Package 'EvidenceSynthesis'

November 20, 2020

Type Package

Title Synthesizing Causal Evidence in a Distributed Research Network

Version 0.2.1

Date 2020-11-20

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.

Depends survival,
R (>= 3.5.0)

Imports ggplot2,
gridExtra,
meta,
EmpiricalCalibration,
rJava,
BeastJar,
Cyclops (>= 3.1.0),
HDInterval,
coda,
rlang,
methods

Suggests knitr,
rmarkdown,
testthat,
sn

License Apache License 2.0

URL https://ohdsi.github.io/EvidenceSynthesis/, https://github.com/OHDSI/EvidenceSynthesis

BugReports https://github.com/OHDSI/EvidenceSynthesis/issues

VignetteBuilder knitr

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

R topics documented:

`approximateLikelihood` *Approximate a likelihood function*

Description

Approximate the likelihood function using a parametric (normal, skew-normal, or custom parametric), 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'`, or `'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

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

---

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

```
computeBayesianMetaAnalysis(
  data,
  chainLength = 1100000,
  burnIn = 1e+05,
  subSampleFrequency = 100,
  priorSd = c(2, 0.5),
  alpha = 0.05
)
```

Arguments

`data`	A data frame containing either normal, skew-normal, custom parametric, or grid likelihood data, with one row per database.
`chainLength`	Number of MCMC iterations.
`burnIn`	Number of MCMC iterations to consider as burn in.
`subSampleFrequency`	Subsample frequency for the MCMC.

priorSd	A two-dimensional vector with the standard deviation of the prior for mu and tau, respectively.
alpha	The alpha (expected type I error) used for the credible intervals.

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

approximateLikelihood, 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)
```

---

computeConfidenceInterval

Compute the point estimate and confidence interval given a likelihood function approximation

---

Description

Compute the point estimate and confidence interval given a likelihood function approximation

Usage

```
computeConfidenceInterval(approximation, alpha = 0.05)
```

Arguments

approximation	An approximation of the likelihood function as fitted using the approximateLikelihood() function.
alpha	The alpha (expected type I error).

Details

Compute the point estimate and confidence interval given a likelihood function approximation.

Value

A data frame containing the point estimate, and upper and lower bound of the confidence interval.

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")
computeConfidenceInterval(approximation)
```

---

computeFixedEffectMetaAnalysis

Compute a fixed-effect meta-analysis

Description

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

Usage

```
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.

Value

The meta-analytic estimate, expressed as the point estimate hazard ratio (rr), 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")
  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

Create simulation settings

Description

Create an object specifying a simulation. Currently only Cox proportional hazard models are supported.

Usage

```
createSimulationSettings(
  nSites = 5,
  n = 10000,
  treatedFraction = 0.2,
  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)
```

---

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

```
# 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

```
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 = (log(lower bound 95 percent confidence interval) - l og(effect estimate))/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

```
# 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")
```

---

plotMcmcTrace *Plot MCMC trace*

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,
  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 ['computeFixedEffectMeta-Analysis()] 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).
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)
```

`plotPerDbMcmcTrace` *Plot MCMC trace for individual databases*

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

```
# 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

```
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

```
# 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

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

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 us for the treated cohort.
`comparatorLabel`	A label to us 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 *Prepare to plot the propensity score distribution*

Description

Prepare to plot the propensity (or preference) score distribution. It computes the distribution, so the output does not contain person-level data.

Usage

```
preparePsPlot(data, unfilteredData = NULL, scale = "preference")
```

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 a 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

Walker AM, Patrick AR, Lauer MS, Hornbrook MC, Marin MG, Platt R, Roger VL, Stang P, and Schneeweiss S. (2013) A tool for assessing the feasibility of comparative effectiveness research, Comparative Effective Research, 3, 11-20

See Also

plotPreparedPs

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)
```

---

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

Azzalini, A. (2013). The Skew-Normal and Related Families. Institute of Mathematical Statistics Monographs. Cambridge University Press.

Examples

```
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 >= 8.

Usage

```
supportsJava8()
```

Value

Returns TRUE if JVM supports Java >= 8.

Examples

```
supportsJava8()
```

Index