted as described above in the documentation of the `autoscale` argument in the **Arguments** section.

Laplace family: Family members:

- `laplace(location, scale)`

Each of these functions also takes an argument `autoscale`.

The Laplace distribution is also known as the double-exponential distribution. It is a symmetric distribution with a sharp peak at its mean / median / mode and fairly long tails. This distribution can be motivated as a scale mixture of normal distributions and the remarks above about the normal distribution apply here as well.

Value

A named list to be used internally by the `stan_foot` model fitting function.

Author(s)

Leonardo Egidi `<legidi@units.it>`

References

Gelman, A., Jakulin, A., Pittau, M. G., and Su, Y. (2008). A weakly informative default prior distribution for logistic and other regression models. *Annals of Applied Statistics*. 2(4), 1360–1383.

See Also

The various vignettes for the **rstanarm** package also discuss and demonstrate the use of some of the supported prior distributions.

stan_foot *Fit football models with Stan*

Description

Stan football modelling for the most famous models: double Poisson, bivariate Poisson, Skellam, student t, diagonal-inflated bivariate Poisson and zero-inflated Skellam.

Usage

```
stan_foot(
  data,
  model,
  predict = 0,
  ranking,
  dynamic_type,
 prior_par = list(ability = normal(0, NULL), ability_sd = cauchy(0, 5), home = normal(0,
    5)),
  home_effect = TRUE,
  norm_method = "none",
  ranking_map = NULL,
  ...
)
```

Arguments

`data` A data frame containing match data with columns:

- `periods`: Time point of each observation (integer >= 1).
- `home_team`: Home team's name (character string).
- `away_team`: Away team's name (character string).
- `home_goals`: Goals scored by the home team (integer >= 0).
- `away_goals`: Goals scored by the away team (integer >= 0).

`model` A character string specifying the Stan model to fit. Options are:

- `"double_pois"`: Double Poisson model.
- `"biv_pois"`: Bivariate Poisson model.
- `"skellam"`: Skellam model.
- `"student_t"`: Student's t model.
- `"diag_infl_biv_pois"`: Diagonal-inflated bivariate Poisson model.
- `"zero_infl_skellam"`: Zero-inflated Skellam model.

`predict` An integer specifying the number of out-of-sample matches for prediction. If missing, the function fits the model to the entire dataset without making predictions.

`ranking` An optional `"btdFoot"` class element or a data frame containing ranking points for teams with the following columns:

- `periods`: Time periods corresponding to the rankings (integer >= 1).
- `team`: Team names matching those in `data` (character string).
- `rank_points`: Ranking points for each team (numeric).

`dynamic_type` A character string specifying the type of dynamics in the model. Options are:

- `"weekly"`: Weekly dynamic parameters.
- `"seasonal"`: Seasonal dynamic parameters.

`prior_par` A list specifying the prior distributions for the parameters of interest:

- `ability`: Prior distribution for team-specific abilities. Possible distributions are `normal`, `student_t`, `cauchy`, `laplace`. Default is `normal(0, NULL)`.
- `ability_sd`: Prior distribution for the team-specific standard deviations. See the `prior` argument for more details. Default is `cauchy(0, 5)`.
- `home`: Prior distribution for the home effect (`home`). Applicable only if `home_effect = TRUE`. Only normal priors are allowed. Default is `normal(0, 5)`.

See the **rstanarm** package for more details on specifying priors.

`home_effect` A logical value indicating the inclusion of a home effect in the model. (default is `TRUE`).

`norm_method` A character string specifying the method used to normalize team-specific ranking points. Options are:

- `"none"`: No normalization (default).
- `"standard"`: Standardization (mean 0, standard deviation 1).
- `"mad"`: Median Absolute Deviation normalization.
- `"min_max"`: Min-max scaling to [0,1].

`ranking_map` An optional vector mapping ranking periods to data periods. If not provided and the number of ranking periods matches the number of data periods, a direct mapping is assumed.

`...` Optional parameters passed to `stan` (e.g., `iter`, `chains`, `cores`, `control`).

Details

Let (y_n^H, y_n^A) denote the observed number of goals scored by the home and the away team in the n-th game, respectively. A general bivariate Poisson model allowing for goals' correlation (Karlis & Ntzoufras, 2003) is the following:

$$\begin{aligned}
Y_n^H, Y_n^A | \lambda_{1n}, \lambda_{2n}, \lambda_{3n} &\sim \mathsf{BivPoisson}(\lambda_{1n}, \lambda_{2n}, \lambda_{3n}) \\
\log(\lambda_{1n}) &= \mu + att_{h_n} + def_{a_n} \\
\log(\lambda_{2n}) &= att_{a_n} + def_{h_n} \\
\log(\lambda_{3n}) &= \beta_0,
\end{aligned}$$

where the case $\lambda_{3n} = 0$ reduces to the double Poisson model (Baio & Blangiardo, 2010). $\lambda_{1n}, \lambda_{2n}$ represent the scoring rates for the home and the away team, respectively, where: μ is the home effect; the parameters att_T and def_T represent the attack and the defence abilities, respectively, for each team T, $T = 1, \dots, N_T$; the nested indexes $h_n, a_n = 1, \dots, N_T$ denote the home and the away team playing in the n-th game, respectively. Attack/defence parameters are imposed a sum-to-zero constraint to achieve identifiability and assigned some weakly-informative prior distributions:

$$att_T \sim \mathcal{N}(\mu_{att}, \sigma_{att})$$
$$def_T \sim \mathcal{N}(\mu_{def}, \sigma_{def}),$$

with hyperparameters $\mu_{att}, \sigma_{att}, \mu_{def}, \sigma_{def}$.

Instead of using the marginal number of goals, another alternative is to modelling directly the score difference $(y^H_n - y^A_n)$. We can use the Poisson-difference distribution (or Skellam distribution) to model goal difference in the n-th match (Karlis & Ntzoufras, 2009):

$$y^H_n - y^A_n | \lambda_{1n}, \lambda_{2n} \sim PD(\lambda_{1n}, \lambda_{2n}),$$

and the scoring rates $\lambda_{1n}, \lambda_{2n}$ are unchanged with respect to the bivariate/double Poisson model. If we want to use a continue distribution, we can use a student t distribution with 7 degrees of freedom (Gelman, 2014):

$$y^H_n - y^A_n \sim t(7, ab_{h_n} - ab_{a(n)}, \sigma_y)$$
$$ab_t \sim \mathcal{N}(\mu + b \times prior\_score_t, sigma_{ab}),$$

where ab_t is the overall ability for the t-th team, whereas $prior\_score_t$ is a prior measure of team's strength (for instance a ranking).

These model rely on the assumption of static parameters. However, we could assume dynamics in the attach/defence abilities (Owen, 2011; Egidi et al., 2018) in terms of weeks or seasons through the argument `dynamic_type`. In such a framework, for a given number of times $1, \dots, \mathcal{T}$, the models above would be unchanged, but the priors for the abilities parameters at each time $\tau, \tau = 2, \dots, \mathcal{T}$, would be:

$$att_{T,\tau} \sim \mathcal{N}(att_{T,\tau-1}, \sigma_{att})$$
$$def_{T,\tau} \sim \mathcal{N}(def_{T,\tau-1}, \sigma_{def}),$$

whereas for $\tau = 1$ we have:

$$att_{T,1} \sim \mathcal{N}(\mu_{att}, \sigma_{att})$$
$$def_{T,1} \sim \mathcal{N}(\mu_{def}, \sigma_{def}).$$

Of course, the identifiability constraint must be imposed for each time τ.

The current version of the package allows for the fit of a diagonal-inflated bivariate Poisson and a zero-inflated Skellam model in the spirit of (Karlis & Ntzoufras, 2003) to better capture draw occurrences. See the vignette for further details.

Value

A list of class "stanFoot" containing:

- fit: The fitted stanfit object returned by stan.
- data: The input data.
- stan_data: The data list for Stan.
- stan_code: The Stan code of the underline model.
- stan_args: The optional parameters passed to (...).

Author(s)

Leonardo Egidi <legidi@units.it>, Roberto Macrì Demartino <roberto.macridemartino@phd.unipd.it>, and Vasilis Palaskas <vasilis.palaskas94@gmail.com>.

References

Baio, G. and Blangiardo, M. (2010). Bayesian hierarchical model for the prediction of football results. Journal of Applied Statistics 37(2), 253-264.

Egidi, L., Pauli, F., and Torelli, N. (2018). Combining historical data and bookmakers' odds in modelling football scores. Statistical Modelling, 18(5-6), 436-459.

Gelman, A. (2014). Stan goes to the World Cup. From "Statistical Modeling, Causal Inference, and Social Science" blog.

Karlis, D. and Ntzoufras, I. (2003). Analysis of sports data by using bivariate poisson models. Journal of the Royal Statistical Society: Series D (The Statistician) 52(3), 381-393.

Karlis, D. and Ntzoufras,I. (2009). Bayesian modelling of football outcomes: Using the Skellam's distribution for the goal difference. IMA Journal of Management Mathematics 20(2), 133-145.

Owen, A. (2011). Dynamic Bayesian forecasting models of football match outcomes with estimation of the evolution variance parameter. IMA Journal of Management Mathematics, 22(2), 99-113.

Examples

```
## Not run:
library(dplyr)



# Example usage with ranking
data("italy")
italy <- as_tibble(italy)
italy_2021 <- italy %>%
  select(Season, home, visitor, hgoal, vgoal) %>%
  filter(Season == "2021")


teams <- unique(italy_2021$home)
n_rows <- 20
```

```
# Create fake ranking
ranking <- data.frame(
  periods = rep(1, n_rows),
  team = sample(teams, n_rows, replace = FALSE),
  rank_points = sample(0:60, n_rows, replace = FALSE)
)

ranking <- ranking %>%
  arrange(periods, desc(rank_points))


colnames(italy_2021) <- c("periods", "home_team", "away_team", "home_goals", "away_goals")

fit_with_ranking <- stan_foot(
  data = italy_2021
  model = "diag_infl_biv_pois",
  ranking = ranking,
  home_effect = TRUE,
  prior_par = list(
    ability = student_t(4, 0, NULL),
    ability_sd = cauchy(0, 3),
    home = normal(1, 10)
  ),
  norm_method = "mad",
  iter = 1000,
  chains = 2,
  cores = 2,
  control = list(adapt_delta = 0.95, max_treedepth = 15)
)

# Print a summary of the model fit
print(fit_with_ranking, pars = c("att","def"))



### Use Italian Serie A from 2000 to 2002

data("italy")
italy <- as_tibble(italy)
italy_2000_2002<- italy %>%
 dplyr::select(Season, home, visitor, hgoal,vgoal) %>%
 dplyr::filter(Season=="2000" |  Season=="2001"| Season=="2002")

colnames(italy_2000_2002) <- c("periods", "home_team", "away_team", "home_goals", "away_goals")

### Fit Stan models
## no dynamics, no predictions

fit_1 <- stan_foot(data = italy_2000_2002,
                  model = "double_pois") # double poisson
print(fit_1, pars = c("home", "sigma_att",
                    "sigma_def"))
```

```
fit_2 <- stan_foot(data = italy_2000_2002,
                  model = "biv_pois")    # bivariate poisson
print(fit_2, pars = c("home", "rho",
                    "sigma_att", "sigma_def"))

fit_3 <- stan_foot(data = italy_2000_2002,
                  mode ="skellam")     # skellam
print(fit_3, pars = c("home", "sigma_att",
                    "sigma_def"))

fit_4 <- stan_foot(data = italy_2000_2002,
                  model = "student_t")   # student_t
print(fit_4, pars = c("beta"))

## seasonal dynamics, no prediction

fit_5 <- stan_foot(data = italy_2000_2002,
                  model = "double_pois",
                  dynamic_type = "seasonal") # double poisson
print(fit_5, pars = c("home", "sigma_att",
                    "sigma_def"))

## seasonal dynamics, prediction for the last season

fit_6 <- stan_foot(data = italy_2000_2002,
                  model = "double_pois",
                  dynamic_type = "seasonal",
                  predict = 170) # double poisson
print(fit_6, pars = c("home", "sigma_att",
                    "sigma_def"))

## other priors' options
# double poisson with
# student_t priors for teams abilities
# and laplace prior for the hyper sds

fit_p <- stan_foot(data = italy_2000_2002,
                   model = "double_pois",
                   prior_par = list(ability = student_t(4, 0, NULL),
                                    ability_sd = laplace(0,1),
                                    home = normal(1, 10)
                                    ))

print(fit_p,  pars = c("home", "sigma_att",
                    "sigma_def"))

## End(Not run)
```

Index

* **datasets**
    england, 7
    italy, 13

btd_foot, 2

cauchy (priors), 23
compare_foot, 5, *22*

data.frame, *10*

england, 7

foot_abilities, 8
foot_prob, 9
foot_rank, 11
foot_round_robin, 12

italy, 13

laplace (priors), 23
list, *8*, *10*

mle_foot, 14

normal (priors), 23

optim, *14*

plot_btdPosterior, 15
plot_logStrength, 18
pp_foot, 19
print.btdFoot, 21
print.compareFoot, 22
print.stanFoot, 22
priors, 23

stan, *3*, *26*, *28*
stan_foot, 25
stanfit, *10–12*, *19*
student_t (priors), 23