# Package ‘DmBTC’

*October 30, 2024*

**Type** Package

**Title** Diagnostic meta-analysis using a Bivariate t distribution with Composite Link Function

**Version:** 0.0.1

**Maintainer:** Chin-Chieh Wu <wujinja@gmail.com>

**Description** The DmBTC package provides functions to conduct meta-analyses on the diagnostic accuracy of biomarkers for disease detection using a Bayesian approach. It allows users to perform Bayesian modeling of diagnostic accuracy, compare various models, generate comprehensive visualizations, and conduct diagnostics on convergence. This package is intended for researchers and statisticians analyzing the diagnostic power of biomarkers in a clinical setting.

**License** GPL (>= 2)

**Encoding** UTF-8

**LazyData** true

**URL** [wujinja-cgu/DmBTC-package]

**Depends** R (>=4.0.1), rstan, meta4diag, R2OpenBUGS, bamdit, meta, coda, forestplot, arm

**Author** Chin-Chieh Wu

Repository CRAN

Date/Publication

**R topics documented:**

- **Functions**

  1. **DmBTCmeta**: Conducts a Bayesian meta-analysis on diagnostic accuracy data, pooling results from multiple studies.
  2. **SCSmeta:** Conducts a meta-analysis based on SCS model
  3. **Bamdit**: Conducts a meta-analysis based on bamdit model
  4. **Simulation**: Conducts a simulation for diagnostic accuracy data
  5. **Compare_DmBTC**: Allows comparison of multiple diagnostic accuracy models with data from included in DmBTC package.
  6. **Compare_Simulation**: Allows comparison of multiple diagnostic accuracy models with data from simulation.
  7. **Plot_Forest**: Visualizes results as forest plots for sensitivity and specificity.
  8. **Traceplot**: Creates trace plots for convergence diagnostics in Bayesian models.
  9. **SROCplot**: Visualizes results as SROC plots for sensitivity and specificity.

- **Datasets**

  **Sepsis_data**: Sample dataset of diagnostic accuracy values for evaluating biomarkers in sepsis detection.

| DmBTC-package | *The 'DmBTC' package* |
|---|---|

## Description

The DmBTC package provides functions to conduct meta-analyses on the diagnostic accuracy of biomarkers for disease detection using a Bayesian approach. It allows users to perform Bayesian modeling of diagnostic accuracy, compare various models, generate comprehensive visualizations, and conduct diagnostics on convergence. This package is intended for researchers and statisticians analyzing the diagnostic power of biomarkers in a clinical setting.

## Author(s)

Chin-Chieh Wu <wujinja@gmail.com>

## References

XXXXXX

| **DmBTCmeta** | *Runs a Bayesian meta-analysis using the DmBTC model to estimate diagnostic accuracy metrics with customizable parameters.* |
|---|---|

## Description

DmBTCmeta performs Bayesian meta-analysis by fitting the DmBTC model using WinBUGS. It estimates diagnostic accuracy metrics, including sensitivity, specificity, and their credible intervals, with options for customizing the analysis.

**Usage**:

DmBTCmeta(data, parameters, n.iter, n.burnin, n.chains, n.thin, digits, debug, inits = NULL)

**Arguments**:

- data: The data frame containing study-level diagnostic accuracy metrics.

- parameters: A list of parameter names to estimate in the Bayesian model.
- n.iter: Integer. Number of MCMC iterations.
- n.burnin: Integer. Number of burn-in iterations to discard.
- n.chains: Integer. Number of Markov chains.
- n.thin: Integer. Thinning interval for MCMC chains.
- digits: Integer. Number of decimal places to display in the output.
- debug: Logical. If TRUE, runs the model in debug mode.
- inits: List of initial values for the parameters. Defaults to NULL.

**Details**:

DmBTCmeta relies on external WinBUGS code to perform Bayesian estimation. The function integrates diagnostic accuracy data across studies to produce posterior estimates of sensitivity and specificity. It includes flexibility for specifying the number of iterations, chains, and thinning intervals.

**Value**:

A list containing:

bugs.out: The output of the WinBUGS model, including estimates, diagnostic statistics, and convergence measures.

**Author(s)**

Chin-Chieh Wu <wujinja@gmail.com>

**References**

XXXXXX

**Examples**:

# Example Usage

library(DmBTC)

# Define the data

data <- list( sensitivity = c(0.85, 0.9, 0.78), specificity = c(0.92, 0.88, 0.85), studyID =

```
c(1, 2, 3) )
# Define parameters
parameters <- c("sensitivity", "specificity")
# Run the model
results <- DmBTCmeta( data = data, parameters = parameters, n.iter = 10000,
n.burnin = 500, n.chains = 3, n.thin = 10, digits = 3, debug = FALSE )
# View results
print(results)
```

|  |  |
|---|---|
| **SCSmeta** | *Calculates pooled estimates for diagnostic accuracy metrics, including sensitivity, specificity, diagnostic odds ratio (DOR), and area under the curve (AUC), using the Split Component Synthesis (SCS) model.* |

**Description**

SCSmeta implements the Split Component Synthesis (SCS) model for meta-analysis of diagnostic test accuracy. It adjusts for studies with zero counts using continuity corrections and provides pooled estimates for key metrics, including sensitivity, specificity, positive likelihood ratio (pLR), and negative likelihood ratio (nLR). The function also estimates the Receiver Operating Characteristic (ROC) curve based on pooled metrics.

**Usage**:

```
SCSmeta(tp, fp, fn, tn)
```

**Arguments**:

- tp: Numeric vector. True-positive counts for each study.
- fp: Numeric vector. False-positive counts for each study.
- fn: Numeric vector. False-negative counts for each study.

- tn: Numeric vector. True-negative counts for each study.

**Details**:

1. Continuity Corrections:

Applies a continuity correction (adding 0.5) to studies with zero counts for true positives, false positives, false negatives, or true negatives.

2. Metrics Computation:

Diagnostic Odds Ratio (DOR): Estimates pooled log DOR using Doi's inverse variance heterogeneity (IVhet) model.

Sensitivity and Specificity: Pooled estimates derived using centered log DOR values and linear regression.

Likelihood Ratios: Computes positive likelihood ratio (pLR) and negative likelihood ratio (nLR) with their 95% confidence intervals.

3. AUC and ROC:

Calculates the area under the curve (AUC) and confidence intervals for the ROC curve.

Produces sensitivity estimates for a sequence of specificity values (1 - specificity).

**Value**:

A list containing:

- Se_ihvet: Pooled sensitivity estimate.
- Sp_ihvet: Pooled specificity estimate.

**Author(s)**

Chin-Chieh Wu <wujinja@gmail.com>

**References**

XXXXXX

**Examples**:

```
# Example 1: Basic usage with diagnostic data
tp <- c(10, 15, 20)   # True positives
fp <- c(5, 7, 10)      # False positives
```

```
fn <- c(3, 4, 6)        # False negatives
tn <- c(50, 60, 80)   # True negatives
# Run the SCSmeta function
results <- SCSmeta(tp, fp, fn, tn)
# View results
cat("Pooled Sensitivity:", results["Se_ihvet"], "\n")
cat("Pooled Specificity:", results["Sp_ihvet"], "\n")

# Example 2: Handling zero counts in studies
tp <- c(10, 0, 20)   # True positives with zero counts in one study
fp <- c(5, 7, 10)     # False positives
fn <- c(3, 0, 6)      # False negatives with zero counts
tn <- c(50, 60, 80) # True negatives
# Run the SCSmeta function with continuity correction applied
results <- SCSmeta(tp, fp, fn, tn)
# Print results
cat("Adjusted Sensitivity:", results["Se_ihvet"], "\n")
cat("Adjusted Specificity:", results["Sp_ihvet"], "\n")
```

| | |
|---|---|
| **Simulation** | *Simulates diagnostic accuracy data for the DmBTC model, incorporating study-level characteristics such as prevalence, diagnostic odds ratio (DOR), and SROC curve symmetry.* |

**Description**

DmBTC_Simulation generates simulated datasets for diagnostic accuracy analysis. The function creates study-level data, including true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), based on input parameters such as disease prevalence, diagnostic odds ratio, and the mean values for diseased and non-diseased populations. Sensitivity and specificity are also calculated.

**Usage**:

DmBTC_Simulation(mu1, mu2, DOR, prevalence, nstudy, is_symmetric, seed)

**Arguments**:

- mu1: Numeric. Mean value of the diagnostic metric for non-diseased individuals.
- mu2: Numeric. Mean value of the diagnostic metric for diseased individuals (must be greater than mu1).
- DOR: Numeric. Diagnostic odds ratio, representing the magnitude of diagnostic accuracy.
- prevalence: Numeric. Disease prevalence across studies.
- nstudy: Integer. Number of studies to simulate.
- is_symmetric: Logical. Indicates whether the summary receiver operating characteristic (SROC) curve is symmetric.
- seed: Integer. Random seed for reproducibility.

**Details**:

1. Simulation Process:

Sensitivity and specificity are derived based on the threshold between the distributions of diseased and non-diseased populations.

Continuously samples study sizes and generates binary diagnostic outcomes based on random draws from normal distributions.

2. Outputs:

Sensitivity and specificity values for the simulated studies.

A metadata table with study-level diagnostic metrics: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

**Value**:

A list containing:

- sensitivity: Calculated sensitivity for each simulated study.
- specificity: Calculated specificity for each simulated study.
- metadata: A data frame containing the following columns:

- ID: Study identifier.
- TP: True positives.
- FN: False negatives.
- TN: True negatives.
- FP: False positives.

**Author(s)**

Chin-Chieh Wu <wujinja@gmail.com>

**References**

XXXXXX

**Examples**:

```
# Example usage
mu1 <- 1.0                  # Mean for non-diseased individuals
mu2 <- 2.5                  # Mean for diseased individuals
DOR <- 20                    # Diagnostic odds ratio
prevalence <- 0.3       # Disease prevalence
nstudy <- 5                 # Number of studies
is_symmetric <- TRUE # Symmetric SROC curve
seed <- 123                 # Random seed

# Generate simulated data
simulation_results <- DmBTC_Simulation(
  mu1 = mu1,
  mu2 = mu2,
  DOR = DOR,
  prevalence = prevalence,
  nstudy = nstudy,
  is_symmetric = is_symmetric,
  seed = seed
)
```

```
# View results
cat("Sensitivity:", simulation_results[[1]], "\n")
cat("Specificity:", simulation_results[[2]], "\n")
print(simulation_results[[3]]) # Study-level metadata
```

| | |
|---|---|
| **Compare_DmBTC** | *Performs meta-analysis using three models—DmBTC, Bamdit, and SCS—to calculate pooled sensitivity and specificity from diagnostic accuracy data.* |

**Description**

Compare_DmBTC runs the DmBTC, Bamdit, and SCS models on a dataset containing diagnostic accuracy metrics. It combines study-level metrics such as true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) to produce pooled estimates of sensitivity and specificity using each model.

**Usage**:

```
Compare_DmBTC(data)
```

**Arguments**:

- data:
  A data frame containing the following required columns:
- TP: True positives.
- FP: False positives.
- TN: True negatives.
- FN: False negatives.

**Details**:

1. Model Implementations:

DmBTC: A Bayesian meta-analysis model that uses the DmBTCmeta function.

SCS: The Split Component Synthesis model implemented via SCSmeta.

Bamdit: A bivariate diagnostic meta-analysis using the metadiag function.

2. Validation:

Ensures the input is a data frame containing all required columns (TP, FP, TN, FN).

Provides error handling to ensure smooth execution and informative error messages.

3. Output Structure:

Results from all three models are returned as a matrix, with pooled sensitivity and specificity values for each model.

**Value**:

A matrix with rows for each model (Bamdit, SCS, DmBTC) and columns for:

- Pooled Sensitivity: Pooled sensitivity estimates for the corresponding model.
- Pooled Specificity: Pooled specificity estimates for the corresponding model.

**Author(s)**

Chin-Chieh Wu <wujinja@gmail.com>

**References**

XXXXXX

**Examples**:

```
# Example usage
data <- data.frame(
   TP = c(10, 20, 30),    # True positives
   FP = c(5, 10, 15),      # False positives
   TN = c(50, 100, 150),# True negatives
   FN = c(3, 6, 9)           # False negatives
)

# Run the Compare_DmBTC function
results <- Compare_DmBTC(data)

# View results
print(results)
```

| | |
|---|---|
| **Compare_Simulation** | *Compares the performance of the DmBTC, Bamdit, and SCS models in estimating pooled sensitivity and specificity using simulated diagnostic accuracy data.* |

**Description**

Compare_Simulation takes a dataset of simulated diagnostic accuracy metrics and fits three different meta-analysis models: DmBTC, Bamdit, and SCS. The function evaluates and compares their performance, producing pooled sensitivity and specificity estimates for each model.

**Usage**:

Compare_Simulation(data)

**Arguments**:

- data:
  - A data frame containing the following required columns:
    - ID: Study identifier.
    - TP: True positives.
    - FP: False positives.
    - TN: True negatives.
    - FN: False negatives.

**Details**:

1. Model Implementations:

DmBTC: A Bayesian diagnostic meta-analysis model using the DmBTCmeta function.

SCS: The Split Component Synthesis model implemented through SCSmeta.

Bamdit: A bivariate diagnostic meta-analysis using the metadiag function.

2. Validation:

Checks if the input is a data frame and contains all required columns.

Provides informative error messages if any required column is missing.

3. Simulation Handling:

Processes simulated data and constructs appropriate input formats for each model.

Incorporates error handling to manage failed model fittings.

4. Output Structure:

Returns a matrix comparing pooled sensitivity and specificity for the three models along with the ground truth values from the simulation.

**Value**:

A matrix with rows for the ground truth and each model (TRUE, Bamdit, SCS, DmBTC) and columns for:

- Pooled Sensitivity: Pooled sensitivity estimates.
- Pooled Specificity: Pooled specificity estimates.

**Author(s)**

Chin-Chieh Wu <wujinja@gmail.com>

**References**

XXXXXX

**Examples**:

```
# Example usage
data <- data.frame(
  ID = 1:5,                      # Study IDs
  TP = c(10, 15, 20, 25, 30),    # True positives
  FP = c(5, 10, 15, 20, 25),     # False positives
  TN = c(50, 60, 70, 80, 90),    # True negatives
  FN = c(3, 6, 9, 12, 15)        # False negatives
)

# Run the Compare_Simulation function
results <- Compare_Simulation(data)
```

```
# View results
print(results)
```

| | |
|---|---|
| **Plot_Forest** | *Generates a forest plot to visualize study-level and pooled sensitivity estimates with confidence intervals based on the DmBTC model output.* |

**Description**

plot_forest creates a forest plot to summarize sensitivity estimates for individual studies and pooled results. The function takes the output from the DmBTCmeta model and visualizes the sensitivity estimates along with their confidence intervals.

**Usage**:

```
plot_forest(bugs.out)
```

**Arguments**:

**bugs.out**: The output from the DmBTCmeta model, containing sensitivity estimates and confidence intervals for each study.

**Details**:

1. Extracts sensitivity estimates (pool.se) and their lower and upper 95% credible intervals from the bugs.out object.
2. Creates a data frame with study-level sensitivity data.
3. Plots a forest plot using the meta::forest function, displaying sensitivity estimates and their confidence intervals.

**Value**:

A forest plot visualizing:

- Sensitivity estimates for individual studies.
- Pooled sensitivity with its confidence interval.

**Author(s)**

Chin-Chieh Wu <wujinja@gmail.com>

**References**

XXXXXX

**Examples**:

```
# Example: Generate forest plot using DmBTCmeta output
library(meta)
# Simulated example data
bugs.out <- list(
   sims.list = list(pool.se = c(0.85, 0.78, 0.90, 0.83)),
   summary = data.frame(
      "pool.se" = c(0.85, 0.78, 0.90, 0.83),
      "2.5%" = c(0.75, 0.70, 0.80, 0.78),
      "97.5%" = c(0.95, 0.85, 1.00, 0.87)
   )
)

# Run the function
forest_plot <- plot_forest(bugs.out)

# View plot
print(forest_plot)
```

| | |
|---|---|
| **Traceplot** | *Generates trace plots for MCMC chains to assess the convergence of parameters in Bayesian models.* |

**Description**

plot_trace creates trace plots for the Markov Chain Monte Carlo (MCMC) chains generated by the R2OpenBUGS::bugs() function. Trace plots are useful for diagnosing the convergence and mixing of MCMC chains during Bayesian estimation.

**Usage**:

plot_trace(bugs.out, parameter = NULL)

**Arguments**:

1. **bugs.out**: The output object from the R2OpenBUGS::bugs() function, containing MCMC samples for the model parameters.
2. **parameter**: Character. The name of the parameter for which the trace plot should be generated. If NULL, trace plots for all parameters will be generated. Default is NULL.

**Details**:

1. Converts the bugs.out object to an MCMC object using coda::read.bugs().
2. If a parameter name is specified, generates the trace plot for that parameter only. Otherwise, generates trace plots for all parameters in the model.
3. Trace plots are visual diagnostics for convergence and mixing, showing the sampled values of a parameter over iterations.

**Value**:

A trace plot for the specified parameter or all parameters in the model.

**Author(s)**

Chin-Chieh Wu <wujinja@gmail.com>

**References**

XXXXXX

**Examples**:

# Example: Generate trace plots for a Bayesian model

```
library(coda)

library(R2OpenBUGS)


# Simulated output from R2OpenBUGS

bugs.out <- bugs(data = ..., inits = ..., parameters.to.save = ...,

                        model.file = ..., n.chains = 3, n.iter = 10000)


# Plot trace plot for all parameters

plot_trace(bugs.out)


# Plot trace plot for a specific parameter

plot_trace(bugs.out, parameter = "mu")
```

---

| | |
|---|---|
| **SROCplot** | *Generates a Summary Receiver Operating Characteristic (SROC) plot using the output from the DmBTC model.* |

---

**Description**

plot_sroc creates an SROC plot to visualize the diagnostic accuracy of a model by plotting sensitivity against specificity. The function utilizes the meta4diag package to fit and visualize hierarchical SROC models.

**Usage**:

```
plot_sroc(data)
```

**Arguments**:

**data**: The output data from the DmBTC model, containing MCMC samples for sensitivity, specificity, and related diagnostic accuracy metrics.

**Details**:

1. Uses the meta4diag::plot function to generate the SROC curve from the provided model output.
2. Highlights the trade-off between sensitivity and specificity across diagnostic thresholds.
3. Suitable for visualizing the performance of diagnostic tests in meta-analyses.

**Value**:

An SROC plot that includes:

- Curve representing the relationship between sensitivity and specificity.
- Confidence and prediction regions for the diagnostic accuracy metrics.

**Author(s)**

Chin-Chieh Wu <wujinja@gmail.com>

**References**

XXXXXX

**Examples**:

```
# Example: Generate an SROC plot from DmBTC model output
library(meta4diag)

# Assuming you have DmBTC model output
data <- read.bugs("your_model_output.txt")    # Replace with your actual model output
# Fit the hierarchical SROC model
sroc_model <- meta4diag(data)

# Plot the SROC curve
plot_sroc(sroc_model)
```

| | |
|---|---|
| Sepsis_data | *a sample dataset with diagnostic accuracy measures from studies evaluating biomarkers for sepsis detection* |

**Description**

Sepsis_data is a sample dataset with diagnostic accuracy measures from studies evaluating biomarkers for sepsis detection. It is intended for use with functions such as DmBTCmeta and Compare.

**Usage**:

```
data("Sepsis_data")
```

**Format**:

A data frame with columns:

- studyID: Identifier for each study.
- sensitivity, specificity: Diagnostic accuracy metrics.
- ppv, npv: Optional columns for Positive Predictive Value and Negative Predictive Value.
- sample_size: Total sample size for each study.

**Details**:

This dataset enables testing and example use of DmBTCmeta functions, providing realistic data for analyzing diagnostic performance of biomarkers in detecting sepsis.

**Author(s)**

Chin-Chieh Wu <wujinja@gmail.com>

**References**

XXXXXX

**Examples**:

```
data("sepsis_data")
head(sepsis_data)
```

## Index