Package ‘EvidenceSynthesis’

May 8, 2023

Type Package
Title Synthesizing Causal Evidence in a Distributed Research Network
Version 0.5.0
Date 2023-05-08
Maintainer Martijn Schuemie <schuemie@ohdsi.org>
Description Routines for combining causal effect estimates and study diagnostics across multiple data sites in a distributed study, without sharing patient-level data. Allows for normal and non-normal approximations of the data-site likelihood of the effect parameter.
SystemRequirements Java (>= 8)
Depends survival, R (>= 3.5.0)
Imports dplyr, ggplot2, ggdist, gridExtra, meta, EmpiricalCalibration, rJava, BeastJar, Cyclops (>= 3.1.0), HDInterval, coda, rlang, methods
Suggests knitr, rmarkdown, testthat, sn
License Apache License 2.0
BugReports https://github.com/OHDSI/EvidenceSynthesis/issues
VignetteBuilder knitr
RoxygenNote 7.2.3
Encoding UTF-8
LazyData true
NeedsCompilation no
Author Martijn Schuemie [aut, cre],
  Marc A. Suchard [aut],
  Fan Bu [aut],
  Observational Health Data Science and Informatics [cph]
Repository CRAN
Date/Publication 2023-05-08 12:20:02 UTC
R topics documented:

approximateHierarchicalNormalPosterior ........................................ 2
approximateLikelihood ................................................................. 4
approximateSimplePosterior .......................................................... 5
biasCorrectionInference ................................................................. 6
computeBayesianMetaAnalysis ......................................................... 8
computeConfidenceInterval ............................................................ 9
computeFixedEffectMetaAnalysis .................................................... 10
createSimulationSettings ............................................................... 11
customFunction ........................................................................... 13
detectApproximationType ............................................................... 13
fitBiasDistribution ........................................................................ 14
ncLikelihoods ................................................................................ 15
ooiLikelihoods ................................................................................. 16
plotBiasCorrectionInference ......................................................... 17
plotBiasDistribution ................................................................. 18
plotCovariateBalances ................................................................. 19
plotEmpiricalNulls ................................................................. 20
plotLikelihoodFit ................................................................. 22
plotMcmcTrace ................................................................. 23
plotMetaAnalysisForest ............................................................. 24
plotPerDbMcmcTrace ............................................................. 26
plotPerDbPosterior ............................................................. 27
plotPosterior ................................................................. 28
plotPreparedPs ................................................................. 29
preparePsPlot ................................................................. 30
sequentialFitBiasDistribution .................................................. 32
simulatePopulations ................................................................. 33
skewNormal ................................................................. 33
supportsJava8 ................................................................. 34

Index ................................................................. 35

approximateHierarchicalNormalPosterior

Approximate Bayesian posterior for hierarchical Normal model

Description

Approximate a Bayesian posterior from a set of Cyclops likelihood profiles under a hierarchical normal model using the Markov chain Monte Carlo engine BEAST.
approximateHierarchicalNormalPosterior

Usage

approximateHierarchicalNormalPosterior(
  likelihoodProfiles,
  chainLength = 1000000,
  burnIn = 1e+05,
  subSampleFrequency = 100,
  effectPriorMean = 0,
  effectPriorSd = 0.5,
  nu0 = 1,
  sigma0 = 1,
  effectStartingValue = 0,
  precisionStartingValue = 1,
  seed = 1
)

Arguments

likelihoodProfiles   List of grid likelihoods profiled with Cyclops.
chainLength          Number of MCMC iterations.
burnIn               Number of MCMC iterations to consider as burn in.
subSampleFrequency   Subsample frequency for the MCMC.
effectPriorMean      Prior mean for global parameter
effectPriorSd        Prior standard deviation for the global parameter
nu0                  Prior "sample size" for precision (with precision ~ gamma(nu0/2, nu0*sigma0/2))
sigma0               Prior "variance" for precision (with precision ~ gamma(nu0/2, nu0*sigma0/2))
effectStartingValue  Initial value for global & local parameter
precisionStartingValue Initial value for the precision
seed                 Seed for the random number generator.

Value

A data frame with the point estimates and 95% credible intervals for the the global and local parameter, as well as the global precision. Attributes of the data frame contain the MCMC trace for diagnostics.

Examples

# TBD
approximateLikelihood  Approximate a likelihood function

Description

Approximate the likelihood function using a parametric (normal, skew-normal, or custom paramet-
ric), or grid approximation. The approximation does not reveal person-level information, and can
therefore be shared among data sites. When counts are low, a normal approximation might not be
appropriate.

Usage

approximateLikelihood(
  cyclopsFit,
  parameter = 1,
  approximation = "custom",
  bounds = c(log(0.1), log(10))
)

Arguments

  cyclopsFit    A model fitted using the Cyclops::fitCyclopsModel() function.
  parameter    The parameter in the cyclopsFit object to profile.
  approximation The type of approximation. Valid options are 'normal', 'skew normal', 'custom',
                  'grid', or 'adaptive grid'.
  bounds       The bounds on the effect size used to fit the approximation.

Value

A vector of parameters of the likelihood approximation.

See Also

computeConfidenceInterval, computeFixedEffectMetaAnalysis, computeBayesianMetaAnalysis

Examples

# Simulate some data for this example:
populations <- simulatePopulations()

cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
  data = populations[[1]],
  modelType = "cox"
)
cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
approximation <- approximateLikelihood(cyclopsFit, "x")
approximation
approximateSimplePosterior

# (Estimates in this example will vary due to the random simulation)

---

approximateSimplePosterior

Approximate simple Bayesian posterior

Description

Approximate a Bayesian posterior from a Cyclops likelihood profile and normal prior using the Markov chain Monte Carlo engine BEAST.

Usage

approximateSimplePosterior(
  likelihoodProfile,  # Named vector containing grid likelihood data from Cyclops.
  chainLength = 1100000,  # Number of MCMC iterations.
  burnIn = 1e+05,  # Number of MCMC iterations to consider as burn in.
  subSampleFrequency = 100,  # Subsample frequency for the MCMC.
  priorMean = 0,  # Prior mean for the regression parameter
  priorSd = 0.5,  # Prior standard deviation for the regression parameter
  startingValue = 0,  # Initial state for regression parameter
  seed = 1  # Seed for the random number generator.
)

Arguments

- likelihoodProfile
- chainLength
- burnIn
- subSampleFrequency
- priorMean
- priorSd
- startingValue
- seed

Value

A data frame with the point estimates and 95% credible intervals for the regression parameter. Attributes of the data frame contain the MCMC trace for diagnostics.
Examples

# Simulate some data for this example:
population <- simulatePopulations(createSimulationSettings(nSites = 1))[[1]]

# Fit a Cox regression at each data site, and approximate likelihood function:
cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
data = population,
modelType = "cox"
)
cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
likelihoodProfile <- approximateLikelihood(cyclopsFit, parameter = "x", approximation = "grid")

# Run MCMC
mcmcTraces <- approximateSimplePosterior(
  likelihoodProfile = likelihoodProfile,
priorMean = 0, priorSd = 100
)

# Report posterior expectation
mean(mcmcTraces$theta)

# (Estimates in this example will vary due to the random simulation)

biasCorrectionInference

Bias Correction with Inference

Description

Perform Bayesian posterior inference regarding an outcome of interest with bias correction using
negative control analysis. There is an option to not perform bias correction so that un-corrected
results can be obtained.

Usage

biasCorrectionInference(
  likelihoodProfiles,
  ncLikelihoodProfiles = NULL,
  biasDistributions = NULL,
priorMean = 0,
priorSd = 1,
numsamps = 10000,
thin = 10,
doCorrection = TRUE,
seed = 1,
...
Arguments

likelihoodProfiles
   A list of grid profile likelihoods for the outcome of interest.
ncLikelihoodProfiles
   Likelihood profiles for the negative control outcomes. Must be a list of lists of profile likelihoods; if there is only one analysis period, then this must be a length-1 list, with the first item as a list all outcome-wise profile likelihoods.
biasDistributions
   Pre-saved bias distribution(s), formatted as the output from fitBiasDistribution() or sequentialFitBiasDistribution(). If NULL, then ncLikelihoodProfiles must be provided.
priorMean
   Prior mean for the effect size (log rate ratio).
priorSd
   Prior standard deviation for the effect size (log rate ratio).
numsamps
   Total number of MCMC samples needed.
thin
   Thinning frequency: how many iterations before another sample is obtained?
doCorrection
   Whether or not to perform bias correction; default: TRUE.
seed
   Seed for the random number generator.
...
   Arguments to be passed to sequentialFitBiasDistribution() to fit bias distributions if biasDistributions is NULL.

Value

A dataframe with five columns, including posterior median and mean of log RR effect size estimates, 95% credible intervals (ci95Lb and ci95Ub), posterior probability that log RR > 0 (p1), and the period or group ID (Id).

It is accompanied by the following attributes:

- samplesCorrected: all MCMC samples for the bias corrected log RR effect size estimate.
- samplesRaw: all MCMC samples for log RR effect size estimate, without bias correction.
- biasDistributions: the learned empirical bias distribution from negative control analysis.
- summaryRaw: a summary dataframe (same format as in the main result) without bias correction.
- corrected: a logical flag indicating if bias correction has been performed; = TRUE if doCorrection = TRUE.

See Also

approximateSimplePosterior, fitBiasDistribution

Examples

# load example data
data("ncLikelihoods")
data("ooiLikelihoods")
# perform sequential analysis with bias correction, using the t model
# NOT RUN
# bbcResults = biasCorrectionInference(ooiLikelihoods, 
#   ncLikelihoodProfiles = ncLikelihoods, 
#   robust = TRUE, 
#   seed = 42)

# check out analysis summary
# bbcResults

---

**computeBayesianMetaAnalysis**

*Compute a Bayesian random-effects meta-analysis*

**Description**

Compute a Bayesian meta-analysis using the Markov chain Monte Carlo (MCMC) engine BEAST. A normal and half-normal prior are used for the mu and tau parameters, respectively, with standard deviations as defined by the `priorSd` argument.

**Usage**

```r
computeBayesianMetaAnalysis(
  data,
  chainLength = 1100000,
  burnIn = 1e+05,
  subSampleFrequency = 100,
  priorSd = c(2, 0.5),
  alpha = 0.05,
  robust = FALSE,
  df = 4,
  seed = 1
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data</code></td>
<td>A data frame containing either normal, skew-normal, custom parametric, or grid likelihood data, with one row per database.</td>
</tr>
<tr>
<td><code>chainLength</code></td>
<td>Number of MCMC iterations.</td>
</tr>
<tr>
<td><code>burnIn</code></td>
<td>Number of MCMC iterations to consider as burn in.</td>
</tr>
<tr>
<td><code>subSampleFrequency</code></td>
<td>Subsample frequency for the MCMC.</td>
</tr>
<tr>
<td><code>priorSd</code></td>
<td>A two-dimensional vector with the standard deviation of the prior for mu and tau, respectively.</td>
</tr>
<tr>
<td><code>alpha</code></td>
<td>The alpha (expected type I error) used for the credible intervals.</td>
</tr>
</tbody>
</table>
computeConfidenceInterval

robust Whether or not to use a t-distribution model; default: FALSE.
df Degrees of freedom for the t-model, only used if robust is TRUE.
seed The seed for the random number generator.

Value

A data frame with the point estimates and 95% credible intervals for the mu and tau parameters (the mean and standard deviation of the distribution from which the per-site effect sizes are drawn). Attributes of the data frame contain the MCMC trace and the detected approximation type.

See Also

approximatelyLikelihood, computeFixedEffectMetaAnalysis

Examples

# Simulate some data for this example:
populations <- simulatePopulations()

# Fit a Cox regression at each data site, and approximate likelihood function:
fitModelInDatabase <- function(population) {
  cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
                                           data = population,
                                           modelType = "cox"
  )
  cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
  approximation <- approximateLikelihood(cyclopsFit, parameter = "x", approximation = "custom")
  return(approximation)
}
approximations <- lapply(populations, fitModelInDatabase)
approximations <- do.call("rbind", approximations)

# At study coordinating center, perform meta-analysis using per-site approximations:
estimate <- computeBayesianMetaAnalysis(approximations)
estimate

# (Estimates in this example will vary due to the random simulation)
**computeFixedEffectMetaAnalysis**

*Description*

Compute a fixed-effect meta-analysis using a choice of various likelihood approximations.

**Usage**

```r
computeFixedEffectMetaAnalysis(data, alpha = 0.05)
```

**Arguments**

- `data`: A data frame containing either normal, skew-normal, custom parametric, or grid likelihood data. One row per database.
- `alpha`: The alpha (expected type I error) used for the confidence intervals.
createSimulationSettings

Value
The meta-analytic estimate, expressed as the point estimate hazard ratio (\( r_r \)), its 95 percent confidence interval (lb, ub), as well as the log of the point estimate (logRr), and the standard error (seLogRr).

See Also
approximateLikelihood, computeBayesianMetaAnalysis

Examples
# Simulate some data for this example:
populations <- simulatePopulations()

# Fit a Cox regression at each data site, and approximate likelihood function:
fitModelInDatabase <- function(population) {
  cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
                                           data = population,
                                           modelType = "cox",
                                           approximation = "custom")
  cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
  approximation <- approximateLikelihood(cyclopsFit, parameter = "x", approximation = "custom")
  return(approximation)
}
approximations <- lapply(populations, fitModelInDatabase)
approximations <- do.call("rbind", approximations)

# At study coordinating center, perform meta-analysis using per-site approximations:
computeFixedEffectMetaAnalysis(approximations)

# (Estimates in this example will vary due to the random simulation)
createSimulationSettings

nStrata = 10,
minBackgroundHazard = 2e-07,
maxBackgroundHazard = 2e-05,
hazardRatio = 2,
randomEffectSd = 0
}

Arguments

nSites Number of database sites to simulate.
n Number of subjects per site. Either a single number, or a vector of length nSites.
treatedFraction Fraction of subjects that is treated. Either a single number, or a vector of length nSites.
nStrata Number of strata per site. Either a single number, or a vector of length nSites.
minBackgroundHazard Minimum background hazard. Either a single number, or a vector of length nSites.
maxBackgroundHazard Maximum background hazard. Either a single number, or a vector of length nSites.
hazardRatio Hazard ratio.
randomEffectSd Standard deviation of the log(hazardRatio). Fixed effect if equal to 0.

Value

An object of type simulationSettings, to be used in the simulatePopulations() function.

See Also

simulatePopulations

Examples

settings <- createSimulationSettings(nSites = 1, hazardRatio = 2)
populations <- simulatePopulations(settings)

# Fit a Cox regression for the simulated data site:
cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
data = populations[[1]],
modelType = "cox"
)
cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
coef(cyclopsFit)

# (Estimates in this example will vary due to the random simulation)
customFunction

A custom function to approximate a log likelihood function

Description

A custom function to approximate a log likelihood function

Usage

customFunction(x, mu, sigma, gamma)

Arguments

x The log(hazard ratio) for which to approximate the log likelihood.
mu The position parameter.
sigma The scale parameter.
gamma The skew parameter.

Details

A custom parametric function designed to approximate the shape of the Cox log likelihood function. When gamma = 0 this function is the normal distribution.

Value

The approximate log likelihood for the given x.

Examples

customFunction(x = 0:3, mu = 0, sigma = 1, gamma = 0)

detectApproximationType

Detect the type of likelihood approximation based on the data format

Description

Detect the type of likelihood approximation based on the data format

Usage

detectApproximationType(data, verbose = TRUE)
Arguments

- **data**: The approximation data. Can be a single approximation, or approximations from multiple sites.
- **verbose**: Should the detected type be communicated to the user?

Value

A character vector with one of the following values: "normal", "custom", "skew normal", "pooled", "grid", or "adaptive grid".

Examples

detectApproximationType(data.frame(logRr = 1, seLogRr = 0.1))

---

**fitBiasDistribution**  
*Fit Bias Distribution*

Description

Learn an empirical distribution on estimation bias by simultaneously analyzing a large set of negative control outcomes by a Bayesian hierarchical model through MCMC. Analysis is based on a list of extracted likelihood profiles.

Usage

```r
fitBiasDistribution(
  likelihoodProfiles,
  priorSds = c(2, 0.5),
  numsamps = 10000,
  thin = 10,
  minNCs = 5,
  robust = FALSE,
  df = 4,
  seed = 1
)
```

Arguments

- **likelihoodProfiles**: A list of grid profile likelihoods regarding negative controls.
- **priorSds**: A two-dimensional vector with the standard deviation of the prior for the average bias and the sd/scale parameter, respectively.
- **numsamps**: Total number of MCMC samples needed.
- **thin**: Thinning frequency: how many iterations before another sample is obtained?
ncLikelihoods

Minimum number of negative controls needed to fit a bias distribution; default (also recommended): 5.

Whether or not to use a t-distribution model; default: FALSE.

Degrees of freedom for the t-model, only used if robust is TRUE.

Seed for the random number generator.

Value

A dataframe with three columns and numsamps number of rows. Column mean includes MCMC samples for the average bias, scale for the sd/scale parameter, and bias for predictive samples of the bias.

See Also

computeBayesianMetaAnalysis

Examples

# load example data
data("ncLikelihoods")

# fit a bias distributions by analyzing a set of negative control outcomes
# for example, for the 5th analysis period, and using the t model
# NOT RUN
# biasDistribution = fitBiasDistribution(ncLikelihoods[[5]], robust = TRUE)

ncLikelihoods

Example profile likelihoods for negative control outcomes

Description

A list that contain profile likelihoods a large set of negative control outcomes. They are extracted from a real-world observational healthcare database, with the likelihoods profiled using adaptive grids using the Cyclops package.

Usage

ncLikelihoods

Format

An object of class list containing 12 lists, where each list includes several dataframes ith column point and value for adaptive grid profile likelihoods.

References

Examples

```r
data("ncLikelihoods")
nclIkEx <- ncLikelihoods["5"][[1]]
plot(value ~ point, data = nclIkEx)
```

---

**ooiLikelihoods**  
*Example profile likelihoods for a synthetic outcome of interest*

**Description**

A list that contain profile likelihoods for a synthetic outcome of interest. They are extracted from a real-world observational healthcare database, with the likelihoods profiled using adaptive grids using the Cyclops package.

**Usage**

`ooiLikelihoods`

**Format**

An objects of class `list`; the list contains 12 lists, where each list includes several dataframes with column `point` and `value` for adaptive grid profile likelihoods.

**References**


**Examples**

```r
data("ooiLikelihoods")
ooiLikEx <- ooiLikelihoods["5" ][[1]]
plot(value ~ point, data = ooiLikEx)
```
**Description**
Plot bias correction inference

**Usage**

```r
plotBiasCorrectionInference(
  bbcResult,
  type = "raw",
  ids = bbcResult$Id,
  limits = c(-3, 3),
  logScale = FALSE,
  numericId = TRUE,
  fileName = NULL
)
```

**Arguments**

- `bbcResult`: A (sequential) analysis object generated by the `biasCorrectionInference()` function.
- `type`: The type of plot. Must be one of c("corrected", "raw", "compare").
- `ids`: IDs of the periods/groups to plot result for; default is all IDs.
- `limits`: The limits on log RR for plotting.
- `logScale`: Whether or not to show bias in log-RR; default FALSE (shown in RR).
- `numericId`: Whether or not to treat Id as a numeric variable; default: TRUE.
- `fileName`: Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave` in the ggplot2 package for supported file formats.

**Details**
Plot empirical bias distributions learned from analyzing negative controls.

**Value**
A ggplot object. Use the `ggplot2::ggsave` function to save to file.

**See Also**

- `biasCorrectionInference`
Examples

# Perform sequential analysis using Bayesian bias correction for this example:
data("ncLikelihoods")
data("ooiLikelihoods")
# NOT RUN
# bbcSequential = biasCorrectionInference(ooiLikelihoods, ncLikelihoodProfiles = ncLikelihoods)

# Plot it
# NOT RUN
# plotBiasCorrectionInference(bbcSequential, type = "corrected")

Description

Plot bias distributions

Usage

plotBiasDistribution(
  biasDist,
  limits = c(-2, 2),
  logScale = FALSE,
  numericId = TRUE,
  fileName = NULL
)

Arguments

biasDist A bias distribution object generated by the fitBiasDistribution() or sequentialFitBiasDistribution function.
limits The lower and upper limits in log-RR to plot.
logScale Whether or not to show bias in log-RR; default FALSE (shown in RR).
numericId (For sequential or group case only) whether or not to treat Id as a numeric variable; default: TRUE.
fileName Name of the file where the plot should be saved, for example 'plot.png'. See the function ggplot2::ggsave in the ggplot2 package for supported file formats.

Details

Plot empirical bias distributions learned from analyzing negative controls.

Value

A ggplot object. Use the ggplot2::ggsave function to save to file.
plotCovariateBalances

See Also

fitBiasDistribution, sequentialFitBiasDistribution

Examples

# Fit a bias distribution for this example:
data("ncLikelihoods")
# NOT RUN
# singleBiasDist = fitBiasDistribution(ncLikelihoods[[5]], seed = 1)

# Plot it
# NOT RUN
# plotBiasDistribution(singleBiasDist)

plotCovariateBalances  Plot covariate balances

Description

Plots the covariate balance before and after matching for multiple data sources.

Usage

plotCovariateBalances(
  balances,
  labels,
  threshold = 0,
  beforeLabel = "Before matching",
  afterLabel = "After matching",
  fileName = NULL
)

Arguments

balances  A list of covariate balance objects as created using the computeCovariateBalance() function in the CohortMethod package. Each balance object is expected to be a data frame with at least these two columns: beforeMatchingStdDiff and afterMatchingStdDiff.

labels  A vector containing the labels for the various sources.

threshold  Show a threshold value for the standardized difference.

beforeLabel  Label for before matching / stratification / trimming.

afterLabel  Label for after matching / stratification / trimming.

fileName  Name of the file where the plot should be saved, for example 'plot.png'. See the function ggplot2::ggsave for supported file formats.
Details

Creates a plot showing the covariate balance before and after matching. Balance distributions are displayed as box plots combined with scatterplots.

Value

A Ggplot object. Use the `ggplot2::ggsave`.

Examples

```r
# Some example data:
balance1 <- data.frame(
  beforeMatchingStdDiff = rnorm(1000, 0.1, 0.1),
  afterMatchingStdDiff = rnorm(1000, 0, 0.01)
)
balance2 <- data.frame(
  beforeMatchingStdDiff = rnorm(1000, 0.2, 0.1),
  afterMatchingStdDiff = rnorm(1000, 0, 0.05)
)
balance3 <- data.frame(
  beforeMatchingStdDiff = rnorm(1000, 0, 0.1),
  afterMatchingStdDiff = rnorm(1000, 0, 0.03)
)
plotCovariateBalances(
  balances = list(balance1, balance2, balance3),
  labels = c("Site A", "Site B", "Site C")
)
```

---

**plotEmpiricalNulls**  
*Plot empirical null distributions*

Description

Plot the empirical null distribution for multiple data sources.

Usage

```r
plotEmpiricalNulls(
  logRr,  
  seLogRr,  
  labels,  
  xLabel = "Relative risk",  
  limits = c(0.1, 10),  
  showCis = TRUE,  
  fileName = NULL  
)
```
Arguments

- **logRr**: A numeric vector of effect estimates for the negative controls on the log scale.
- **seLogRr**: The standard error of the log of the effect estimates. Hint: often the standard error = \(\frac{\log(\text{lower bound 95 percent confidence interval}) - \log(\text{effect estimate})}{\text{qnorm}(0.025)}\).
- **labels**: A vector containing the labels for the various sources. Should be of equal length as `logRr` and `seLogRr`.
- **xLabel**: The label on the x-axis: the name of the effect estimate.
- **limits**: The limits of the effect size axis.
- **showCis**: Show the 95 percent confidence intervals on the null distribution and distribution parameter estimates?
- **fileName**: Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave()` for supported file formats.

Details

Creates a plot showing the empirical null distributions. Distributions are shown as mean plus minus one standard deviation, as well as a distribution plot.

Value

A Ggplot object. Use the `ggplot2::ggsave()` function to save to file.

See Also

- `EmpiricalCalibration::fitNull`, `EmpiricalCalibration::fitMcmcNull`

Examples

```r
# Some example data:
site1 <- EmpiricalCalibration::simulateControls(n = 50, mean = 0, sd = 0.1, trueLogRr = 0)
site1$label <- "Site 1"
site2 <- EmpiricalCalibration::simulateControls(n = 50, mean = 0.1, sd = 0.2, trueLogRr = 0)
site2$label <- "Site 2"
site3 <- EmpiricalCalibration::simulateControls(n = 50, mean = 0.15, sd = 0.25, trueLogRr = 0)
site3$label <- "Site 3"
sites <- rbind(site1, site2, site3)
plotEmpiricalNulls(logRr = sites$logRr, seLogRr = sites$seLogRr, labels = sites$label)
```
plotLikelihoodFit

Plot the likelihood approximation

Description
Plot the likelihood approximation

Usage
plotLikelihoodFit(
  approximation,
  cyclopsFit,
  parameter = "x",
  logScale = TRUE,
  xLabel = "Hazard Ratio",
  limits = c(0.1, 10),
  fileName = NULL
)

Arguments
approximation An approximation of the likelihood function as fitted using the approximateLikelihood() function.
cyclopsFit A model fitted using the Cyclops::fitCyclopsModel() function.
parameter The parameter in the cyclopsFit object to profile.
logScale Show the y-axis on the log scale?
xLabel The title of the x-axis.
limits The limits on the x-axis.
fileName Name of the file where the plot should be saved, for example 'plot.png'. See the function ggplot2::ggsave in the ggplot2 package for supported file formats.

Details
Plots the (log) likelihood and the approximation of the likelihood. Allows for reviewing the approximation.

Value
A Ggplot object. Use the ggplot2::ggsave function to save to file.
Examples

# Simulate a single database population:
population <- simulatePopulations(createSimulationSettings(nSites = 1))[[1]]

# Approximate the likelihood:
cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
   data = population,
   modelType = "cox"
)  
cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
approximation <- approximateLikelihood(cyclopsFit, parameter = "x", approximation = "custom")
plotLikelihoodFit(approximation, cyclopsFit, parameter = "x")

Description

Plot MCMC trace

Usage

plotMcmcTrace(  
estimate,  
showEstimate = TRUE,
dataCutoff = 0.01,
fileName = NULL
)

Arguments

estimate An object as generated using the computeBayesianMetaAnalysis() function.
showEstimate Show the parameter estimates (mode) and 95 percent confidence intervals?
dataCutoff This fraction of the data at both tails will be removed.
fileName Name of the file where the plot should be saved, for example 'plot.png'. See the function ggplot2::ggsave in the ggplot2 package for supported file formats.

Details

Plot the samples of the posterior distribution of the mu and tau parameters. Samples are taken using Markov-chain Monte Carlo (MCMC).

Value

A Ggplot object. Use the ggplot2::ggsave function to save to file.
See Also

computeBayesianMetaAnalysis

Examples

# Simulate some data for this example:
populations <- simulatePopulations()

# Fit a Cox regression at each data site, and approximate likelihood function:
fitModelInDatabase <- function(population) {
  cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
                                          data = population,
                                          modelType = "cox"
  }
  cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
  approximation <- approximateLikelihood(cyclopsFit, parameter = "x", approximation = "custom")
  return(approximation)
}

approximations <- lapply(populations, fitModelInDatabase)
approximations <- do.call("rbind", approximations)

# At study coordinating center, perform meta-analysis using per-site approximations:
estimate <- computeBayesianMetaAnalysis(approximations)
plotMcmcTrace(estimate)

plotMetaAnalysisForest

Create a forest plot

Description

Creates a forest plot of effect size estimates, including the summary estimate.

Usage

plotMetaAnalysisForest(
  data, 
  labels, 
  estimate, 
  xLabel = "Relative risk", 
  summaryLabel = "Summary", 
  limits = c(0.1, 10), 
  alpha = 0.05, 
  showLikelihood = TRUE, 
  fileName = NULL 
)
Arguments

data A data frame containing either normal, skew-normal, custom parametric, or grid likelihood data. One row per database.
labels A vector of labels for the data sources.
estimate The meta-analytic estimate as created using either ['computeFixedEffectMetaAnalysis()'] or ['computeBayesianMetaAnalysis()'] function.
xLabel The label on the x-axis: the name of the effect estimate.
summaryLabel The label for the meta-analytic estimate.
limits The limits of the effect size axis.
alpha The alpha (expected type I error).
showLikelihood Show the likelihood curve for each estimate?
fileName Name of the file where the plot should be saved, for example 'plot.png'. See the function ggplot2::ggsave ifor supported file formats.

Details

Creates a forest plot of effect size estimates, including a meta-analysis estimate.

Value

A Ggplot object. Use the ggplot2::ggsave function to save to file.

Examples

# Simulate some data for this example:
populations <- simulatePopulations()
labels <- paste("Data site", LETTERS[1:length(populations)])

# Fit a Cox regression at each data site, and approximate likelihood function:
fitModelInDatabase <- function(population) {
  cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
                                           data = population,
                                           modelType = "cox"
  )
  cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
  approximation <- approximateLikelihood(cyclopsFit, parameter = "x", approximation = "custom")
  return(approximation)
}
approximations <- lapply(populations, fitModelInDatabase)
approximations <- do.call("rbind", approximations)

# At study coordinating center, perform meta-analysis using per-site approximations:
estimate <- computeBayesianMetaAnalysis(approximations)
plotMetaAnalysisForest(approximations, labels, estimate)

# (Estimates in this example will vary due to the random simulation)
Description

Plot MCMC trace for individual databases

Usage

plotPerDbMcmcTrace(
  estimate,
  showEstimate = TRUE,
  dataCutoff = 0.01,
  fileName = NULL
)

Arguments

- `estimate`: An object as generated using the `computeBayesianMetaAnalysis()` function.
- `showEstimate`: Show the parameter estimates (mode) and 95 percent confidence intervals?
- `dataCutoff`: This fraction of the data at both tails will be removed.
- `fileName`: Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave` in the ggplot2 package for supported file formats.

Details

Plot the samples of the posterior distribution of the theta parameter (the estimated log hazard ratio) at each site. Samples are taken using Markov-chain Monte Carlo (MCMC).

Value

A `ggplot` object. Use the `ggplot2::ggsave` function to save to file.

See Also

`computeBayesianMetaAnalysis`

Examples

```r
# Simulate some data for this example:
populations <- simulatePopulations()

# Fit a Cox regression at each data site, and approximate likelihood function:
fitModelInDatabase <- function(population) {
  cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
                                          data = population,
                                          modelType = "cox"
}
```
cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
approximation <- approximateLikelihood(cyclopsFit, parameter = "x", approximation = "custom")
return(approximation)
}
approximations <- lapply(populations, fitModelInDatabase)
approximations <- do.call("rbind", approximations)

# At study coordinating center, perform meta-analysis using per-site approximations:
estimate <- computeBayesianMetaAnalysis(approximations)
plotPerDbMcmcTrace(estimate)

---

**plotPerDbPosterior**

Plot posterior density per database

### Description

Plot posterior density per database

### Usage

```r
plotPerDbPosterior(
  estimate,
  showEstimate = TRUE,
  dataCutoff = 0.01,
  fileName = NULL
)
```

### Arguments

- **estimate**: An object as generated using the `computeBayesianMetaAnalysis()` function.
- **showEstimate**: Show the parameter estimates (mode) and 95 percent confidence intervals?
- **dataCutoff**: This fraction of the data at both tails will be removed.
- **fileName**: Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave` in the ggplot2 package for supported file formats.

### Details

Plot the density of the posterior distribution of the theta parameter (the estimated log hazard ratio) at each site.

### Value

A Ggplot object. Use the `ggplot2::ggsave` function to save to file.
Examples

# Simulate some data for this example:
populations <- simulatePopulations()

# Fit a Cox regression at each data site, and approximate likelihood function:
fitModelInDatabase <- function(population) {
  cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
                                         data = population,
                                         modelType = "cox"
  }
  cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
  approximation <- approximateLikelihood(cyclopsFit, parameter = "x", approximation = "custom")
  return(approximation)
}
approximations <- lapply(populations, fitModelInDatabase)
approximations <- do.call("rbind", approximations)

# At study coordinating center, perform meta-analysis using per-site approximations:
estimate <- computeBayesianMetaAnalysis(approximations)
plotPerDbPosterior(estimate)

plotPosterior

Plot posterior density

Description

Plot posterior density

Usage

plotPosterior(
  estimate,
  showEstimate = TRUE,
  dataCutoff = 0.01,
  fileName = NULL
)

Arguments

estimate      An object as generated using the computeBayesianMetaAnalysis() function.
showEstimate  Show the parameter estimates (mode) and 95 percent confidence intervals?
dataCutoff     This fraction of the data at both tails will be removed.
fileName      Name of the file where the plot should be saved, for example 'plot.png'. See the function ggplot2::ggsave in the ggplot2 package for supported file formats.
Details

Plot the density of the posterior distribution of the \( \mu \) and \( \tau \) parameters.

Value

A Ggplot object. Use the `ggplot2::ggsave` function to save to file.

See Also

`computeBayesianMetaAnalysis`

Examples

```r
# Simulate some data for this example:
populations <- simulatePopulations()

# Fit a Cox regression at each data site, and approximate likelihood function:
fitModelInDatabase <- function(population) {
  cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
                                           data = population,
                                           modelType = "cox"
  }
  cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
  approximation <- approximateLikelihood(cyclopsFit, parameter = "x", approximation = "custom")
  return(approximation)
}
approximations <- lapply(populations, fitModelInDatabase)
approximations <- do.call("rbind", approximations)

# At study coordinating center, perform meta-analysis using per-site approximations:
estimate <- computeBayesianMetaAnalysis(approximations)
plotPosterior(estimate)
```

---

**plotPreparedPs**

Plot the propensity score distribution

**Description**

Plot the propensity score distribution

**Usage**

```r
plotPreparedPs(
  preparedPsPlots,
  labels,
  treatmentLabel = "Target",
  comparatorLabel = "Comparator",
  fileName = NULL
)
```
preparePsPlot

Prepare to plot the propensity score distribution

Arguments

preparedPsPlots
- list of prepared propensity score data as created by the preparePsPlot() function.

labels
- A vector containing the labels for the various sources.

treatmentLabel
- A label to use for the treated cohort.

comparatorLabel
- A label to use for the comparator cohort.

fileName
- Name of the file where the plot should be saved, for example 'plot.png'. See the function ggplot2::ggsave for supported file formats.

Value

A ggplot object. Use the ggplot2::ggsave function to save to file in a different format.

See Also

preparePsPlot

Examples

# Simulate some data for this example:
treatment <- rep(0:1, each = 100)
propensityScore <- c(rnorm(100, mean = 0.4, sd = 0.25), rnorm(100, mean = 0.6, sd = 0.25))
data <- data.frame(treatment = treatment, propensityScore = propensityScore)
data <- data[data$propensityScore > 0 & data$propensityScore < 1, ]
preparedPlot <- preparePsPlot(data)

# Just reusing the same data three times for demonstration purposes:
preparedPsPlots <- list(preparedPlot, preparedPlot, preparedPlot)
labels <- c("Data site A", "Data site B", "Data site C")
plotPreparedPs(preparedPsPlots, labels)
preparePsPlot

**Arguments**

- **data**
  A data frame with at least the two columns described below
- **unfilteredData**
  To be used when computing preference scores on data from which subjects have already been removed, e.g., through trimming and/or matching. This data frame should have the same structure as `data`.
- **scale**
  The scale of the graph. Two scales are supported: `scale = 'propensity'` or `scale = 'preference'`. The preference score scale is defined by Walker et al. (2013).

**Details**

The data frame should have at least the following two columns:

- **treatment** (integer): Column indicating whether the person is in the treated (1) or comparator (0) group.
- **propensityScore** (numeric): Propensity score.

**Value**

A data frame describing the propensity score (or preference score) distribution at 100 equally-spaced points.

**References**


**See Also**

plotPreparedPs

**Examples**

```r
# Simulate some data for this example:
treatment <- rep(0:1, each = 100)
propensityScore <- c(rnorm(100, mean = 0.4, sd = 0.25), rnorm(100, mean = 0.6, sd = 0.25))
data <- data.frame(treatment = treatment, propensityScore = propensityScore)
data <- data[data$propensityScore > 0 & data$propensityScore < 1, ]

preparedPlot <- preparePsPlot(data)
```
sequentialFitBiasDistribution

Fit Bias Distribution Sequentially or in Groups

Description

Learn empirical bias distributions sequentially or in groups; for each sequential step or analysis group, bias distributions is learned by by simultaneously analyzing a large set of negative control outcomes by a Bayesian hierarchical model through MCMC.

Usage

sequentialFitBiasDistribution(LikelihoodProfileList, ...)

Arguments

LikelihoodProfileList

A list of lists, each of which is a set of grid profile likelihoods regarding negative controls, indexed by analysis period ID for sequential analyses or group ID for group analyses.

... Arguments passed to the fitBiasDistribution() function.

Value

A (long) dataframe with four columns. Column mean includes MCMC samples for the average bias, scale for the sd/scale parameter, bias for predictive samples of the bias, and Id for the period ID or group ID.

See Also

fitBiasDistribution, computeBayesianMetaAnalysis

Examples

# load example data
data("ncLikelihoods")

# fit bias distributions over analysis periods
# NOT RUN
# biasDistributions = sequentialFitBiasDistribution(ncLikelihoods, seed = 42)
simulatePopulations

Simulate survival data for multiple databases

Description

Simulate survival data for multiple databases

Usage

simulatePopulations(settings = createSimulationSettings())

Arguments

settings  
An object of type simulationSettings, created by the createSimulationSettings() function.

Value

A object of class simulation, which is a list of populations, each a data frame with columns rowId, stratumId, x, time, and y.

Examples

settings <- createSimulationSettings(nSites = 1, hazardRatio = 2)
populations <- simulatePopulations(settings)

# Fit a Cox regression for the simulated data site:
cyclopsData <- Cyclops::createCyclopsData(Surv(time, y) ~ x + strata(stratumId),
   data = populations[[1]],
   modelType = "cox"
)
cyclopsFit <- Cyclops::fitCyclopsModel(cyclopsData)
coef(cyclopsFit)

# (Estimates in this example will vary due to the random simulation)

skewNormal

The skew normal function to approximate a log likelihood function

Description

The skew normal function to approximate a log likelihood function

Usage

skewNormal(x, mu, sigma, alpha)
**Arguments**

- \( x \) The log(hazard ratio) for which to approximate the log likelihood.
- \( \mu \) The position parameter.
- \( \sigma \) The scale parameter.
- \( \alpha \) The skew parameter.

**Details**

The skew normal function. When \( \alpha = 0 \) this function is the normal distribution.

**Value**

The approximate log likelihood for the given \( x \).

**References**


**Examples**

\[
\text{skewNormal}(x = 0:3, \mu = 0, \sigma = 1, \alpha = 0)
\]

---

**supportsJava8**

*Determine if Java virtual machine supports Java*

**Description**

Tests Java virtual machine (JVM) java.version system property to check if version \( \geq 8 \).

**Usage**

\`
\text{supportsJava8()}
\`

**Value**

Returns TRUE if JVM supports Java \( \geq 8 \).

**Examples**

\`
\text{supportsJava8()}
\`

---
Index

* datasets
  - nclikelihoods, 15
  - ooilikelihoods, 16

approximateHierarchicalNormalPosterior, 2
approximateLikelihood, 4, 9, 11
approximate Likelihood(), 10, 22
approximateSimplePosterior, 5, 7
biasCorrectionInference, 6, 17
biasCorrectionInference(), 17
computeBayesianMetaAnalysis, 4, 8, 11, 15, 24, 26, 29, 32
computeBayesianMetaAnalysis(), 23, 26–28
computeConfidenceInterval, 4, 9
computeFixedEffectMetaAnalysis, 4, 9, 10
createSimulationSettings, 11
createSimulationSettings(), 33
customFunction, 13
cyclops::fitCyclopsModel(), 4, 22
detectApproximationType, 13
empiricalCalibration::fitMcmcNull, 21
empiricalCalibration::fitNull, 21
fitBiasDistribution, 7, 14, 19, 32
fitBiasDistribution(), 7, 18, 32
ggplot2::ggsave, 17–20, 22, 23, 25–30
ggplot2::ggsave(), 21
ncLikelihoods, 15
ooiLikelihoods, 16
plotBiasCorrectionInference, 17
plotBiasDistribution, 18
plotCovariateBalances, 19
plotEmpiricalNulls, 20
plotLikelihoodFit, 22
plotMcmcTrace, 23
plotMetaAnalysisForest, 24
plotPerDbMcmcTrace, 26
plotPerDbPosterior, 27
plotPosterior, 28
plotPreparedPs, 29, 31
preparePsPlot, 30, 30
preparePsPlot(), 30
sequentialFitBiasDistribution, 19, 32
sequentialFitBiasDistribution(), 7, 18
simulatePopulations, 12, 33
simulatePopulations(), 12
skewNormal, 33
supportsJava8, 34

35