Package 'postlink'

September 4, 2026

Type Package

Title Post-Linkage Data Analysis

Version 0.1.1

Depends R (>= 3.5.0)

Imports stats, survival, label.switching, methods, nleqslv, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), rstantools (>= 2.5.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Description Provides a suite of statistical tools for post-linkage data analysis (PLDA), designed to account for record linkage errors in downstream modeling. The package implements a familiar, formula-based regression interface that adjusts for linkage uncertainty, accommodating workflows where direct access to unlinked primary files is restricted. It consolidates diverse adjustment methodologies, all of which support generalized linear models (linear, logistic, Poisson, and Gamma). These methodologies include weighting approaches (Chambers (2009) <https://hdl.handle.net/10779/uow.27788247>; Chambers et al. (2023) <doi:10.1002/wics.1596>), mixture modeling (Slawski et al. (2025) <doi:10.1093/jrsssa/qnae083>), and Bayesian mixture modeling (Gutman et al. (2016) <doi:10.1002/sim.6586>). For time-to-event data, both the weighting (Vo et al. (2024) <doi:10.1002/sim.9960>) and mixture modeling approaches accommodate Cox proportional hazards models, while the Bayesian approaches extend to parametric survival analysis. Additionally, the package leverages mixture modeling for contingency table analyses and Bayesian methods to enable the multiple imputation of latent match status.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Config/testthat/edition 3

BugReports https://github.com/postlink-group/postlink/issues

Biarch true

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

URL https://postlink-group.github.io/postlink/

Config/Needs/website rmarkdown

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Priyanjali Bukke [aut, cre],
Gauri Kamat [aut],
Jiahao Cui [aut],
Roee Gutman [aut],
Martin Slawski [aut],
Zhenbang Wang [ctb],
Brady T. West [ctb],
Emanuel Ben-David [ctb],
Guoqing Diao [ctb]

Maintainer Priyanjali Bukke <postlink.group@gmail.com>

Repository CRAN

Date/Publication 2026-09-04 07:00:14 UTC

Contents

postlink-package 4
adjELE 6
adjMixBayes 8
adjMixture 9
brfss 11
coef.glmELE 12
coef.glmMixBayes 13
coef.glmMixture 13
coef.survMixBayes 14
confint.coxphELE 14
confint.coxphMixture 15
confint.ctableMixture 17
confint.glmELE 18
confint.glmMixBayes 19
confint.glmMixture 20
confint.survMixBayes 21
coxphELE 23
coxphMixture 25
ctableMixture 27
df.residual.plglm 29
glmELE 30
glmMixBayes 32
glmMixture 34
LD1000 36
lifem 38
mi_with 39
mi_with.glmMixBayes 39

mi_with.survMixBayes 41
plcoxph 43
plctable 45
plglm 47
plsurvreg 49
predict.coxphELE 50
predict.coxphMixture 52
predict.glmELE 54
predict.glmMixBayes 55
predict.glmMixture 57
predict.survMixBayes 59
print.adjELE 60
print.adjMixBayes 61
print.adjMixture 62
print.adjustment 63
print.coxphELE 64
print.coxphMixture 65
print.ctableMixture 66
print.glmELE 67
print.glmMixBayes 68
print.glmMixture 69
print.mi_link_pool_glm 70
print.mi_link_pool_survreg 72
print.survMixBayes 73
summary.coxphELE 74
summary.coxphMixture 76
summary.ctableMixture 77
summary.glmELE 79
summary.glmMixBayes 80
summary.glmMixture 81
summary.survMixBayes 82
survregMixBayes 83
unsupported_generics 87
vcov.coxphELE 88
vcov.coxphMixture 89
vcov.ctableMixture 90
vcov.glmELE 91
vcov.glmMixBayes 92
vcov.glmMixture 93
vcov.survMixBayes 94

Index **96**

postlink-package *postlink: Post-Linkage Data Analysis Under Record Linkage Errors*

Description

The postlink package provides a unified suite of statistical tools designed to rigorously account for record linkage errors in downstream modeling.

Record linkage is often error-prone. When datasets are merged using noisy or non-unique identifiers, mismatches (false links) are inadvertently introduced. Ignoring these errors acts as a contaminant in regression analysis, typically leading to attenuated estimates and biased statistical inference. postlink equips researchers with methodologies to propagate linkage uncertainty into their models, in particular accommodating "secondary analysis" workflows where direct access to the primary, unlinked files is restricted.

Details

A Two-Phase Workflow

The package is built on a modular, object-oriented S3 architecture to provide a familiar, standard formula-based modeling interface. The specification of linkage error is decoupled from the substantive statistical modeling.

Phase 1: Adjustment Specification

The analyst encapsulates the linked data and the chosen error-adjustment methodology by using one of the adj* constructor functions. These functions validate the data and return a structured adjustment object:

- adjELE(): Specifies the Exchangeable Linkage Error (ELE) model, which corrects for bias using known or audited mismatch rates.
- adjMixture(): Specifies a frequentist mixture model approach that treats match status as a latent variable, estimating error rates directly from data (e.g., using match scores) via the EM algorithm.
- adjMixBayes(): Specifies a Bayesian mixture model approach, enabling parameter estimation and multiple imputation of latent match statuses using Stan.

Phase 2: Estimation & Inference

The adjustment object is subsequently passed to a standard modeling wrapper, integrating the error correction into the familiar R modeling syntax:

- plglm(): Generalized Linear Models (linear, logistic, Poisson, Gamma).
- plcoxph(): Cox Proportional Hazards models.
- plctable(): Contingency table analyses.
- plsurvreg(): Parametric survival models.

As a note, estimation and inference supported for each type of adjustment object vary. Refer to the adj* documentation for models currently supported.

Post-estimation tools function as expected in standard R workflows (e.g., `summary()`, `predict()`, `vcov()`, and `confint()`). These methods specially are derived to account for the additional steps introduced by the linkage error adjustment.

Note: While the two-phase workflow is recommended for standard analyses, the package's architecture intentionally isolates the core logic of each method into independent internal routines. Advanced users, developers, or researchers running large-scale simulations can bypass the wrapper functions and formula-parsing overhead entirely. By supplying pre-computed design matrices and response vectors, the underlying computational functions can be directly used if preferred (e.g., `coxphELE()`, `glmMixture()`, `survregMixBayes()`).

Author(s)

Maintainer: Priyanjali Bukke <postlink.group@gmail.com>

Authors:

- Priyanjali Bukke <postlink.group@gmail.com>
- Gauri Kamat
- Jiahao Cui
- Roee Gutman
- Martin Slawski

Other contributors:

- Zhenbang Wang [contributor]
- Brady T. West [contributor]
- Emanuel Ben-David [contributor]
- Guoqing Diao [contributor]

References

Chambers, R. (2009). Regression analysis of probability-linked data. *Official Statistics Research Series*, 4, 1-15.

Chambers, R. L., Fabrizi, E., Ranalli, M. G., Salvati, N., & Wang, S. (2023). Robust regression using probabilistically linked data. *Wiley Interdisciplinary Reviews: Computational Statistics*, 15(2), e1596. doi:10.1002/wics.1596

Gutman, R., Sammartino, C., Green, T., & Montague, B. (2016). Error adjustments for file linking methods using encrypted unique client identifier (eUCI) with application to recently released prisoners who are HIV+. *Statistics in Medicine*, 35(1), 115–129. doi:10.1002/sim.6586

Slawski, M., West, B. T., Bukke, P., Wang, Z., Diao, G., & Ben-David, E. (2025). A general framework for regression with mismatched data based on mixture modelling. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 188(3), 896-919. doi:10.1093/jrsssa/qnae083

Vo, T. H., Garès, V., Zhang, L. C., Happe, A., Oger, E., Paquelet, S., & Chauvet, G. (2024). Cox regression with linked data. *Statistics in Medicine*, 43(2), 296-314. doi:10.1002/sim.9960

See Also

Useful links:

- https://postlink-group.github.io/postlink/
- Report bugs at https://github.com/postlink-group/postlink/issues

---

`adjELE` *Secondary Analysis Constructor Assuming ELE*

Description

Specifies the linked data and information on the underlying record linkage process for regression of linked data assuming exchangeable linkage errors (ELE) as developed by Chambers (2009) and Vo et al. (2024). These approaches correct for bias from mismatch error via weighting matrices estimated using known mismatch rates or clerical reviews (audit samples).

Usage

```
adjELE(
  linked.data,
  m.rate,
  audit.size = NULL,
  blocks = NULL,
  weight.matrix = c("ratio", "LL", "BLUE", "all")
)
```

Arguments

`linked.data`	A data.frame containing the data after record linkage.
`m.rate`	Numeric vector; known or estimated probability of mismatch for each record or block. Values must be between 0 and 1. Can be a single global rate, a vector of length equal to the number of unique blocks, or a vector of length equal to the number of rows in `linked.data`.
`audit.size`	Numeric vector; If the m.rate is estimated, provide sample sizes for the clerical review audit. Used for variance estimation. If provided, must align with `blocks` similar to `m.rate`. Defaults to `NULL` (assume m.rate is known).
`blocks`	A vector or an unquoted variable name found in `linked.data` identifying the blocking structure used during linkage. If `NULL` (default), all records are assumed to belong to a single block.
`weight.matrix`	Character; the method for estimating the weight matrix. Must be one of "ratio" (default), "LL", "BLUE", or "all".

Details

The constructor validates consistency between the mismatch rates, audit sizes, and block identifiers. If `blocks` are provided, `m.rate` must be specified either per-block (length equals number of unique blocks) or per-record (length equals number of rows).

Explicit provision of `linked.data` is strongly recommended for reproducibility and to ensure the adjustment object fully encapsulates the necessary data for downstream model fitting.

Value

An object of class `c("adjELE", "adjustment")`. To minimize memory overhead, the underlying `linked.data` is stored by reference within an environment inside this object.

Note

The internal algorithmic structure for the estimating equations was informed by the foundational work presented in Chambers (2009) and Vo et al. (2024). Additionally, the original code provided in the Appendix of Chambers (2009) was utilized as a benchmark oracle during the unit testing phase of package development to check for numerical accuracy and validity of the implementation.

References

Chambers, R. (2009). Regression analysis of probability-linked data. *Official Statistics Research Series*, 4, 1-15.

Chambers, R. L., Fabrizi, E., Ranalli, M. G., Salvati, N., & Wang, S. (2023). Robust regression using probabilistically linked data. *Wiley Interdisciplinary Reviews: Computational Statistics*, 15(2), e1596. doi:10.1002/wics.1596

Vo, T. H., Garès, V., Zhang, L. C., Happe, A., Oger, E., Paquelet, S., & Chauvet, G. (2024). Cox regression with linked data. *Statistics in Medicine*, 43(2), 296-314. doi:10.1002/sim.9960

See Also

- `plglm()` for generalized linear regression modeling
- `plcoxph()` for Cox proportional hazards regression modeling

Examples

```
data(brfss, package = "postlink")

adj_object <- adjELE(linked.data = brfss,
                     m.rate = unique(brfss$m.rate),
                     blocks = imonth,
                     weight.matrix = "BLUE")
```

`adjMixBayes` *Secondary Analysis Constructor Based on Bayesian Mixture Modeling*

Description

Specifies the linked data and information on the underlying record linkage process for a mismatch error adjustment using a Bayesian framework based on mixture modeling as developed by Gutman et al. (2016). This framework uses a mixture model for pairs of linked records whose two components reflect distributions conditional on match status, i.e., correct match or mismatch. Posterior inference is carried out via data augmentation or multiple imputation.

Usage

```
adjMixBayes(linked.data = NULL, priors = NULL)
```

Arguments

`linked.data`	A data.frame containing the linked dataset.
`priors`	A named `list` (or `NULL`) of prior specifications. Because the Stan models are pre-compiled, these strings are parsed into numeric hyperparameters and passed to the model's data block. Any missing entries are automatically filled with symmetric defaults dynamically during the model fitting phase. For GLM models, intercept and slope priors are decoupled. Use `intercept1`/`intercept2` for the intercept of each component and `beta1`/`beta2` for the slope coefficients. This is particularly useful when the mismatch component slopes should be shrunk toward zero while allowing the intercept to remain unrestricted.

Details

Explicit provision of `linked.data` is strongly recommended for reproducibility and to ensure the adjustment object fully encapsulates the necessary data for downstream model fitting.

Value

An object of class `c("adjMixBayes", "adjustment")`. To minimize memory overhead, the underlying `linked.data` is stored by reference within an environment inside this object.

References

Gutman, R., Sammartino, C., Green, T., & Montague, B. (2016). Error adjustments for file linking methods using encrypted unique client identifier (eUCI) with application to recently released prisoners who are HIV+. *Statistics in Medicine*, 35(1), 115–129. doi:10.1002/sim.6586

See Also

- `plglm()` for generalized linear regression modeling
- `plsurvreg()` for parametric survival modeling

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

# Construct the Bayesian mixture adjustment object
adj_bayes2 <- adjMixBayes(
  linked.data = lifem_small,
  priors = list(
    intercept2 = "normal(0, 10)",
    beta2 = "normal(0, 0.01)",
    theta = "beta(2, 2)"
  )
)

class(adj_bayes2)
```

`adjMixture` *Secondary Analysis Constructor Based on Mixture Modeling*

Description

Specifies the linked data and information on the underlying record linkage process for the "General Framework for Regression with Mismatched Data" developed by Slawski et al. (2025). This framework uses a mixture model for pairs of linked records whose two components reflect distributions conditional on match status, i.e., correct match or mismatch. Inference is based on composite likelihood and the EM algorithm. Examples of information about the underlying record linkage process that can be incorporated into the method if available are the assumed overall mismatch rate, safe matches, predictors of match status, or predicted probabilities of correct matches.

Usage

```
adjMixture(
  linked.data = NULL,
  m.formula = ~1,
  m.rate = NULL,
  safe.matches = NULL
)
```

Arguments

`linked.data`	A data.frame containing the linked dataset. If `NULL`, the function attempts to resolve variables specified in `m.formula` from the environment.
`m.formula`	A one-sided formula object for the mismatch indicator model, with the covariates on the right of "~". The default is an intercept-only model corresponding to a constant mismatch rate.
`m.rate`	Numeric; an optional estimate (a proportion between 0 and 1) to constrain the global mismatch rate estimate. Defaults to `NULL`.
`safe.matches`	A logical vector or an unquoted variable name found in `linked.data`; an indicator variable for safe matches (TRUE : record can be treated as a correct match and FALSE : record may be mismatched). The default is FALSE for all matches.

Details

The constructor assumes that any variables defined in `m.formula` and `safe.matches` are in `linked.data` or in the same environment. Explicit provision of `linked.data` is strongly recommended for reproducibility and to ensure the adjustment object fully encapsulates the necessary data for downstream model fitting.

Value

An object of class `c("adjMixture", "adjustment")`. To minimize memory overhead, the underlying `linked.data` is stored by reference within an environment inside this object.

References

Slawski, M., West, B. T., Bukke, P., Wang, Z., Diao, G., & Ben-David, E. (2025). A general framework for regression with mismatched data based on mixture modelling. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 188(3), 896-919. doi:10.1093/jrsssa/qnae083

Bukke, P., Ben-David, E., Diao, G., Slawski, M., & West, B. T. (2025). Cox Proportional Hazards Regression Using Linked Data: An Approach Based on Mixture Modeling. In *Frontiers of Statistics and Data Science* (pp. 181-200). Singapore: Springer Nature Singapore. doi:10.1007/97898196-07426_8

Slawski, M., Diao, G., Ben-David, E. (2021). A pseudo-likelihood approach to linear regression with partially shuffled data. *Journal of Computational and Graphical Statistics*. 30(4), 991-1003. doi:10.1080/10618600.2020.1870482

See Also

- `plglm()` for generalized linear regression modeling
- `plcoxph()` for Cox proportional hazards regression modeling
- `plctable()` for contingency table analysis

Examples

```
# Load the LIFE-M demo dataset
data(lifem)

# Phase 1: Adjustment Specification
# We model the correct match indicator via logistic regression using
# name commonness scores (commf, comml) and a 5% expected mismatch rate.
adj_object <- adjMixture(
 linked.data = lifem,
 m.formula = ~ commf + comml,
 m.rate = 0.05,
 safe.matches = hndlnk
)
```

---

brfss *Behavioral Risk Factor Surveillance System (BRFSS) 2013 Subsample*

Description

The `brfss` data set contains a randomly selected subsample of 2,000 respondents from the 2013 Behavioral Risk Factor Surveillance System (BRFSS) survey. The BRFSS is an ongoing, state-based telephone survey conducted by the U.S. Centers for Disease Control and Prevention (CDC) that collects information on health-related risk behaviors, chronic health conditions, and the use of preventive health services from non-institutionalized adults.

Usage

```
data(brfss)
```

Format

A data frame with 2,000 rows and 8 variables.

Details

In the context of post-linkage data analysis (PLDA), this data set is partitioned into two separate files to simulate a secondary analysis scenario where disparate data sources must be linked using quasi-identifiers (e.g., state, month of interview, sex, and marital status) prior to performing regression analyses (e.g., predicting weight based on height, physical health, and mental health).

Pre-processing steps have been applied to this subsample to remove missing values ("Refused" or `NA`) and to filter out extreme outliers in physical measurements.

- `X`: State of residence (categorical).
- `imonth`: Month of interview (categorical).
- `Weight`: Weight of the respondent in pounds. Records with a weight less than 50 lbs or greater than 650 lbs were excluded.

- `Height`: Height of the respondent, formulated as (feet * 100 + inches). Records with a height less than 54 inches or greater than 84 inches were excluded.
- `Physhlth`: Number of days with poor physical health. Records marked "Refused" or NA were excluded.
- `Menthlth`: Number of days with poor mental health. Records marked "Refused" or NA were excluded.
- `Exerany`: Indicator of whether the respondent engaged in physical activity. Records marked "Refused" or NA were excluded.
- `m.rate`: Block-wise (by interview month) mismatch rate.

References

Centers for Disease Control and Prevention (CDC) (2013). "Behavioral Risk Factor Surveillance System Survey Data." U.S. Department of Health and Human Services, Centers for Disease Control and Prevention. Available at https://www.cdc.gov/brfss/annual_data/annual_2013.html.

`coef.glmELE` *Extract Coefficients for glmELE*

Description

Extract Coefficients for glmELE

Usage

```
## S3 method for class 'glmELE'
coef(object, weight.matrix = NULL, ...)
```

Arguments

`object`	A fitted glmELE object.
`weight.matrix`	Character specifying which weighting method to return.
`...`	Additional arguments.

coef.glmMixBayes *Extract Coefficients for glmMixBayes*

Description

Extract Coefficients for glmMixBayes

Usage

```
## S3 method for class 'glmMixBayes'
coef(object, ...)
```

Arguments

object	A fitted glmMixBayes object.
...	Additional arguments.

coef.glmMixture *Extract Coefficients for glmMixture*

Description

Extract Coefficients for glmMixture

Usage

```
## S3 method for class 'glmMixture'
coef(object, ...)
```

Arguments

object	A fitted glmMixture object.
...	Additional arguments.

coef.survMixBayes *Extract Coefficients for survMixBayes*

Description

Extract Coefficients for survMixBayes

Usage

```
## S3 method for class 'survMixBayes'
coef(object, ...)
```

Arguments

object	A fitted survMixBayes object.
...	Additional arguments.

confint.coxphELE *Confidence Intervals for coxphELE Objects*

Description

Computes Wald confidence intervals for the model coefficients.

Usage

```
## S3 method for class 'coxphELE'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	An object of class coxphELE.
parm	A specification of which parameters are to be given confidence intervals, either a vector of numbers or names. If missing, all parameters are considered.
level	The confidence level required (default 0.95).
...	Additional arguments (currently ignored).

Value

A matrix (or vector) with lower and upper confidence limits for each parameter.

Examples

```
library(survival)
set.seed(104)
n <- 200

# 1. Simulate covariates
age_centered <- rnorm(n, 0, 5)
treatment <- rbinom(n, 1, 0.5)

# 2. Simulate true survival times
true_time <- rexp(n, rate = exp(0.05 * age_centered - 0.6 * treatment))
cens_time <- rexp(n, rate = 0.2)
time <- pmin(true_time, cens_time)
status <- as.numeric(true_time <= cens_time)

# 3. Induce 15% Exchangeable Linkage Error (ELE)
mis_idx <- sample(1:n, size = floor(0.15 * n))
linked_age <- age_centered
linked_trt <- treatment

# False links drawn uniformly from the target population
false_link_idx <- sample(1:n, size = length(mis_idx), replace = TRUE)
linked_age[mis_idx] <- age_centered[false_link_idx]
linked_trt[mis_idx] <- treatment[false_link_idx]

linked_data <- data.frame(time = time, status = status,
                          age = linked_age, treatment = linked_trt)

# 4. Fit the adjusted Cox PH model
adj <- adjELE(linked.data = linked_data, m.rate = 0.15)
fit <- plcoxph(Surv(time, status) ~ age + treatment, adjustment = adj)

# 5. Compute confidence intervals
confint(fit) # 95% CI for all coefficients
confint(fit, parm = "treatment", level = 0.90) # 90% CI for a specific parameter
```

`confint.coxphMixture` *Confidence Intervals for coxphMixture Objects*

Description

Computes Wald confidence intervals for the coefficients of the outcome model and the mismatch indicator model.

Usage

```
## S3 method for class 'coxphMixture'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	An object of class coxphMixture.
parm	A specification of which parameters are to be given confidence intervals, either a vector of numbers or names. If missing, all parameters are considered.
level	The confidence level required (default is 0.95).
...	Additional arguments (currently ignored).

Details

The intervals are calculated based on the variance-covariance matrix returned by vcov.coxphMixture, using the standard normal approximation: Estimate +/- z_crit * SE.

Value

A matrix (or vector) with lower and upper confidence limits for each parameter.

Examples

```
library(survival)
set.seed(201)

# Simulate survival data (N = 200)
n <- 200
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.5)
true_time <- rexp(n, rate = exp(0.5 * x1 - 0.5 * x2))
cens_time <- rexp(n, rate = 0.5)

# Simulate auxiliary match scores and linkage errors
# Lower match scores correspond to a higher probability of mismatch
match_score <- runif(n, 0.5, 1.0)
is_mismatch <- rbinom(n, 1, prob = 1 - match_score)

# Induce linkage errors by shuffling covariates of mismatched records
linked_x1 <- x1
linked_x2 <- x2
mis_idx <- which(is_mismatch == 1)
shuffled_idx <- sample(mis_idx)
linked_x1[mis_idx] <- x1[shuffled_idx]
linked_x2[mis_idx] <- x2[shuffled_idx]

linked_data <- data.frame(
  time = pmin(true_time, cens_time),
  status = as.numeric(true_time <= cens_time),
  x1 = linked_x1, x2 = linked_x2,
  match_score = match_score
)

# Fit the Cox PH Mixture Model (Slawski et al., 2023)
adj <- adjMixture(linked.data = linked_data, m.formula = ~ match_score)
```

```
fit <- plcoxph(Surv(time, status) ~ x1 + x2, adjustment = adj,
               control = list(max.iter = 15))

# Extract Confidence Intervals
confint(fit)
confint(fit, parm = "treatment", level = 0.90)
```

`confint.ctableMixture` *Confidence Intervals for Adjusted Cell Probabilities*

Description

Computes Wald-type confidence intervals for the estimated cell probabilities of the correctly matched population.

Usage

```
## S3 method for class 'ctableMixture'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	An object of class `ctableMixture`.
`parm`	A specification of which parameters are to be given confidence intervals. If missing, all parameters (cells) are considered.
`level`	The confidence level required. Defaults to 0.95.
`...`	Additional arguments (currently ignored).

Details

The intervals are calculated using the standard error estimates derived from `vcov.ctableMixture`. The lower and upper bounds are truncated at 0 and 1, respectively, to ensure valid probability estimates.

Value

A matrix with columns giving lower and upper confidence limits for each parameter.

Examples

```
set.seed(125)
n <- 300

# 1. Simulate true categorical data with dependency
exposure <- sample(c("low", "high"), n, replace = TRUE)

# Induce dependency - High exposure -> higher disease probability
```

```
prob_disease <- ifelse(exposure == "high", 0.7, 0.3)
true_disease <- ifelse(runif(n) < prob_disease, "yes", "no")

# 2. Induce 15% linkage error
mis_idx <- sample(1:n, size = floor(0.15 * n))
obs_disease <- true_disease
obs_disease[mis_idx] <- sample(obs_disease[mis_idx])

linked_df <- data.frame(exposure = exposure, disease = obs_disease)

# 3. Fit the adjusted contingency table model
adj <- adjMixture(linked.data = linked_df, m.rate = 0.15)
fit <- plctable(~ exposure + disease, adjustment = adj)

# 4. Compute confidence intervals
# 95% CI for all cell probabilities
confint(fit)

# 90% CI for specific cells by name
confint(fit, parm = c("(low, yes)", "(high, no)"), level = 0.90)
```

`confint.glmELE` *Confidence Intervals for* `glmELE` *Objects*

Description

Computes Wald confidence intervals for one or more parameters in a `glmELE` object.

Usage

```
## S3 method for class 'glmELE'
confint(object, parm, level = 0.95, weight.matrix = NULL, ...)
```

Arguments

`object`	An object of class `"glmELE"`.
`parm`	A specification of which parameters are to be given confidence intervals, either a vector of numbers or names. If missing, all parameters are considered.
`level`	The confidence level required.
`weight.matrix`	Character string specifying the weighting method to use. Defaults to the first method found.
`...`	Additional arguments passed to methods.

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter.

Examples

```
data(brfss, package = "postlink")

adj_object <- adjELE(linked.data = brfss,
                     m.rate = unique(brfss$m.rate),
                     blocks = imonth,
                     weight.matrix = "BLUE")

fit <- plglm(Weight ~ Height + Physhlth + Menthlth + Exerany,
             family = "gaussian", adjustment = adj_object)

confint(fit)
```

---

confint.glmMixBayes *Credible Intervals for Regression Coefficients from a glmMixBayes Fit*

Description

Computes posterior credible intervals for the regression coefficients in a fitted `glmMixBayes` model. By default, intervals are returned for all coefficients in component 1 of the mixture model. A subset of coefficients can be selected using `parm`.

Usage

```
## S3 method for class 'glmMixBayes'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

`object`	A `glmMixBayes` model object.
`parm`	Optional. Names or numeric indices of coefficients for which credible intervals should be returned. If `NULL`, intervals are returned for all coefficients.
`level`	Probability level for the credible intervals. Defaults to `0.95`.
`...`	Not used.

Value

A matrix with one row per coefficient and two columns giving the lower and upper credible interval bounds. Row names correspond to coefficient names.

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
```

```
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

x <- cbind(1, poly(lifem_small$unit_yob, 3, raw = TRUE))
y <- lifem_small$age_at_death

adj <- adjMixBayes(
  linked.data = lifem_small,
  priors = list(theta = "beta(2, 2)")
)

fit <- plglm(
  age_at_death ~ poly(unit_yob, 3, raw = TRUE),
  family = "gaussian",
  adjustment = adj,
  control = list(
    iterations = 200,
    burnin.iterations = 100,
    seed = 123
  )
)

confint(fit)
```

`confint.glmMixture` *Confidence Intervals for glmMixture Objects*

Description

Computes Wald confidence intervals for one or more parameters in a `glmMixture` object.

Usage

```
## S3 method for class 'glmMixture'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	An object of class `glmMixture`.
`parm`	A specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	The confidence level required.
`...`	Additional arguments (currently ignored).

Details

The intervals are calculated based on the sandwich variance estimator: `Estimate +/- z_crit * SE`. For Gaussian and Gamma families, a t-distribution is used with residual degrees of freedom. For Binomial and Poisson families, a standard normal distribution is used.

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter.

Examples

```
# Load the LIFE-M demo dataset
data(lifem)

# Phase 1: Adjustment Specification
# We model the correct match indicator via logistic regression using
# name commonness scores (commf, comml) and a 5% expected mismatch rate.
adj_object <- adjMixture(
 linked.data = lifem,
 m.formula = ~ commf + comml,
 m.rate = 0.05,
 safe.matches = hndlnk
)

# Phase 2: Estimation & Inference
# Fit a Gaussian regression model utilizing a cubic polynomial for year of birth.
fit <- plglm(
 age_at_death ~ poly(unit_yob, 3, raw = TRUE),
 family = "gaussian",
 adjustment = adj_object
)

confint(fit)
```

---

confint.survMixBayes *Credible Intervals for Parameters from a survMixBayes Fit*

Description

Computes posterior credible intervals for the regression coefficients, mixing weight, and family-specific distribution parameters from a fitted `survMixBayes` model. The returned intervals are organized by mixture component and parameter type. Component 1 corresponds to the correct-match component and component 2 to the incorrect-match component.

Usage

```
## S3 method for class 'survMixBayes'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

object	An object of class survMixBayes.
parm	Optional character vector selecting which interval blocks to return. For example, "theta" returns only the credible interval for the mixing weight. If NULL, credible intervals are returned for all available parameter blocks.
level	Probability level for the credible intervals. Defaults to 0.95.
...	Further arguments (unused).

Value

A named list of credible intervals. Elements coef1 and coef2 are matrices with one row per regression coefficient and two columns giving the lower and upper interval bounds for components 1 and 2, respectively, where component 1 is the correct-match component and component 2 is the incorrect-match component. Elements such as theta, shape1, shape2, scale1, and scale2 are numeric vectors of length 2 containing the lower and upper credible interval bounds for the corresponding scalar parameters.

Examples

```
set.seed(301)
n <- 150
trt <- rbinom(n, 1, 0.5)

# Simulate Weibull AFT data
true_time <- rweibull(n, shape = 1.5, scale = exp(1 + 0.8 * trt))
cens_time <- rexp(n, rate = 0.1)
true_obs_time <- pmin(true_time, cens_time)
true_status <- as.integer(true_time <= cens_time)

# Induce linkage mismatch errors in approximately 20% of records
is_mismatch <- rbinom(n, 1, 0.2)
obs_time <- true_obs_time
obs_status <- true_status
mismatch_idx <- which(is_mismatch == 1)

shuffled <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled]
obs_status[mismatch_idx] <- obs_status[shuffled]

linked_df <- data.frame(time = obs_time, status = obs_status, trt = trt)

adj <- adjMixBayes(linked.data = linked_df)

fit <- plsurvreg(
  survival::Surv(time, status) ~ trt,
  dist = "weibull",
  adjustment = adj,
  control = list(iterations = 200, burnin.iterations = 100, seed = 123)
)
```

```
# Calculate 95% credible intervals for all parameters
confint(fit, level = 0.95)

# Extract credible intervals specifically for the mixing weight
confint(fit, parm = "theta", level = 0.90)
```

---

coxphELE *Fit a CoxPH Model Assuming Exchangeable Linkage Errors*

Description

Fit Cox proportional hazards regression adjusted for mismatched data based on the approach developed in Vo et al., 2024 assuming exchangeable linkage errors. Block-wise mismatch rates are assumed to be known.

Usage

```
coxphELE(
  x,
  y,
  cens,
  m.rate,
  blocks,
  audit.size = NULL,
  control = list(init.beta = NULL),
  ...
)
```

Arguments

x	A matrix or data.frame of covariates (design matrix).
y	A numeric vector of observed time-to-event outcomes.
cens	A numeric vector indicating censoring status (1 = censored, 0 = event). Note: This is the reverse of the standard Surv object convention where 1 usually indicates an event.
m.rate	block-wise mismatch rates (should be a vector with length equal to the number of blocks) - by default assume a single block.
blocks	block indicators.
audit.size	a vector of block sizes in the audit sample (selected by simple random sampling) if used to estimate the m.rate (optional). If a single value is provided, assume the same value for all blocks and put out a warning.
control	an optional list variable to of control arguments including "init.beta" for the initial outcome model coefficient estimates) - by default is the naive estimator.
...	the option to directly pass "control" arguments

Value

a list of results from the function called depending on the "family" specified.

`coefficients`	the outcome model coefficient estimates
`var`	the variance-covariance matrix
`linear.predictors`	the linear predictors
`means`	Column means of the covariate matrix `x`.
`n`	Number of observations.
`nevent`	Number of events.

References

Vo, T. H., Garès, V., Zhang, L. C., Happe, A., Oger, E., Paquelet, S., & Chauvet, G. (2024). Cox regression with linked data. Statistics in Medicine, 43(2), 296-314.

Examples

```
set.seed(101)
n <- 250

# 1. Simulate true data
age <- rnorm(n, 60, 8)
treatment <- rbinom(n, 1, 0.5)

# Simulate true hazard
# Older age and lack of treatment increase hazard
true_hazard <- exp(0.05 * (age - 60) - 0.5 * treatment)
true_time <- rexp(n, rate = true_hazard)
cens_time <- rexp(n, rate = 0.1)

obs_time <- pmin(true_time, cens_time)
status <- as.numeric(true_time <= cens_time) # 1 = event, 0 = censored

# 2. Induce 15% exchangeable linkage errors
mis_idx <- sample(1:n, size = floor(0.15 * n))
linked_age <- age
linked_treatment <- treatment
false_link_idx <- sample(1:n, size = length(mis_idx), replace = TRUE)
linked_age[mis_idx] <- age[false_link_idx]
linked_treatment[mis_idx] <- treatment[false_link_idx]

# 3. Prepare matrices for the internal routine
cens_ele <- 1 - status
X_mat <- cbind(age = linked_age, treatment = linked_treatment)

# 4. Fit the model directly
# cens = 1 for censored, 0 for event
fit_ele <- coxphELE(x = X_mat, y = obs_time, cens = cens_ele, m.rate = 0.15)
```

```
print(fit_ele$coefficients)
```

`coxphMixture` *CoxPH with Mixture-Based Linkage Error Adjustment*

Description

Fits a Cox proportional hazards regression adjusting for mismatched data using a mixture modeling framework in the secondary analysis setting. The method relies on a two-component mixture model where true matches follow the Cox model and mismatches follow the marginal distribution of the survival outcome. Variance estimates are obtained via Louis' method.

Usage

```
coxphMixture(
  x,
  y,
  cens,
  z,
  m.rate = NULL,
  safe.matches = NULL,
  control = list(),
  ...
)
```

Arguments

`x`	A matrix or data.frame of covariates (design matrix).
`y`	A numeric vector of observed time-to-event outcomes.
`cens`	A numeric vector indicating censoring status (1 = censored, 0 = event). Note: This is the reverse of the standard `Surv` object convention where 1 usually indicates an event.
`z`	A matrix or data.frame of mismatch covariates (e.g., match scores, blocking variables). Used to model the probability of a mismatch.
`m.rate`	An optional numeric value between 0 and 1 specifying the assumed overall mismatch rate upper bound. If provided, the mismatch indicator model is constrained such that the average estimated mismatch rate does not exceed this bound.
`safe.matches`	A logical vector indicating records known to be correct matches (TRUE). These records are fixed as matches (probability 1) during estimation. Defaults to all FALSE.
`control`	An optional list of control parameters. Parameters can also be passed directly via `...`.

- `louis.k`: Number of Monte Carlo iterations for variance estimation (default: 1000).
- `max.iter`: Maximum EM iterations (default: 1000).
- `cmax.iter`: Maximum iterations for the constrained optimization subroutine (default: 1000).
- `tol`: Convergence tolerance (default: 1e-4).
- `init.beta`: Initial estimates for outcome model coefficients.
- `init.gamma`: Initial estimates for mismatch model coefficients.
- `fy`: Pre-calculated marginal density of the response. If NULL, estimated non-parametrically.

`...` Additional arguments passed to `control`.

Value

An list of results:

`coefficients`	Estimated coefficients for the outcome model (beta).
`m.coefficients`	Estimated coefficients for the mismatch model (gamma).
`var`	Variance-covariance matrix of the estimates.
`linear.predictors`	Linear predictors for the outcome model.
`means`	Column means of the covariate matrix `x`.
`n`	Number of observations.
`nevent`	Number of events.
`match.prob`	Posterior probabilities that each observation is a correct match.
`objective`	Value of the negative log pseudo-likelihood at each iteration.
`converged`	Logical indicating if the algorithm converged.
`Lambdahat0`	Estimated baseline cumulative hazard.
`gLambdahat0`	the baseline cumulative hazard for the marginal density of the response variable (using Nelson-Aalen estimator)

References

Bukke, P., Ben-David, E., Diao, G., Slawski, M., & West, B. T. (2025). Cox Proportional Hazards Regression Using Linked Data: An Approach Based on Mixture Modelling.

Examples

```
library(survival)
set.seed(123)
n <- 200
# Generate covariates
x_cov <- seq(-3, 3, length = n)
d_cov <- rep(0:1, each = n/2)
X <- cbind(d_cov, x_cov, x_cov * d_cov)
```

```
# True parameters
b <- c(-1.5, 1, 0.5)
sigma <- 0.25
mu <- X %*% b
y <- exp(drop(mu)) * rweibull(n, shape = 1/sigma)

# Censoring
cens <- (y >= 1.5)
y[cens] <- 1.5

# Generate mismatch errors
ps <- rbeta(n, 4.5, 0.5)
logit_ps <- log(ps / (1 - ps))
mp <- cbind(1, logit_ps)
gamma_true <- c(-0.5, 1)
m <- 1 - rbinom(n, prob = plogis(mp %*% gamma_true), size = 1)
yperm <- y
shuffled_ix <- sample(which(m == 1))
yperm[shuffled_ix] <- yperm[sample(shuffled_ix)]

# Fit model
fit <- coxphMixture(x = X, y = yperm, cens = as.numeric(cens),
                    z = matrix(logit_ps, ncol = 1),
                    control = list(max.iter = 50))

print(fit)
summary(fit)
```

`ctableMixture` *Contingency Table Analysis with Mixture-Based Adjustment*

Description

Estimates the cell probabilities of a two-way contingency table in the presence of linkage errors. The function implements the methodology described in Slawski et al. (2025), modeling the observed table as a mixture of correctly matched records (following a saturated model) and mismatched records (assumed to follow an independence model).

Usage

```
ctableMixture(tab, m.rate, control = list(), ...)
```

Arguments

`tab`	A numeric matrix or table of counts representing the observed two-way contingency table.
`m.rate`	A numeric value between 0 and 1 indicating the assumed rate of mismatched records in the data.

`control` A list of control parameters. See `...` for details.

`...` Additional control arguments. If not provided in `control`, these will be used. Supported arguments:

- `max.iter`: Integer. Maximum number of EM iterations (default: 1000).
- `tol`: Numeric. Convergence tolerance for the negative log-likelihood (default: 1e-6).

Details

In the absence of linkage errors, the standard estimator for cell probabilities is the matrix of observed relative frequencies. When linkage errors are present (specifically mismatches), this estimator is biased.

This function corrects for this bias using an Expectation-Maximization (EM) algorithm. The observed data is modeled as a mixture:

$$P_{obs} = (1 - \alpha)P_{sat} + \alpha P_{ind}$$

where:

- α (`m.rate`) is the mismatch rate (fixed/known).
- P_{sat} is the distribution of correct matches (saturated model).
- P_{ind} is the distribution of mismatches (independence model).

The algorithm iteratively updates the posterior probability of a record being a correct match (E-step) and the estimates for the saturated and independence distributions (M-step).

Value

A list of results:

- `phat`: Matrix of estimated cell probabilities for the correctly matched population (the target parameter).
- `phat0`: Matrix of estimated cell probabilities for the mismatched population (independence model).
- `var`: Estimated variance-covariance matrix of the estimators in `phat`.
- `ftable`: The estimated contingency table of counts for the correctly matched population (adjusted for bias).
- `objective`: The final value of the negative log-likelihood.
- `converged`: Logical indicating if the algorithm converged within `max.iter`.

References

Slawski, M., West, B. T., Bukke, P., Wang, Z., Diao, G., & Ben-David, E. (2025). A general framework for regression with mismatched data based on mixture modelling. *Journal of the Royal Statistical Society Series A*.

Examples

```
# Generate Synthetic Data
set.seed(1234)
K <- 3; L <- 4
n <- 1000
# Define true probabilities for a KxL table
cellprobs <- c(0.18, 0.05, 0.03, 0.04, 0.02, 0.14,
               0.02, 0.02, 0.10, 0.21, 0.15, 0.04)
matrix_probs <- matrix(cellprobs, nrow = K, ncol = L)

# Generate multinomial counts
dat <- stats::rmultinom(n = n, size = 1, prob = cellprobs)
obs_idx <- apply(dat, 2, function(x) which(x == 1))
X <- ceiling(obs_idx / L) # Row indices
Y <- (obs_idx %% L); Y[Y == 0] <- L # Col indices

# Introduce Linkage Error (Mismatches)
alpha <- 0.20 # 20% mismatch rate
n_mismatch <- round(n * alpha)
Y_perm <- Y
# Shuffle the first n_mismatch Y values to break dependence
Y_perm[1:n_mismatch] <- sample(Y[1:n_mismatch])

# Create Observed Table (with error)
tab_obs <- table(X, Y_perm)

# Apply Adjustment Method
fit <- ctableMixture(tab = tab_obs, m.rate = alpha)

# Inspect Results
print(fit$converged)
# Compare estimated Correct Counts vs True Counts (approx)
print(round(fit$ftable))
print(round(table(X, Y))) # True table without errors
```

---

df.residual.plglm *Extract Residual Degrees of Freedom for plglm Models*

Description

Extract Residual Degrees of Freedom for plglm Models

Usage

```
## S3 method for class 'plglm'
df.residual(object, ...)
```

Arguments

object	A fitted plglm object.
...	Additional arguments.

---

glmELE *GLM with ELE-Based Linkage Error Adjustment*

Description

Fits a generalized linear model (GLM) accounting for exchangeable linkage errors (ELE) as defined by Chambers (2009). It solves the unbiased estimating equations resulting from the modified mean function induced by mismatch errors.

Usage

```
glmELE(
  x,
  y,
  family = "gaussian",
  m.rate,
  audit.size = NULL,
  blocks,
  weight.matrix = "all",
  control = list(init.beta = NULL),
  ...
)
```

Arguments

x	A numeric matrix of predictors (design matrix).
y	A numeric vector of responses.
family	the type of regression model ("gaussian" - default, "poisson", "binomial", "gamma"). Standard link functions are used ("identity" for Gaussian, "log" for Poisson and Gamma, and "logit" for binomial).
m.rate	A numeric vector of mismatch rates. If the length is 1, it is replicated for all blocks. If length > 1, it must match the number of unique blocks.
audit.size	a vector of block sizes in the audit sample (selected by simple random sampling) if used to estimate the m.rate (optional). If a single value is provided, assume the same value for all blocks and put out a warning.
blocks	A vector indicating the block membership of each observation.
weight.matrix	A character string specifying the weighting method ("ratio-type", "Lahiri-Larsen", "BLUE", or "all" (default))

`control` an optional list variable to of control arguments including "init.beta" for the initial outcome model coefficient estimates) - by default is the naive estimator when the weight matrix is ratio-type or Lahiri-Larsen and is the Lahiri-Larsen estimator for the BLUE weight matrix.

`...` Pass control arguments directly.

Value

A list of results:

`coefficients` A named vector (or matrix) of coefficients for the outcome model.

`residuals` The working residuals, defined as `y - fitted.values`.

`fitted.values` The fitted mean values of the outcome model, obtained by transforming the linear predictors by the inverse of the link function.

`linear.predictors` The linear fit on the link scale.

`deviance` The deviance of the weighted outcome model at convergence.

`null.deviance` The deviance of the weighted null outcome model.

`var` The estimated variance-covariance matrix of the parameters (sandwich estimator).

`dispersion` The estimated dispersion parameter (e.g., variance for Gaussian, 1/shape for Gamma).

`rank` The numeric rank of the fitted linear model.

`df.residual` The residual degrees of freedom.

`df.null` The residual degrees of freedom for the null model.

`family` The `family` object used.

`call` The matched call.

References

Chambers, R. (2009). Regression analysis of probability-linked data. *Official Statistics Research Series*, 4, 1-15.

Examples

```
data(brfss)
brfss <- na.omit(brfss)

x <- cbind(1, subset(brfss, select = c(Height,Physhlth,Menthlth,Exerany)))
y <- brfss$Weight

fit <- glmELE(x, y, family = "gaussian",
            m.rate = unique(brfss$m.rate), blocks = brfss$imonth,
            weight.matrix = "BLUE")
```

glmMixBayes *Bayesian Two-Component Mixture Generalized Linear Model*

Description

Fits a Bayesian two-component mixture generalized linear model (GLM) using Stan. Each observation is assumed to arise from one of two latent components with component-specific regression coefficients.

Usage

```
glmMixBayes(
  X,
  y,
  family = "gaussian",
  priors = NULL,
  control = list(iterations = 10000, burnin.iterations = 1000, seed =
    sample.int(.Machine$integer.max, 1), cores = getOption("mc.cores", 1L)),
  ...
)
```

Arguments

X	A numeric design matrix (N x K), typically from `model.matrix()` upstream. Missing values are not allowed.
y	A response vector of length N. For `"gaussian"` and `"gamma"`, y should be numeric; for `"poisson"` and `"binomial"`, y should be integer-valued (or coercible without loss).
family	One of `"gaussian"`, `"poisson"`, `"binomial"`, or `"gamma"`. Controls the component-specific likelihood.
priors	A named `list` (or `NULL`) of prior specifications. Because the Stan models are pre-compiled, these strings are parsed into numeric hyperparameters and passed to the model's data block. Any missing entries are automatically filled with symmetric defaults via `fill_defaults(priors, p_family = family, model_type = "glm")`. Intercept and slope priors are decoupled. Use `intercept1`/`intercept2` for the intercept of each component and `beta1`/`beta2` for the slope coefficients. By default, intercept priors are `normal(0, 10)` (weakly informative), while slope priors follow the family-specific defaults.
control	A named `list` of MCMC tuning parameters. Supported elements include: • `iterations`: total number of MCMC iterations per chain (default `1e4`); • `burnin.iterations`: number of warm-up iterations (default `1e3`); • `seed`: random seed used for reproducibility; • `cores`: number of CPU cores used for sampling (default `getOption("mc.cores", 1L)`).

Values supplied through ... override entries in control.

... Optional arguments that override elements of control. For example, iterations = 4000, burnin.iterations = 1000, seed = 123, or cores = 2.

Details

The function supports Gaussian, Poisson, Binomial, and Gamma outcome families and returns posterior samples of the component-specific regression parameters and mixture weight.

Value

An object of class "glmMixBayes" containing (at least):

m_samples Posterior draws of aligned latent component labels (matrix of size draws × N), where component 1 corresponds to the correct-match component and component 2 to the incorrect-match component.

estimates$coefficients Posterior draws of regression coefficients for the correct-match component (component 1; draws × K).

estimates$m.coefficients Posterior draws of regression coefficients for the incorrect-match component (component 2; draws × K).

estimates$dispersion Posterior draws of the dispersion parameter for the correct-match component (component 1; family-specific).

estimates$m.dispersion Posterior draws of the dispersion parameter for the incorrect-match component (component 2; family-specific).

family The GLM family used in the model.

call The matched function call.

Label switching

Mixture models are invariant to permutations of component labels, which can lead to label switching in MCMC output. To ensure interpretable posterior summaries, this function applies a post-processing step that aligns component labels across posterior draws.

First, an optional global swap of labels (1 and 2) is performed if component 2 is more frequent overall. Then, labels are aligned across draws using the ECR-ITERATIVE-1 relabeling algorithm.

References

Gutman, R., Sammartino, C., Green, T., & Montague, B. (2016). Error adjustments for file linking methods using encrypted unique client identifier (eUCI) with application to recently released prisoners who are HIV+. *Statistics in Medicine*, 35(1), 115–129. doi:10.1002/sim.6586

Stephens, M. (2000). Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4), 795–809. doi:10.1111/14679868.00265

Papastamoulis, P. (2016). *label.switching*: An R package for dealing with the label switching problem in MCMC outputs. *Journal of Statistical Software*, 69(1), 1–24. doi:10.18637/jss.v069.c01

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

x <- cbind(1, poly(lifem_small$unit_yob, 3, raw = TRUE))
y <- lifem_small$age_at_death

fit <- glmMixBayes(
  X = x,
  y = y,
  family = "gaussian",
  control = list(
    iterations = 200,
    burnin.iterations = 100,
    seed = 123
  )
)
```

`glmMixture` *GLM with Mixture-Based Linkage Error Adjustment*

Description

Fits a generalized linear model (GLM) accounting for mismatch errors using a mixture model framework in the secondary analysis setting. The variance-covariance matrix is estimated using the sandwich formula.

Usage

```
glmMixture(
  x,
  y,
  family,
  z = cbind(rep(1, nrow(x))),
  m.rate = NULL,
  safe.matches = NULL,
  control = list(),
  ...
)
```

Arguments

x	Design matrix for the primary outcome model (numeric matrix or data frame).
y	Response vector for the primary outcome model.
family	A family object (e.g., gaussian, binomial) specifying the error distribution and link function. Can be a character string or a function.
z	Design matrix for the mismatch indicator model (mismatch covariates). If NULL, an intercept-only model is assumed.
m.rate	The assumed overall mismatch rate (a proportion between 0 and 1). If provided, it imposes a constraint on the mismatch model intercept.
safe.matches	Logical vector; TRUE indicates a "safe match" (treated as definitely correct), FALSE indicates a potential mismatch.
control	An optional list of control parameters. Arguments passed via ... will override values in this list. • max.iter: Maximum EM iterations (default: 1000). • cmax.iter: Maximum iterations for the subroutine in the constrained logistic regression function (default: 1000). • tol: Convergence tolerance (default: 1e-4). • init.beta: Initial parameter estimates for the outcome model. • init.gamma: Initial parameter estimates for the mismatch indicator model. • fy: Estimated marginal density of the response. If NULL, estimated using Kernel Density Estimation or parametric assumption.
...	Additional arguments passed to control.

Value

A list of results:

coefficients	A named vector of coefficients for the outcome model.
m.coefficients	A named vector of coefficients for the mismatch indicator model (gamma).
match.prob	The posterior correct match probabilities (weights) for each observation.
residuals	The working residuals, defined as y - fitted.values.
fitted.values	The fitted mean values of the outcome model, obtained by transforming the linear predictors by the inverse of the link function.
linear.predictors	The linear fit on the link scale.
deviance	The deviance of the weighted outcome model at convergence.
null.deviance	The deviance of the weighted null outcome model.
var	The estimated variance-covariance matrix of the parameters (sandwich estimator).
dispersion	The estimated dispersion parameter (e.g., variance for Gaussian, 1/shape for Gamma).
objective	A vector tracking the negative log pseudo-likelihood at each iteration of the EM algorithm.

converged	Logical indicating if the EM algorithm converged within max.iter.
rank	The numeric rank of the fitted linear model.
df.residual	The residual degrees of freedom.
df.null	The residual degrees of freedom for the null model.
family	The family object used.
call	The matched call.

References

Slawski, M.*, West, B. T., Bukke, P., Wang, Z., Diao, G., & Ben-David, E. (2025). A general framework for regression with mismatched data based on mixture modelling. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 188(3), 896-919. doi:10.1093/jrsssa/qnae083

Slawski, M.*, Diao, G., Ben-David, E. (2021). A pseudo-likelihood approach to linear regression with partially shuffled data. *Journal of Computational and Graphical Statistics*. 30(4), 991-1003. doi:10.1080/10618600.2020.1870482

Examples

```
data(lifem)

x <- cbind(1, poly(lifem$unit_yob, 3, raw = TRUE))
y <- lifem$age_at_death
z <- cbind(1, lifem$commf, lifem$comml)

fit <- glmMixture(x, y, family = "gaussian",
                  z, m.rate = 0.05, safe.matches = lifem$hndlnk)
```

LD1000 *Linked Dataset (n = 1000) for a Simple Illustration*

Description

The LD1000 data set contains a simulated cohort of 1,000 individuals designed to replicate a secondary analysis scenario where records from disparate data sources must be linked prior to performing regression analyses.

Usage

```
data(LD1000)
```

Format

A data frame with 1,000 rows and 10 variables.

Details

To establish this cohort, demographic covariates were generated in one file and a binary clinical outcome was generated in a separate file. The two files were linked probabilistically using the Fellegi-Sunter model without unique identifiers, and 1:1 matching was enforced via a linear sum assignment problem (LSAP). This record linkage results in a mismatch rate of approximately 20%.

`LD1000` provides variables of primary interest for secondary analysis alongside examples of linkage paradata.

- `Disease_Status`: Binary clinical outcome variable (0 or 1).
- `BMI`: Body Mass Index of the individual.
- `Age`: Age of the individual.
- `Treatment`: Binary indicator of treatment assignment (0 or 1).
- `fl_prob`: The posterior match probability assigned to the linked record pair by the probabilistic linkage algorithm (`fastLink`).
- `safe_match`: A logical indicator (`TRUE` / `FALSE`) denoting whether the linked record pair had perfect string agreement on both first and last names and exact agreement on birth year.
- `bm`: Birth month (1-12) used as a blocking variable during the probabilistic record linkage process and treated as blocks for the clerical audit.
- `block_mrate`: Estimated block-level mismatch rate. This variable was generated by simulating a clerical review (audit) of 20 randomly sampled records per birth-month block. Rates were adjusted as in Chambers (2009) to prevent values of exactly 1 or 0.
- `True_Disease_Status`: The unobservable true clinical outcome variable (0 or 1) that corresponds to the individual's actual record.
- `is_match`: The latent true match status (`TRUE` = correct match, `FALSE` = mismatch). Note: This variable is provided for benchmarking and is unobserved in real-world secondary analysis settings.

References

Chambers, R. (2009). Regression analysis of probability-linked data. *Official Statistics Research Series*, 4, 1-15.

Sariyar, M., & Borg, A. (2026). RecordLinkage: Record Linkage Functions for Linking and Deduplicating Data Sets. R package version 0.4-12.6.

Enamorado, T., Fifield, B., & Imai, K. (2023). fastLink: Fast Probabilistic Record Linkage with Missing Data. R package version 0.6.1.

`lifem` *Longitudinal Intergenerational Family Electronic Micro-Database*

Description

The `lifem` data set contains a subset of data from the Life-M project (https://life-m.org/) on 3,238 individuals born between 1883 to 1906. These records were obtained from linking birth certificates and death certificates either of two ways. A fraction of the records (2,159 records) were randomly sampled to be "hand-linked at some level" (HL). These records are high quality and were manually linked at some point by trained research assistants. The remaining records were "purely machine-linked" (ML) based on probabilistic record linkage without clerical review. The Life-M team expects the mismatch rate among these records to be around 5% (Bailey et al. 2022). Of interest is the relationship between age at death and year of birth. The `lifem` demo data set consists of 2,159 hand-linked records and 1,079 records that were randomly sampled from the purely machine-linked records (~2:1 HL-ML ratio).

Usage

```
data(lifem)
```

Format

a data frame with 3,238 rows and 6 variables

Details

- yob: year of birth (value from 1883 and 1906)
- unit_yob: yob re-scaled to the unit interval for analysis (between 0 and 1). If X is the yob, we use the following: (X – min(X)) / (max(X) – min(X)) = a * X + b, a = 1/(max(X) – min(X)), b = -min(X)*a
- age_at_death: age at death (in years)
- hndlnk: whether record was purely machine-linked or hand-linked at some level.
- commf: commonness score of first name (between 0 and 1). It is based on the 1940 census. It is a ratio of the log count of the individual's first name over the log count of the most commonly occurring first name in the census.
- comml: commonness score of last name (between 0 and 1). It is based on the 1940 census. It is a ratio of the log count of the individual's last name over the log count of the most commonly occurring last name in the census.

References

Bailey, Martha J., Lin, Peter Z., Mohammed, A.R. Shariq, Mohnen, Paul, Murray, Jared, Zhang, Mengying, and Prettyman, Alexa. LIFE-M: The Longitudinal, Intergenerational Family Electronic Micro-Database. Ann Arbor, MI: Inter-university Consortium for Political and Social Research (distributor), 2022-12-21. < doi:10.3886/E155186V5 >

mi_with *Pool Parameter Estimates Across Posterior Draws*

Description

Generic function for pooling parameter estimates from Bayesian mixture models using posterior draws of the latent component indicators.

Usage

```
mi_with(object, ...)
```

Arguments

object	A fitted Bayesian mixture model object.
...	Additional arguments passed to methods.

Value

A pooled model object combining parameter estimates across posterior component-indicator draws.

Examples

```
# mi_with() is a generic function for posterior allocation-based pooling.
# See ?mi_with.glmMixBayes for a complete example illustrating its use
# with Bayesian GLM mixture models.
```

mi_with.glmMixBayes *Pooling Regression Fits Across Posterior Draws of Correct-Match Classifications*

Description

Use posterior draws of the latent match indicators from `glmMixBayes()` to repeatedly identify which records are treated as correct matches, refit the requested regression model on those records, and pool the resulting estimates.

Each retained posterior draw defines one subset of records classified as correct matches. The function fits the specified `lm()` or `glm()` model to that subset, extracts the estimated coefficients and their covariance matrix, and combines the results across draws using multiple-imputation pooling rules.

Usage

```
## S3 method for class 'glmMixBayes'
mi_with(
  object,
  data,
  formula,
  family = NULL,
  min_n = NULL,
  quietly = TRUE,
  ...
)
```

Arguments

object	A glmMixBayes model object containing posterior draws of the latent match indicators.
data	A data.frame with all candidate records in the same row order as used in the model.
formula	Model formula for refitting on each draw (required).
family	A stats::family() object for the refitted model. If not supplied, the function chooses a default family based on object$family.
min_n	Minimum number of records required to fit the model for a given posterior draw. The default is p + 1, where p is the number of columns in the model matrix.
quietly	If TRUE, draws that lead to fitting errors are skipped without printing the full error message.
...	Additional arguments passed through (currently unused).

Value

An object of class c("mi_link_pool_glm", "mi_link_pool") containing pooled coefficient estimates, standard errors, confidence intervals, and related summary information.

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

x <- cbind(1, poly(lifem_small$unit_yob, 3, raw = TRUE))
y <- lifem_small$age_at_death

adj <- adjMixBayes(
```

```
  linked.data = lifem_small,
  priors = list(theta = "beta(2, 2)")
)

fit <- plglm(
  age_at_death ~ poly(unit_yob, 3, raw = TRUE),
  family = "gaussian",
  adjustment = adj,
  control = list(
    iterations = 200,
    burnin.iterations = 100,
    seed = 123
  )
)

pooled_fit <- mi_with(
  object = fit,
  data = lifem_small,
  formula = age_at_death ~ poly(unit_yob, 3, raw = TRUE),
  family = gaussian()
)

print(pooled_fit)
```

`mi_with.survMixBayes` *Pool Regression Fits Across Posterior Draws of Correct-Match Classifications*

Description

Use posterior draws of the latent match indicators from `survregMixBayes()` to repeatedly identify which records are treated as correct matches, refit a Cox proportional hazards model on those records, and pool the resulting estimates using multiple-imputation pooling rules.

Each retained posterior draw defines one subset of records classified as correct matches. The function fits the specified `survival::coxph()` model to that subset, extracts the estimated coefficients and covariance matrix, and combines the results across draws using Rubin's rules.

Usage

```
## S3 method for class 'survMixBayes'
mi_with(
  object,
  data,
  formula,
  min_n = NULL,
  quietly = TRUE,
  ties = "efron",
```

```
  ...
)
```

Arguments

object	A survMixBayes model object containing posterior draws of the latent match indicators.
data	A data.frame with all candidate records in the same row order as used in the model.
formula	Model formula for refitting on each draw (required), typically of the form survival::Surv(time, event) ~ ....
min_n	Minimum number of records required to fit the model for a given posterior draw. The default is p + 2, where p is the number of non-intercept columns in the model matrix.
quietly	If TRUE, draws that lead to fitting errors are skipped without printing the full error message.
ties	Method for handling tied event times in survival::coxph(). Default is "efron".
...	Additional arguments passed to survival::coxph().

Value

An object of class c("mi_link_pool_survreg", "mi_link_pool") containing pooled coefficient estimates, standard errors, confidence intervals, and related summary information.

Examples

```
set.seed(301)
n <- 150
trt <- rbinom(n, 1, 0.5)

# Simulate Weibull AFT data
true_time <- rweibull(n, shape = 1.5, scale = exp(1 + 0.8 * trt))
cens_time <- rexp(n, rate = 0.1)
true_obs_time <- pmin(true_time, cens_time)
true_status <- as.integer(true_time <= cens_time)

# Induce linkage mismatch errors in approximately 20% of records
is_mismatch <- rbinom(n, 1, 0.2)
obs_time <- true_obs_time
obs_status <- true_status
mismatch_idx <- which(is_mismatch == 1)

shuffled <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled]
obs_status[mismatch_idx] <- obs_status[shuffled]

linked_df <- data.frame(time = obs_time, status = obs_status, trt = trt)
adj <- adjMixBayes(linked.data = linked_df)
```

```
fit <- plsurvreg(
  survival::Surv(time, status) ~ trt,
  dist = "weibull",
  adjustment = adj,
  control = list(iterations = 200, burnin.iterations = 100, seed = 123)
)

pooled_obj <- mi_with(
  object = fit,
  data = linked_df,
  formula = survival::Surv(time, status) ~ trt
)

print(pooled_obj)
```

---

plcoxph *Fit Cox Proportional Hazards Models with Linkage Error Adjustment*

Description

plcoxph fits Cox proportional hazards models to linked data, adjusting for potential mismatch errors. It serves as a wrapper around the internal fitcoxph function, for compatibility with the coxph syntax.

Usage

```
plcoxph(
  formula,
  adjustment,
  subset,
  na.action,
  model = TRUE,
  x = FALSE,
  y = FALSE,
  control = list(),
  ...
)
```

Arguments

formula	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the Surv function.
adjustment	An object inheriting from class "adjustment", or a list containing the necessary parameter specifications.
subset	An optional vector specifying a subset of observations.

na.action	A function for handling NAs.
model	Logical; if TRUE, the model frame is returned.
x, y	Logical; if TRUE, the model matrix (x) and response (y) are returned. Defaults are FALSE and FALSE.
control	A list of parameters for controlling the linkage error adjustment process.
...	Additional arguments passed to the internal function.

Value

An object representing the fitted model. The specific class and structure of the returned object depend directly on the adjustment method provided:

- If adjustment is of class adjELE, returns an object of class coxphELE.
- If adjustment is of class adjMixture, returns an object of class coxphMixture.

In all cases, the returned object is a list containing the original adjustment object under the adjustment element.

See Also

adjELE, adjMixture, coxphELE, coxphMixture

Examples

```
library(survival)
set.seed(101)
n <- 250

# Simulate true survival data
x <- rnorm(n)
true_hazard <- exp(0.5 * x)
true_time <- rexp(n, true_hazard)
true_status <- rbinom(n, 1, 0.8)

# Induce linkage mismatch errors
match_score <- rbeta(n, 5, 1)
is_mismatch <- rbinom(n, 1, 1 - match_score)

obs_time <- true_time
obs_status <- true_status
mismatch_idx <- which(is_mismatch == 1)

# Shuffle time and status together for mismatched records
shuffled_idx <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled_idx]
obs_status[mismatch_idx] <- obs_status[shuffled_idx]

linked_data <- data.frame(time = obs_time, status = obs_status, x = x, match_score)

# Specify the Adjustment Method
adj <- adjMixture(
```

```
  linked.data = linked_data,
  m.formula = ~ match_score
)

# Fit the Adjusted Cox Proportional Hazards Model
fit <- plcoxph(
  Surv(time, status) ~ x,
  adjustment = adj,
  control = list(max.iter = 50)
)
```

`plctable` *Analysis of Contingency Tables with Linkage Error Adjustment*

Description

`plctable` constructs a contingency table and adjusts the fitted model for mismatch errors.

Usage

```
plctable(
  formula,
  adjustment,
  subset,
  na.action,
  exclude = c(NA, NaN),
  control = list(),
  ...
)
```

Arguments

`formula`	a formula object with the left and right hand sides specifying the column and row variable of the flat table, respectively.
`adjustment`	An object inheriting from class `"adjustment"`, or a `list` containing the necessary parameter specifications.
`subset`	An optional vector specifying a subset of observations to be used.
`na.action`	A function which indicates what should happen when the data contain NAs.
`exclude`	Vector of values to be excluded when forming the table (passed to `xtabs`).
`control`	A list of parameters for controlling the linkage error adjustment process.
`...`	Additional arguments passed to the internal function.

Value

An object representing the fitted model. The specific class and structure of the returned object depend directly on the `adjustment` method provided:

- If `adjustment` is of class `adjMixture`, returns an object of class `ctableMixture`.

The returned object is a list containing the original `adjustment` object under the `adjustment` element.

See Also

`adjMixture`, `ctableMixture`

Examples

```
set.seed(102)
n <- 400

# Simulate true categorical data
exposure <- sample(c("low", "high"), n, replace = TRUE)
# True relationship: high exposure -> higher chance of disease
prob_disease <- ifelse(exposure == "high", 0.7, 0.3)
true_outcome <- ifelse(runif(n) < prob_disease, "disease", "healthy")

# Induce linkage (mismatch) errors at a fixed overall rate
true_mismatch_rate <- 0.20
is_mismatch <- rbinom(n, 1, true_mismatch_rate)

obs_outcome <- true_outcome
mismatch_idx <- which(is_mismatch == 1)
# Shuffle outcomes for the mismatched records
obs_outcome[mismatch_idx] <- sample(obs_outcome[mismatch_idx])

linked_df <- data.frame(exposure, outcome = obs_outcome)

# Specify the Adjustment Method
adj <- adjMixture(
  linked.data = linked_df,
  m.rate = true_mismatch_rate
)

# Fit the adjusted contingency table model
fit <- plctable(
  ~ exposure + outcome,
  adjustment = adj
)
```

plglm *Fit Generalized Linear Models with Linkage Error Adjustment*

Description

plglm fits generalized linear models (GLMs) to linked data, incorporating adjustments for linkage error as specified in the provided adjustment object. It mimics the interface of glm to ensure familiarity for users.

Usage

```
plglm(
  formula,
  family = gaussian,
  adjustment,
  subset,
  na.action,
  model = TRUE,
  x = FALSE,
  y = FALSE,
  control = list(),
  ...
)
```

Arguments

formula	A symbolic description of the model to be fitted.
family	A description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function.
adjustment	An object inheriting from class "adjustment" (e.g., adjELE, adjMixture), or a list containing the necessary data and parameter specifications.
subset	An optional vector specifying a subset of observations to be used in the fitting process.
na.action	A function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset.
model	Logical; if TRUE (default), the model frame is returned.
x, y	Logical; if TRUE, the model matrix (x) and response vector (y) are returned. Default is FALSE for both.
control	A list of parameters for controlling the linkage error adjustment process.
...	Additional arguments passed to the underlying fitting function.

Details

This function attempts to extract the linked data from the `adjustment` object. It supports both reference-based storage (via `data_ref`) and direct list components (`adjustment$data`). If the data is not present (e.g., NULL), the function will attempt to resolve variables from the environment of the `formula`.

It applies the standard `model.frame` processing steps (formula parsing, subsetting, NA handling) and dispatches the resulting design matrix and response vector to the appropriate `fitglm` method.

Value

An object representing the fitted model. The specific class and structure of the returned object depend directly on the `adjustment` method provided:

- If `adjustment` is of class `adjELE`, returns an object of class `glmELE`.
- If `adjustment` is of class `adjMixture`, returns an object of class `glmMixture`.
- If `adjustment` is of class `adjMixBayes`, returns an object of class `glmMixBayes`.

In all cases, the returned object is a list containing the original `adjustment` object under the `adjustment` element.

See Also

`adjELE`, `adjMixture`, `adjMixBayes`, `glmELE`, `glmMixture`, `glmMixBayes`

Examples

```
# Load the LIFE-M demo dataset
data(lifem)

# Phase 1: Adjustment Specification
# We model the correct match indicator via logistic regression using
# name commonness scores (commf, comml) and a 5% expected mismatch rate.
adj_object <- adjMixture(
 linked.data = lifem,
 m.formula = ~ commf + comml,
 m.rate = 0.05,
 safe.matches = hndlnk
)

# Phase 2: Estimation & Inference
# Fit a Gaussian regression model utilizing a cubic polynomial for year of birth.
fit <- plglm(
 age_at_death ~ poly(unit_yob, 3, raw = TRUE),
 family = "gaussian",
 adjustment = adj_object
)
```

plsurvreg *Fit Parametric Survival Models with Linkage Error Adjustment*

Description

plsurvreg fits parametric survival models (Accelerated Failure Time models) to linked data. It serves as a wrapper for the internal engines, compatible with survreg specifications.

Usage

```
plsurvreg(
  formula,
  adjustment,
  subset,
  na.action,
  dist = "weibull",
  model = TRUE,
  x = FALSE,
  y = FALSE,
  control = list(),
  ...
)
```

Arguments

formula	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the Surv function.
adjustment	An object inheriting from class "adjustment", or a list containing the necessary parameter specifications.
subset	An optional vector specifying a subset of observations.
na.action	A function for handling NAs.
dist	Character string specifying the survival distribution (currently it must be "weibull" or "gamma").
model	Logical; if TRUE, the model frame is returned.
x, y	Logical; if TRUE, the model matrix (x) and response (y) are returned.
control	A list of control parameters.
...	Additional arguments passed to the internal engine.

Value

An object representing the fitted model. The specific class and structure of the returned object depend directly on the adjustment method provided:

- If adjustment is of class adjMixBayes, returns an object of class survregMixBayes.

The returned object is a list containing the original `adjustment` object under the `adjustment` element.

See Also

`adjMixBayes`, `survregMixBayes`

Examples

```
library(survival)
set.seed(202)
n <- 200

# Simulate Weibull AFT data
trt <- rbinom(n, 1, 0.5)
true_time <- rweibull(n, shape = 1.5, scale = exp(1 + 0.8 * trt))
cens_time <- rexp(n, 0.05)
true_obs_time <- pmin(true_time, cens_time)
true_status <- as.numeric(true_time <= cens_time)

# Induce linkage mismatch errors by...
is_mismatch <- rbinom(n, 1, 0.2) # ~20% overall mismatch rate
obs_time <- true_obs_time
obs_status <- true_status
mismatch_idx <- which(is_mismatch == 1)

# Shuffle time and status together for mismatched records
shuffled <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled]
obs_status[mismatch_idx] <- obs_status[shuffled]

linked_df <- data.frame(time = obs_time, status = obs_status, trt)

# Specify the Bayesian Mixture Adjustment
adj <- adjMixBayes(linked.data = linked_df)

# Fit the Adjusted Parametric Survival Model
fit <- plsurvreg(
  Surv(time, status) ~ trt,
  dist = "weibull",
  adjustment = adj,
  control = list(iterations = 2000, burnin.iterations = 500)
)
```

`predict.coxphELE` *Predictions from a coxphELE Object*

Description

Computes linear predictors or risk scores from a fitted coxphELE model. If newdata is provided, the method attempts to construct the model matrix from the call stored in the object.

Usage

```
## S3 method for class 'coxphELE'
predict(object, newdata = NULL, type = c("lp", "risk"), ...)
```

Arguments

object	An object of class coxphELE.
newdata	Optional new data frame to obtain predictions for. If omitted, the linear predictors of the original data are returned.
type	The type of prediction required. Type "lp" (default) returns the linear predictor. Type "risk" returns the hazard ratio, exp(lp).
...	Additional arguments (currently ignored).

Details

If newdata is supplied, the function attempts to retrieve the formula from object$call or object$formula. If the original model was fitted using the internal engine directly without a wrapper that stores the call/formula, prediction on new data will fail.

Value

A numeric vector of predictions.

Examples

```
library(survival)
set.seed(105)

# Simulate linked data subject to mismatch error
n <- 200
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.7)
true_time <- rexp(n, rate = exp(0.5 * x1 - 0.5 * x2))
cens_time <- rexp(n, rate = 0.5)

# Induce 15% linkage error
mis_idx <- sample(1:n, size = 0.15 * n)
x1[mis_idx] <- x1[sample(mis_idx)]
x2[mis_idx] <- x2[sample(mis_idx)]

# Linked data
linked_data <- data.frame(
  time = pmin(true_time, cens_time),
  status = as.numeric(true_time <= cens_time),
  x1 = x1, x2 = x2
```

```
)

# Fit the adjusted Cox PH model
adj <- adjELE(linked.data = linked_data, m.rate = 0.15)
fit <- plcoxph(Surv(time, status) ~ x1 + x2, adjustment = adj)

# 1. Extract linear predictors for the original data
lp_train <- predict(fit, type = "lp")
head(lp_train)

# 2. Predict hazard ratios (risk) for a new cohort
new_cohort <- data.frame(x1 = c(0, 1.5, -1), x2 = c(0, 1, 1))
risk_scores <- predict(fit, newdata = new_cohort, type = "risk")
print(risk_scores)
```

`predict.coxphMixture` *Model Predictions for coxphMixture Objects*

Description

Compute fitted values and predictions for the outcome model component of the mixture. The predictions are conditional on the latent status being a "correct match".

Usage

```
## S3 method for class 'coxphMixture'
predict(
  object,
  newdata,
  type = c("lp", "risk", "expected", "survival"),
  se.fit = FALSE,
  na.action = stats::na.pass,
  reference = "strata",
  ...
)
```

Arguments

`object`	An object of class `coxphMixture`.
`newdata`	Optional new data frame. If missing, predictions are for the original data.
`type`	The type of prediction. • `"lp"`: Linear predictor (eta = X * beta). • `"risk"`: Risk score (exp(eta)). • `"expected"`: Expected number of events (approximate). • `"survival"`: Survival probability at the observed times.

se.fit	Logical; whether to compute standard errors (based on the sandwich/Louis variance).
na.action	Function to handle missing values in newdata.
reference	Reference for centering (currently ignored, defaults to uncentered).
...	Additional arguments passed to methods.

Details

When newdata is supplied, the function constructs the model matrix using the terms from the original fit. Standard errors are computed using the estimated variance-covariance matrix of the mixture model coefficients.

For type = "expected" and "survival", the function reconstructs the cumulative baseline hazard step function $\Lambda_0(t)$ using the Breslow estimator stored in the object and evaluates it at the time points found in newdata.

Value

A vector or matrix of predictions, or a list containing fit and se.fit if standard errors are requested.

Examples

```
library(survival)
set.seed(205)

# Simulate auxiliary match scores and linkage errors
# Lower match scores correspond to a higher probability of mismatch
n <- 200
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.5)
true_time <- rexp(n, rate = exp(0.5 * x1 - 0.5 * x2))
cens_time <- rexp(n, rate = 0.5)
match_score <- runif(n, 0.5, 1.0)

mis_idx <- which(rbinom(n, 1, prob = 1 - match_score) == 1)
x1[mis_idx] <- x1[sample(mis_idx)]
x2[mis_idx] <- x2[sample(mis_idx)]

linked_data <- data.frame(
  time = pmin(true_time, cens_time),
  status = as.numeric(true_time <= cens_time),
  x1 = x1, x2 = x2, match_score = match_score
)

# Fit the Cox PH Mixture Model
# Note: We set `y = TRUE` to store the response for baseline hazard reconstruction
adj <- adjMixture(linked.data = linked_data, m.formula = ~ match_score)
fit <- plcoxph(Surv(time, status) ~ x1 + x2, adjustment = adj,
               y = TRUE, control = list(max.iter = 15))
```

```
# 1. Extract linear predictors for the original data
lp_train <- predict(fit, type = "lp")
head(lp_train)

# 2. Predict hazard ratios (risk) and expected events for a new cohort
new_cohort <- data.frame(
  time = c(1.0, 2.5, 0.5), # Required for expected/survival types
  x1 = c(0, 1.5, -1),
  x2 = c(0, 1, 1)
)

# Predict risk (exp(lp))
risk_scores <- predict(fit, newdata = new_cohort, type = "risk")
print(risk_scores)

# Predict expected number of events based on the reconstructed baseline hazard
exp_events <- predict(fit, newdata = new_cohort, type = "expected")
print(exp_events)
```

`predict.glmELE` *Predictions for* `glmELE` *Objects*

Description

Obtains predictions and optionally standard errors from a fitted `glmELE` object. This method handles new data, factor level consistency, offsets, and confidence intervals.

Usage

```
## S3 method for class 'glmELE'
predict(
  object,
  newdata = NULL,
  weight.matrix = NULL,
  type = c("link", "response"),
  se.fit = FALSE,
  interval = c("none", "confidence"),
  level = 0.95,
  na.action = stats::na.pass,
  ...
)
```

Arguments

`object`	An object of class `"glmELE"`.
`newdata`	An optional data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors from the object are used.

weight.matrix	Character string specifying the weighting method to use (e.g., "ratio", "LL", "BLUE"). Defaults to the first method found in the object.
type	The type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable.
se.fit	Logical; if TRUE, standard errors are returned.
interval	Type of interval calculation. Can be "none" or "confidence".
level	The confidence level required (default 0.95).
na.action	A function determining what should be done with missing values in newdata. The default is to predict NA.
...	Additional arguments passed to methods.

Value

If se.fit = FALSE and interval = "none", a vector of predictions. Otherwise, a list containing:

fit	Predictions (or a matrix with columns fit, lwr, upr if intervals are requested).
se.fit	Estimated standard errors (if requested).
residual.scale	The dispersion parameter used.

Examples

```
data(brfss, package = "postlink")

adj_object <- adjELE(linked.data = brfss,
                     m.rate = unique(brfss$m.rate),
                     blocks = imonth,
                     weight.matrix = "BLUE")

fit <- plglm(Weight ~ Height + Physhlth + Menthlth + Exerany,
             family = "gaussian", adjustment = adj_object)

predict(fit)
```

---

predict.glmMixBayes *Predictions from a glmMixBayes Model*

Description

Predictions from a glmMixBayes Model

Usage

```
## S3 method for class 'glmMixBayes'
predict(
  object,
  newx,
  type = c("link", "response"),
  se.fit = FALSE,
  interval = c("none", "credible"),
  level = 0.95,
  ...
)
```

Arguments

object	A glmMixBayes model object.
newx	A numeric matrix of new observations (n_new x K) with columns aligned to the design matrix X used for fitting.
type	Either "link" or "response", indicating the scale of predictions.
se.fit	Logical; if TRUE, also return posterior SD of predictions.
interval	Either "none" or "credible", indicating whether to compute a credible interval.
level	Probability level for the credible interval (default 0.95).
...	Not used.

Value

If se.fit = FALSE and interval = "none", a numeric vector of predicted values. Otherwise, a matrix with columns for the fit, (optional) se.fit, and (optional) credible interval bounds.

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

x <- cbind(1, poly(lifem_small$unit_yob, 3, raw = TRUE))
y <- lifem_small$age_at_death

adj <- adjMixBayes(
  linked.data = lifem_small,
  priors = list(theta = "beta(2, 2)")
)
```

```
fit <- plglm(
  age_at_death ~ poly(unit_yob, 3, raw = TRUE),
  family = "gaussian",
  adjustment = adj,
  control = list(
    iterations = 200,
    burnin.iterations = 100,
    seed = 123
  )
)

newx <- cbind(1, poly(c(0.2, 0.5, 0.8), 3, raw = TRUE))
predict(fit, newx = newx, type = "response")
```

`predict.glmMixture` *Model Predictions for glmMixture Objects*

Description

Obtains predictions and optionally estimates standard errors of those predictions from a fitted `glmMixture` object.

Usage

```
## S3 method for class 'glmMixture'
predict(
  object,
  newdata = NULL,
  type = c("link", "response"),
  se.fit = FALSE,
  interval = c("none", "confidence"),
  level = 0.95,
  na.action = stats::na.pass,
  ...
)
```

Arguments

`object`	An object of class `glmMixture`.
`newdata`	An optional data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
`type`	The type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable.
`se.fit`	Logical switch indicating if standard errors are required.

interval	Type of interval calculation ("none" or "confidence").
level	Confidence level.
na.action	Function determining what should be done with missing values in newdata. The default is to predict NA.
...	Additional arguments (currently ignored).

Details

If newdata is omitted, the predictions are based on the data used for the fit. In that case, type = "link" corresponds to object$linear.predictors and type = "response" corresponds to object$fitted.values. If newdata is supplied, the function manually constructs the design matrix from the terms object stored in the model. Standard errors are computed using the sandwich covariance matrix (object$var).

Value

If se.fit = FALSE, a vector of predictions. If se.fit = TRUE, a list with components:

fit	Predictions.
se.fit	Estimated standard errors.
residual.scale	A scalar giving the square root of the dispersion used in computing the standard errors.

Examples

```
# Load the LIFE-M demo dataset
data(lifem)

# Phase 1: Adjustment Specification
# We model the correct match indicator via logistic regression using
# name commonness scores (commf, comml) and a 5% expected mismatch rate.
adj_object <- adjMixture(
 linked.data = lifem,
 m.formula = ~ commf + comml,
 m.rate = 0.05,
 safe.matches = hndlnk
)

# Phase 2: Estimation & Inference
# Fit a Gaussian regression model utilizing a cubic polynomial for year of birth.
fit <- plglm(
 age_at_death ~ poly(unit_yob, 3, raw = TRUE),
 family = "gaussian",
 adjustment = adj_object
)

predict(fit)
```

`predict.survMixBayes` *Predictions from a survMixBayes Model*

Description

Computes posterior predictions for each latent component of a `survMixBayes` model. By default, predictions are returned on the linear predictor scale for both components.

Usage

```
## S3 method for class 'survMixBayes'
predict(
  object,
  newdata = NULL,
  se.fit = FALSE,
  interval = c("none", "credible"),
  level = 0.95,
  ...
)
```

Arguments

`object`	A `survMixBayes` model object.
`newdata`	A numeric matrix of new observations ($n_{new} \times K$) with columns aligned to the design matrix used for fitting. If `NULL`, the fitted design matrix stored in `object$X` is used.
`se.fit`	Logical; if `TRUE`, also return posterior SD of predictions.
`interval`	Either `"none"` or `"credible"`, indicating whether to compute credible intervals.
`level`	Probability level for the credible interval (default 0.95).
`...`	Not used.

Details

Component 1 is interpreted as the correct-match component and component 2 as the incorrect-match component (after label-switching correction).

Value

A list with two components, `component1` and `component2`, corresponding to the two latent mixture components. If `se.fit = FALSE` and `interval = "none"`, each element is a numeric vector of posterior mean linear predictors. Otherwise, each element is a matrix containing the fitted values and, optionally, posterior SDs and credible interval bounds.

Examples

```
set.seed(301)
n <- 150
trt <- rbinom(n, 1, 0.5)

# Simulate Weibull AFT data
true_time <- rweibull(n, shape = 1.5, scale = exp(1 + 0.8 * trt))
cens_time <- rexp(n, rate = 0.1)
true_obs_time <- pmin(true_time, cens_time)
true_status <- as.integer(true_time <= cens_time)

# Induce linkage mismatch errors in approximately 20% of records
is_mismatch <- rbinom(n, 1, 0.2)
obs_time <- true_obs_time
obs_status <- true_status
mismatch_idx <- which(is_mismatch == 1)

shuffled <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled]
obs_status[mismatch_idx] <- obs_status[shuffled]

linked_df <- data.frame(time = obs_time, status = obs_status, trt = trt)
adj <- adjMixBayes(linked.data = linked_df)

fit <- plsurvreg(
  survival::Surv(time, status) ~ trt,
  dist = "weibull",
  adjustment = adj,
  control = list(
    iterations = 200,
    burnin.iterations = 100,
    seed = 123
  )
)

# Create a design matrix for new covariate values
newdata <- stats::model.matrix(~ trt, data = data.frame(trt = c(0, 1)))

# Predict posterior mean linear predictors for each latent component
preds <- predict(fit, newdata = newdata, se.fit = TRUE, interval = "credible")
print(preds$component1)
print(preds$component2)
```

`print.adjELE` *Print Method for* `adjELE` *Objects*

Description

Provides a concise summary of the adjustment object created by adjELE, including linkage error assumptions, blocking structure, and weight estimation settings.

Usage

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

Arguments

x	An object of class adjELE.
digits	Integer; the number of significant digits to use when printing numeric values. Defaults to 3.
...	Additional arguments passed to methods.

Details

This method inspects the internal structure of the adjustment object. It calculates summaries for mismatch rates and audit sizes (e.g., means/ranges) if they vary across blocks, providing a snapshot of the error assumption complexity. It safely handles cases where the reference data is missing or empty.

Value

Invisibly returns the input object x.

Examples

```
data(brfss, package = "postlink")

adj_object <- adjELE(linked.data = brfss,
                     m.rate = unique(brfss$m.rate),
                     blocks = imonth,
                     weight.matrix = "BLUE")
print(adj_object)
```

---

print.adjMixBayes *Print Method for* adjMixBayes *Objects*

Description

Provides a concise summary of the Bayesian adjustment object created by adjMixBayes.

Usage

```
## S3 method for class 'adjMixBayes'
print(x, ...)
```

Arguments

x	An object of class adjMixBayes.
...	Additional arguments passed to methods.

Details

This method inspects the reference-based data environment to report the number of linked records without copying the full dataset. It safely handles cases where the linked data is unspecified (NULL). It also prints the user-specified priors or outlines the defaults that will be used.

Value

Invisibly returns x.

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

adj_obj <- adjMixBayes(
  linked.data = lifem_small,
  priors = list(theta = "beta(2, 2)")
)

# Implicitly calls print.adjMixBayes()
adj_obj
```

---

print.adjMixture *Print Method for* adjMixture *Objects*

Description

Provides a concise summary of the adjustment object created by adjMixture, including dataset dimensions, model specifications, and constraints.

Usage

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

Arguments

x	An object of class adjMixture.
digits	Integer; the number of significant digits to use when printing numeric values (e.g., mismatch rates). Defaults to 3.
...	Additional arguments passed to methods.

Details

This method inspects the reference-based data environment to report the number of linked records without copying the full dataset. It considers cases where components (like constraints or safe matches) are unspecified.

Value

Invisibly returns the input object x.

Examples

```
# Load the LIFE-M demo dataset
data(lifem)

# Phase 1: Adjustment Specification
adj_object <- adjMixture(
 linked.data = lifem,
 m.formula = ~ commf + comml,
 m.rate = 0.05,
 safe.matches = hndlnk
)

# Print specified adjustment
print(adj_object)
```

---

print.adjustment *Print Base Method for Adjustment Objects*

Description

Print Base Method for Adjustment Objects

Usage

```
## S3 method for class 'adjustment'
print(x, ...)
```

Arguments

x	An object of class "adjustment".
...	Additional arguments.

---

print.coxphELE *Print a coxphELE Object*

Description

Prints a short summary of the fitted CoxPH model with linkage error adjustment, displaying the call (if available) and the estimated coefficients.

Usage

```
## S3 method for class 'coxphELE'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object of class coxphELE.
digits	The number of significant digits to use when printing.
...	Additional arguments passed to print.default.

Value

Invisibly returns the input object x.

Examples

```
library(survival)
set.seed(104)
n <- 200

# 1. Simulate covariates
age_centered <- rnorm(n, 0, 5)
treatment <- rbinom(n, 1, 0.5)

# 2. Simulate true survival times
true_time <- rexp(n, rate = exp(0.05 * age_centered - 0.6 * treatment))
cens_time <- rexp(n, rate = 0.2)
time <- pmin(true_time, cens_time)
status <- as.numeric(true_time <= cens_time)

# 3. Induce 15% Exchangeable Linkage Error (ELE)
mis_idx <- sample(1:n, size = floor(0.15 * n))
linked_age <- age_centered
linked_trt <- treatment
```

```
# False links drawn uniformly from the target population
false_link_idx <- sample(1:n, size = length(mis_idx), replace = TRUE)
linked_age[mis_idx] <- age_centered[false_link_idx]
linked_trt[mis_idx] <- treatment[false_link_idx]

linked_data <- data.frame(time = time, status = status,
                          age = linked_age, treatment = linked_trt)

# 4. Fit the adjusted Cox PH model
adj <- adjELE(linked.data = linked_data, m.rate = 0.15)
fit <- plcoxph(Surv(time, status) ~ age + treatment, adjustment = adj)

# 5. Print the basic model output
print(fit)
```

`print.coxphMixture` *Print a coxphMixture Object*

Description

Print a coxphMixture Object

Usage

```
## S3 method for class 'coxphMixture'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	An object of class `coxphMixture`.
`digits`	The number of significant digits to use.
`...`	Additional arguments.

Value

Invisibly returns the input object `x`.

Examples

```
library(survival)
set.seed(201)

# Simulate survival data (N = 200)
n <- 200
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.5)
true_time <- rexp(n, rate = exp(0.5 * x1 - 0.5 * x2))
cens_time <- rexp(n, rate = 0.5)
```

```
# Simulate auxiliary match scores and linkage errors
# Lower match scores correspond to a higher probability of mismatch
match_score <- runif(n, 0.5, 1.0)
is_mismatch <- rbinom(n, 1, prob = 1 - match_score)

# Induce linkage errors by shuffling covariates of mismatched records
linked_x1 <- x1
linked_x2 <- x2
mis_idx <- which(is_mismatch == 1)
shuffled_idx <- sample(mis_idx)
linked_x1[mis_idx] <- x1[shuffled_idx]
linked_x2[mis_idx] <- x2[shuffled_idx]

linked_data <- data.frame(
  time = pmin(true_time, cens_time),
  status = as.numeric(true_time <= cens_time),
  x1 = linked_x1, x2 = linked_x2,
  match_score = match_score
)

# Fit the Cox PH Mixture Model (Slawski et al., 2023)
adj <- adjMixture(linked.data = linked_data, m.formula = ~ match_score)
fit <- plcoxph(Surv(time, status) ~ x1 + x2, adjustment = adj,
               control = list(max.iter = 15))

# Explicitly call the print method
print(fit)
```

`print.ctableMixture` *Print Method for Adjusted Contingency Tables*

Description

Prints the estimated contingency table (corrected for linkage error) and a summary of the adjustment parameters used by the mixture model.

Usage

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

Arguments

`x`	An object of class `ctableMixture`.
`digits`	Integer; the number of significant digits to use when printing numeric values. Defaults to 3.
`...`	Additional arguments passed to `print.default`.

Value

The argument x, invisibly.

See Also

plctable, ctableMixture

Examples

```
set.seed(125)
n <- 300

# 1. Simulate true categorical data with dependency
exposure <- sample(c("low", "high"), n, replace = TRUE)

# Induce dependency - High exposure -> higher disease probability
prob_disease <- ifelse(exposure == "high", 0.7, 0.3)
true_disease <- ifelse(runif(n) < prob_disease, "yes", "no")

# 2. Induce 15% linkage error
mis_idx <- sample(1:n, size = floor(0.15 * n))
obs_disease <- true_disease
obs_disease[mis_idx] <- sample(obs_disease[mis_idx])

linked_df <- data.frame(exposure = exposure, disease = obs_disease)

# 3. Fit the adjusted contingency table model
adj <- adjMixture(linked.data = linked_df, m.rate = 0.15)
fit <- plctable(~ exposure + disease, adjustment = adj)

# 4. Explicitly call the print method
print(fit)
```

---

print.glmELE *Print a* glmELE *Object*

Description

Prints the function call and the estimated coefficient matrices from a fitted glmELE object.

Usage

```
## S3 method for class 'glmELE'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object of class "glmELE".
digits	The number of significant digits to print. Defaults to max(3L, getOption("digits") - 3L).
...	Additional arguments passed to methods.

Value

Invisibly returns the input object x.

Examples

```
data(brfss, package = "postlink")

adj_object <- adjELE(linked.data = brfss,
                     m.rate = unique(brfss$m.rate),
                     blocks = imonth,
                     weight.matrix = "BLUE")

fit <- plglm(Weight ~ Height + Physhlth + Menthlth + Exerany,
             family = "gaussian", adjustment = adj_object)

print(fit)
```

---

print.glmMixBayes *Print a glmMixBayes Model Object*

Description

Print a glmMixBayes Model Object

Usage

```
## S3 method for class 'glmMixBayes'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object of class glmMixBayes.
digits	Minimum number of significant digits to show.
...	Further arguments (unused).

Value

The input x, invisibly.

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

x <- cbind(1, poly(lifem_small$unit_yob, 3, raw = TRUE))
y <- lifem_small$age_at_death

adj <- adjMixBayes(
  linked.data = lifem_small,
  priors = list(theta = "beta(2, 2)")
)

fit <- plglm(
  age_at_death ~ poly(unit_yob, 3, raw = TRUE),
  family = "gaussian",
  adjustment = adj,
  control = list(
    iterations = 200,
    burnin.iterations = 100,
    seed = 123
  )
)

print(fit)
```

`print.glmMixture` *Print a glmMixture Object*

Description

Print a glmMixture Object

Usage

```
## S3 method for class 'glmMixture'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	An object of class `glmMixture`.

digits	The number of significant digits to use.
...	Additional arguments.

Value

Invisibly returns the input object x.

Examples

```
# Load the LIFE-M demo dataset
data(lifem)

# Phase 1: Adjustment Specification
# We model the correct match indicator via logistic regression using
# name commonness scores (commf, comml) and a 5% expected mismatch rate.
adj_object <- adjMixture(
 linked.data = lifem,
 m.formula = ~ commf + comml,
 m.rate = 0.05,
 safe.matches = hndlnk
)

# Phase 2: Estimation & Inference
# Fit a Gaussian regression model utilizing a cubic polynomial for year of birth.
fit <- plglm(
 age_at_death ~ poly(unit_yob, 3, raw = TRUE),
 family = "gaussian",
 adjustment = adj_object
)

print(fit)
```

---

print.mi_link_pool_glm

Print Pooled Regression Results

Description

Print Pooled Regression Results

Usage

```
## S3 method for class 'mi_link_pool_glm'
print(x, digits = max(3L, getOption("digits") - 2L), ...)
```

Arguments

x	An object of class mi_link_pool_glm, typically returned by mi_with() for a glmMixBayes fit.
digits	the number of significant digits to print.
...	further arguments (unused).

Value

The input x, invisibly.

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

x <- cbind(1, poly(lifem_small$unit_yob, 3, raw = TRUE))
y <- lifem_small$age_at_death

adj <- adjMixBayes(
  linked.data = lifem_small,
  priors = list(theta = "beta(2, 2)")
)

fit <- plglm(
  age_at_death ~ poly(unit_yob, 3, raw = TRUE),
  family = "gaussian",
  adjustment = adj,
  control = list(
    iterations = 200,
    burnin.iterations = 100,
    seed = 123
  )
)

pooled_fit <- mi_with(
  object = fit,
  data = lifem_small,
  formula = age_at_death ~ poly(unit_yob, 3, raw = TRUE),
  family = gaussian()
)

print(pooled_fit, digits = 4)
```

print.mi_link_pool_survreg *Print Pooled Cox Regression Results*

Description

Print Pooled Cox Regression Results

Usage

```
## S3 method for class 'mi_link_pool_survreg'
print(x, digits = max(3L, getOption("digits") - 2L), ...)
```

Arguments

x	An object of class mi_link_pool_survreg, typically returned by mi_with() for a survMixBayes fit.
digits	the number of significant digits to print.
...	further arguments (unused).

Value

The input x, invisibly.

Examples

```
set.seed(301)
n <- 150
trt <- rbinom(n, 1, 0.5)

# Simulate Weibull AFT data
true_time <- rweibull(n, shape = 1.5, scale = exp(1 + 0.8 * trt))
cens_time <- rexp(n, rate = 0.1)
true_obs_time <- pmin(true_time, cens_time)
true_status <- as.integer(true_time <= cens_time)

# Induce linkage mismatch errors in approximately 20% of records
is_mismatch <- rbinom(n, 1, 0.2)
obs_time <- true_obs_time
obs_status <- true_status
mismatch_idx <- which(is_mismatch == 1)

shuffled <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled]
obs_status[mismatch_idx] <- obs_status[shuffled]

linked_df <- data.frame(time = obs_time, status = obs_status, trt = trt)
adj <- adjMixBayes(linked.data = linked_df)
```

```
fit <- plsurvreg(
  survival::Surv(time, status) ~ trt,
  dist = "weibull",
  adjustment = adj,
  control = list(iterations = 200, burnin.iterations = 100, seed = 123)
)

pooled_obj <- mi_with(
  object = fit,
  data = linked_df,
  formula = survival::Surv(time, status) ~ trt
)

print(pooled_obj, digits = 4)
```

`print.survMixBayes` *Print a survMixBayes Model Object*

Description

Prints the model call and posterior mean regression coefficients for the first mixture component of the fitted survival model. In this package, component 1 is interpreted as the correct-match component and component 2 as the incorrect-match component.

Usage

```
## S3 method for class 'survMixBayes'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	An object of class `survMixBayes`.
`digits`	Minimum number of significant digits to show.
`...`	Further arguments (unused).

Value

The input `x`, invisibly.

Examples

```
set.seed(301)
n <- 150
trt <- rbinom(n, 1, 0.5)

# Simulate Weibull AFT data
```

```
true_time <- rweibull(n, shape = 1.5, scale = exp(1 + 0.8 * trt))
cens_time <- rexp(n, rate = 0.1)
true_obs_time <- pmin(true_time, cens_time)
true_status <- as.integer(true_time <= cens_time)

# Induce linkage mismatch errors in approximately 20% of records
is_mismatch <- rbinom(n, 1, 0.2)
obs_time <- true_obs_time
obs_status <- true_status
mismatch_idx <- which(is_mismatch == 1)

shuffled <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled]
obs_status[mismatch_idx] <- obs_status[shuffled]

linked_df <- data.frame(time = obs_time, status = obs_status, trt = trt)
adj <- adjMixBayes(linked.data = linked_df)

fit <- plsurvreg(
  survival::Surv(time, status) ~ trt,
  dist = "weibull",
  adjustment = adj,
  control = list(iterations = 200, burnin.iterations = 100, seed = 123)
)

print(fit)
```

`summary.coxphELE` *Summary of a coxphELE Object*

Description

Produces a summary of the fitted CoxPH model with linkage error adjustment, including coefficient estimates, hazard ratios, standard errors, z-statistics, and p-values.

Usage

```
## S3 method for class 'coxphELE'
summary(object, conf.int = 0.95, ...)
```

Arguments

`object`	An object of class `coxphELE`.
`conf.int`	The confidence level for the confidence intervals (default is 0.95).
`...`	Additional arguments (currently ignored).

Value

An object of class `summary.coxphELE` containing:

`call`	The matched call.
`coefficients`	A matrix with columns for coefficients, hazard ratios (exp(coef)), standard errors, z-values, and p-values.
`conf.int`	A matrix of confidence intervals for the hazard ratios.
`n`	The number of observations.
`nevent`	The number of events.

Examples

```
library(survival)
set.seed(104)
n <- 200

# 1. Simulate covariates
age_centered <- rnorm(n, 0, 5)
treatment <- rbinom(n, 1, 0.5)

# 2. Simulate true survival times
true_time <- rexp(n, rate = exp(0.05 * age_centered - 0.6 * treatment))
cens_time <- rexp(n, rate = 0.2)
time <- pmin(true_time, cens_time)
status <- as.numeric(true_time <= cens_time)

# 3. Induce 15% Exchangeable Linkage Error (ELE)
mis_idx <- sample(1:n, size = floor(0.15 * n))
linked_age <- age_centered
linked_trt <- treatment

# False links drawn uniformly from the target population
false_link_idx <- sample(1:n, size = length(mis_idx), replace = TRUE)
linked_age[mis_idx] <- age_centered[false_link_idx]
linked_trt[mis_idx] <- treatment[false_link_idx]

linked_data <- data.frame(time = time, status = status,
                          age = linked_age, treatment = linked_trt)

# 4. Fit the adjusted Cox PH model
adj <- adjELE(linked.data = linked_data, m.rate = 0.15)
fit <- plcoxph(Surv(time, status) ~ age + treatment, adjustment = adj)

# 5. Generate and print the detailed statistical summary
sum_fit <- summary(fit)
print(sum_fit)
```

summary.coxphMixture *Summarizing Cox PH Mixture Fits*

Description

summary method for class coxphMixture. Provides a detailed summary of the fitted model, including coefficients, hazard ratios, standard errors, z-statistics, and p-values for both the outcome model and the mismatch model.

Usage

```
## S3 method for class 'coxphMixture'
summary(object, conf.int = 0.95, scale = 1, ...)
```

Arguments

object	An object of class coxphMixture.
conf.int	The confidence level for the confidence intervals of the hazard ratios.
scale	Scale factor for the standard errors (default is 1).
...	Additional arguments.

Value

An object of class summary.coxphMixture containing:

call	The function call.
n	Total number of observations.
nevent	Number of events.
coefficients	Matrix of coefficients for the outcome model.
m.coefficients	Matrix of coefficients for the mismatch model.
conf.int	Matrix of confidence intervals for the hazard ratios.
logtest	Log-likelihood information (Outcome Model).
avgcmr	The average posterior probability of a correct match.

Examples

```
library(survival)
set.seed(201)

# Simulate survival data (N = 200)
n <- 200
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.5)
true_time <- rexp(n, rate = exp(0.5 * x1 - 0.5 * x2))
cens_time <- rexp(n, rate = 0.5)
```

```
# Simulate auxiliary match scores and linkage errors
# Lower match scores correspond to a higher probability of mismatch
match_score <- runif(n, 0.5, 1.0)
is_mismatch <- rbinom(n, 1, prob = 1 - match_score)

# Induce linkage errors by shuffling covariates of mismatched records
linked_x1 <- x1
linked_x2 <- x2
mis_idx <- which(is_mismatch == 1)
shuffled_idx <- sample(mis_idx)
linked_x1[mis_idx] <- x1[shuffled_idx]
linked_x2[mis_idx] <- x2[shuffled_idx]

linked_data <- data.frame(
  time = pmin(true_time, cens_time),
  status = as.numeric(true_time <= cens_time),
  x1 = linked_x1, x2 = linked_x2,
  match_score = match_score
)

# Fit the Cox PH Mixture Model (Slawski et al., 2023)
adj <- adjMixture(linked.data = linked_data, m.formula = ~ match_score)
fit <- plcoxph(Surv(time, status) ~ x1 + x2, adjustment = adj,
               control = list(max.iter = 15))

# Print detailed statistical summary
sum_fit <- summary(fit)
print(sum_fit)
```

`summary.ctableMixture` *Summary Method for Adjusted Contingency Tables*

Description

Provides a detailed summary of the `ctableMixture` model fit, including the estimated cell probabilities with standard errors, convergence status, and a Chi-squared test of independence performed on the adjusted counts.

Usage

```
## S3 method for class 'ctableMixture'
summary(object, ...)
```

Arguments

`object`	An object of class `ctableMixture`.
`...`	Additional arguments (currently ignored).

Value

An object of class `summary.ctableMixture` containing:

`call`	The function call.
`m.rate`	The assumed mismatch rate.
`ftable`	The estimated contingency table of correctly matched counts.
`coefficients`	A matrix containing estimates, standard errors, z-values, and p-values for cell probabilities.
`chisq`	The result of a Pearson's Chi-squared test on the adjusted table.
`converged`	Logical indicating if the EM algorithm converged.
`iterations`	Number of iterations performed.

Examples

```
set.seed(125)
n <- 300

# 1. Simulate true categorical data with dependency
exposure <- sample(c("low", "high"), n, replace = TRUE)

# Induce dependency - High exposure -> higher disease probability
prob_disease <- ifelse(exposure == "high", 0.7, 0.3)
true_disease <- ifelse(runif(n) < prob_disease, "yes", "no")

# 2. Induce 15% linkage error
mis_idx <- sample(1:n, size = floor(0.15 * n))
obs_disease <- true_disease
obs_disease[mis_idx] <- sample(obs_disease[mis_idx])

linked_df <- data.frame(exposure = exposure, disease = obs_disease)

# 3. Fit the adjusted contingency table model
adj <- adjMixture(linked.data = linked_df, m.rate = 0.15)
fit <- plctable(~ exposure + disease, adjustment = adj)

# 4. Generate the detailed summary object
sum_fit <- summary(fit)

# 5. Access specific components of the summary
print(sum_fit$coefficients)
print(sum_fit$chisq)
```

summary.glmELE *Summarize a* glmELE *Object*

Description

Summarizes the results from a glmELE fit, providing coefficient estimates, standard errors, test statistics, and p-values for each weighting method used.

Usage

```
## S3 method for class 'glmELE'
summary(object, ...)
```

Arguments

object	An object of class "glmELE".
...	Additional arguments passed to methods.

Value

An object of class "summary.glmELE", which is a list containing:

call	The matched call.
family	The family object used.
coefficients	A list of matrices, one per weighting method, containing estimates, SEs, t/z values, and p-values.
dispersion	The estimated dispersion parameter(s).
deviance	The deviance of the fitted model.
df.residual	The residual degrees of freedom.

Examples

```
data(brfss, package = "postlink")

adj_object <- adjELE(linked.data = brfss,
                     m.rate = unique(brfss$m.rate),
                     blocks = imonth,
                     weight.matrix = "BLUE")

fit <- plglm(Weight ~ Height + Physhlth + Menthlth + Exerany,
             family = "gaussian", adjustment = adj_object)

summary(fit)
```

summary.glmMixBayes *Summary Method for glmMixBayes Models*

Description

Summary Method for glmMixBayes Models

Usage

```
## S3 method for class 'glmMixBayes'
summary(object, ...)
```

Arguments

object	An object of class glmMixBayes.
...	Not used.

Value

An object of class "summary.glmMixBayes", which is printed with a custom method.

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

x <- cbind(1, poly(lifem_small$unit_yob, 3, raw = TRUE))
y <- lifem_small$age_at_death

adj <- adjMixBayes(
  linked.data = lifem_small,
  priors = list(theta = "beta(2, 2)")
)

fit <- plglm(
  age_at_death ~ poly(unit_yob, 3, raw = TRUE),
  family = "gaussian",
  adjustment = adj,
  control = list(
    iterations = 200,
    burnin.iterations = 100,
    seed = 123
```

```
  )
)

summary(fit)
```

---

summary.glmMixture *Summarizing GLM Mixture Fits*

Description

summary method for class glmMixture.

Usage

```
## S3 method for class 'glmMixture'
summary(object, dispersion = NULL, ...)
```

Arguments

object	An object of class glmMixture.
dispersion	The dispersion parameter for the family used. If NULL, it is inferred from object.
...	Additional arguments.

Value

An object of class summary.glmMixture containing:

call	The component from object.
family	The component from object.
df.residual	The residual degrees of freedom.
coefficients	Matrix of coefficients for the outcome model.
m.coefficients	Matrix of coefficients for the mismatch model.
dispersion	Estimated dispersion parameter.
cov.unscaled	The estimated covariance matrix.
match.prob	The posterior match probabilities.

Examples

```
# Load the LIFE-M demo dataset
data(lifem)

# Phase 1: Adjustment Specification
# We model the correct match indicator via logistic regression using
# name commonness scores (commf, comml) and a 5% expected mismatch rate.
adj_object <- adjMixture(
 linked.data = lifem,
 m.formula = ~ commf + comml,
 m.rate = 0.05,
 safe.matches = hndlnk
)

# Phase 2: Estimation & Inference
# Fit a Gaussian regression model utilizing a cubic polynomial for year of birth.
fit <- plglm(
 age_at_death ~ poly(unit_yob, 3, raw = TRUE),
 family = "gaussian",
 adjustment = adj_object
)

summary(fit)
```

`summary.survMixBayes` *Summary Method for survMixBayes Models*

Description

Computes posterior summaries for the regression coefficients, mixing weight, and component-specific distribution parameters in a fitted `survMixBayes` model. Throughout, component 1 is interpreted as the correct-match component and component 2 as the incorrect-match component.

Usage

```
## S3 method for class 'survMixBayes'
summary(object, probs = c(0.025, 0.5, 0.975), ...)
```

Arguments

`object`	An object of class `survMixBayes`.
`probs`	Numeric vector of probabilities used to compute posterior quantiles for the model parameters. The default, `c(0.025, 0.5, 0.975)`, gives a posterior median and a 95\ credible interval.
`...`	Further arguments (unused).

Value

An object of class `summary.survMixBayes` containing posterior quantile summaries for the regression coefficients in both mixture components, the mixing weight, and any family-specific distribution parameters included in the fitted model. Component 1 corresponds to the correct-match component and component 2 to the incorrect-match component.

Examples

```
set.seed(301)
n <- 150
trt <- rbinom(n, 1, 0.5)

# Simulate Weibull AFT data
true_time <- rweibull(n, shape = 1.5, scale = exp(1 + 0.8 * trt))
cens_time <- rexp(n, rate = 0.1)
true_obs_time <- pmin(true_time, cens_time)
true_status <- as.integer(true_time <= cens_time)

# Induce linkage mismatch errors in approximately 20% of records
is_mismatch <- rbinom(n, 1, 0.2)
obs_time <- true_obs_time
obs_status <- true_status
mismatch_idx <- which(is_mismatch == 1)

shuffled <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled]
obs_status[mismatch_idx] <- obs_status[shuffled]

linked_df <- data.frame(time = obs_time, status = obs_status, trt = trt)
adj <- adjMixBayes(linked.data = linked_df)

fit <- plsurvreg(
  survival::Surv(time, status) ~ trt,
  dist = "weibull",
  adjustment = adj,
  control = list(iterations = 200, burnin.iterations = 100, seed = 123)
)

fit_summary <- summary(fit, probs = c(0.025, 0.5, 0.975))
print(fit_summary)
```

`survregMixBayes` *Bayesian Two-Component Mixture Survival Regression Model*

Description

Fits a Bayesian two-component parametric survival regression model using Stan. Each observation is assumed to arise from one of two latent components with component-specific survival regression parameters.

Usage

```
survregMixBayes(
  X,
  y,
  dist = "weibull",
  priors = NULL,
  control = list(iterations = 10000, burnin.iterations = 1000, seed =
    sample.int(.Machine$integer.max, 1), cores = getOption("mc.cores", 1L)),
  ...
)
```

Arguments

X	A numeric design matrix ($N \times K$), typically created by `model.matrix()`. Each row corresponds to one observation and each column to one covariate in the survival model. Missing values are not allowed.
y	A survival response. This can be either a two-column numeric matrix with columns `time` and `event`, where `event = 1` indicates an observed event and `event = 0` indicates right censoring, or a list with elements `time` and `event`. Missing values are not allowed.
dist	Character string specifying the parametric survival distribution used for both mixture components. Supported values are `"gamma"` and `"weibull"`.
priors	A named `list` of prior specifications, or `NULL`. Since the Stan models are precompiled, prior specifications are converted into the corresponding numeric hyperparameters and passed to the model as data. Any missing entries are automatically filled in using symmetric default values via `fill_defaults(priors, p_family = dist, model_type = "survival")`. Intercept and slope priors are decoupled. Use `intercept1`/`intercept2` for the intercept of each component and `beta1`/`beta2` for the slope coefficients. By default, intercept priors are `normal(0, 10)` (weakly informative), while slope priors follow the distribution-specific defaults.
control	A named `list` of control parameters for posterior sampling. Defaults are: • `iterations` (default `1e4`): total number of iterations per chain; • `burnin.iterations` (default `1e3`): number of warm-up iterations; • `seed` (default: a random integer): random seed for reproducibility; • `cores` (default `getOption("mc.cores", 1L)`): number of CPU cores used. Values supplied through `...` override the corresponding entries in `control`.
...	Optional overrides for elements of `control`, such as `iterations = 4000`, `burnin.iterations = 1000`, `seed = 123`, or `cores = 2`.

Details

The function supports "gamma" and "weibull" component distributions, with both components sharing the same family. Right-censored survival outcomes are supported.

Posterior draws are returned for the component-specific regression parameters and mixing weight. To improve interpretability of posterior summaries, the function applies a post-processing step that aligns component labels across posterior draws.

Value

An object of class "survMixBayes" containing (at least):

`m_samples` Posterior draws of aligned latent component labels (matrix of size draws × N), where component 1 corresponds to the correct-match component and component 2 to the incorrect-match component.

`estimates$coefficients` Posterior draws of regression coefficients for the correct-match component (component 1; draws × K).

`estimates$m.coefficients` Posterior draws of regression coefficients for the incorrect-match component (component 2; draws × K).

`estimates$theta` Posterior draws of the mixing weight for the correct-match component (component 1; vector of length draws).

`estimates$shape` Posterior draws of the shape parameter for the correct-match component (component 1; family-specific).

`estimates$m.shape` Posterior draws of the shape parameter for the incorrect-match component (component 2; family-specific).

`estimates$scale` Posterior draws of the scale parameter for the correct-match component (component 1; Weibull only).

`estimates$m.scale` Posterior draws of the scale parameter for the incorrect-match component (component 2; Weibull only).

`family` The survival distribution used in the model.

`call` The matched function call.

Label switching

Mixture models are invariant to permutations of component labels, which can lead to label switching in MCMC output. To ensure interpretable posterior summaries, this function applies a post-processing step that aligns component labels across posterior draws.

First, an optional global swap of labels (1 and 2) is performed if component 2 is more frequent overall. Then, labels are aligned across draws using the `ECR-ITERATIVE-1` relabeling algorithm.

References

Gutman, R., Sammartino, C., Green, T., & Montague, B. (2016). Error adjustments for file linking methods using encrypted unique client identifier (eUCI) with application to recently released prisoners who are HIV+. *Statistics in Medicine*, 35(1), 115–129. doi:10.1002/sim.6586

Stephens, M. (2000). Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4), 795–809. doi:10.1111/14679868.00265

Papastamoulis, P. (2016). *label.switching*: An R package for dealing with the label switching problem in MCMC outputs. *Journal of Statistical Software*, 69(1), 1–24. doi:10.18637/jss.v069.c01

Examples

```
# Example: Bayesian mixture survival model fit to linked survival data
# with induced linkage mismatch errors

# 1. Simulate a linked survival dataset
set.seed(301)
n <- 150
X <- matrix(rnorm(n * 2), ncol = 2)
colnames(X) <- c("x1", "x2")

# Generate survival times from a Weibull AFT model
true_time <- rweibull(
  n,
  shape = 1.5,
  scale = exp(0.5 * X[, 1] - 0.5 * X[, 2])
)

# Apply right-censoring
cens_time <- rexp(n, rate = 0.1)
event <- as.integer(true_time <= cens_time)
obs_time <- pmin(true_time, cens_time)

# Induce linkage mismatch errors in approximately 20% of records
is_mismatch <- rbinom(n, 1, 0.2)
mismatch_idx <- which(is_mismatch == 1)

shuffled <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled]
event[mismatch_idx] <- event[shuffled]

y <- cbind(time = obs_time, event = event)

# 2. Fit the Bayesian two-component mixture survival model
# Note: Iterations are set artificially low for run time
fit <- survregMixBayes(
  X = X,
  y = y,
  dist = "weibull",
  control = list(iterations = 200, burnin.iterations = 100, seed = 123)
)

# 3. Inspect posterior summaries
# (Label switching is handled automatically)
cat("Component 1 (Correct Links):\n")
print(colMeans(fit$estimates$coefficients))

cat("Component 2 (Incorrect Links):\n")
print(colMeans(fit$estimates$m.coefficients))
```

```
cat("Estimated probability of correct linkage:\n")
print(mean(fit$estimates$theta))
```

---

unsupported_generics *Unsupported Generics for postlink Models*

Description

Intercepts standard likelihood and residual-based generics that are not directly applicable to post-linkage models because the true match status remains latent.

Usage

```
## S3 method for class 'plmodel'
logLik(object, ...)

## S3 method for class 'plmodel'
profile(fitted, ...)

## S3 method for class 'plmodel'
anova(object, ...)

## S3 method for class 'plmodel'
extractAIC(fit, scale, k = 2, ...)

## S3 method for class 'plglm'
cooks.distance(model, ...)

## S3 method for class 'plglm'
rstudent(model, ...)
```

Arguments

object	A fitted model object.
...	Additional arguments ignored by these methods.
fitted	A fitted model object.
fit	A fitted model object.
scale	Optional numeric specifying the scale parameter.
k	Numeric specifying the "weight" of the equivalent degrees of freedom.
model	A fitted model object.

vcov.coxphELE *Variance-Covariance Matrix for coxphELE Objects*

Description

Extracts the variance-covariance matrix of the main parameters (outcome model coefficients) from a fitted coxphELE model.

Usage

```
## S3 method for class 'coxphELE'
vcov(object, ...)
```

Arguments

object	An object of class coxphELE.
...	Additional arguments (currently ignored).

Value

A matrix of the estimated covariances between the parameter estimates.

Examples

```
library(survival)
set.seed(104)
n <- 200

# 1. Simulate covariates
age_centered <- rnorm(n, 0, 5)
treatment <- rbinom(n, 1, 0.5)

# 2. Simulate true survival times
true_time <- rexp(n, rate = exp(0.05 * age_centered - 0.6 * treatment))
cens_time <- rexp(n, rate = 0.2)
time <- pmin(true_time, cens_time)
status <- as.numeric(true_time <= cens_time)

# 3. Induce 15% Exchangeable Linkage Error (ELE)
mis_idx <- sample(1:n, size = floor(0.15 * n))
linked_age <- age_centered
linked_trt <- treatment

# False links drawn uniformly from the target population
false_link_idx <- sample(1:n, size = length(mis_idx), replace = TRUE)
linked_age[mis_idx] <- age_centered[false_link_idx]
linked_trt[mis_idx] <- treatment[false_link_idx]

linked_data <- data.frame(time = time, status = status,
```

```
                          age = linked_age, treatment = linked_trt)

# 4. Fit the adjusted Cox PH model
adj <- adjELE(linked.data = linked_data, m.rate = 0.15)
fit <- plcoxph(Surv(time, status) ~ age + treatment, adjustment = adj)

# 5. Extract the sandwich variance-covariance matrix
vmat <- vcov(fit)
print(vmat)
```

`vcov.coxphMixture` *Extract Variance-Covariance Matrix from a coxphMixture Object*

Description

Extracts the variance-covariance matrix of the main parameters from a fitted `coxphMixture` object. The matrix is estimated using Louis' method (1982) to account for the missing data structure (latent match status) inherent in the mixture model.

Usage

```
## S3 method for class 'coxphMixture'
vcov(object, ...)
```

Arguments

`object`	An object of class `coxphMixture`.
`...`	Additional arguments (currently ignored).

Value

A matrix of the estimated covariances between the parameter estimates. The rows and columns correspond to the outcome model coefficients (`beta`) and the mismatch model coefficients (`gamma`).

References

Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(2), 226-233.

Examples

```
library(survival)
set.seed(201)

# Simulate survival data (N = 200)
n <- 200
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.5)
```

```
true_time <- rexp(n, rate = exp(0.5 * x1 - 0.5 * x2))
cens_time <- rexp(n, rate = 0.5)

# Simulate auxiliary match scores and linkage errors
# Lower match scores correspond to a higher probability of mismatch
match_score <- runif(n, 0.5, 1.0)
is_mismatch <- rbinom(n, 1, prob = 1 - match_score)

# Induce linkage errors by shuffling covariates of mismatched records
linked_x1 <- x1
linked_x2 <- x2
mis_idx <- which(is_mismatch == 1)
shuffled_idx <- sample(mis_idx)
linked_x1[mis_idx] <- x1[shuffled_idx]
linked_x2[mis_idx] <- x2[shuffled_idx]

linked_data <- data.frame(
  time = pmin(true_time, cens_time),
  status = as.numeric(true_time <= cens_time),
  x1 = linked_x1, x2 = linked_x2,
  match_score = match_score
)

# Fit the Cox PH Mixture Model (Slawski et al., 2023)
adj <- adjMixture(linked.data = linked_data, m.formula = ~ match_score)
fit <- plcoxph(Surv(time, status) ~ x1 + x2, adjustment = adj,
               control = list(max.iter = 15))

# Extract the variance-covariance matrix
# Note: For outcome coefficients (beta) and mismatch coefficients (gamma)
vmat <- vcov(fit)
print(vmat)
```

`vcov.ctableMixture` *Extract Variance-Covariance Matrix from ctableMixture Objects*

Description

Extracts the estimated variance-covariance matrix of the cell probabilities from a fitted `ctableMixture` object. The variance is estimated using the observed information matrix (via the Hessian of the mixture log-likelihood).

Usage

```
## S3 method for class 'ctableMixture'
vcov(object, ...)
```

Arguments

object	An object of class ctableMixture.
...	Additional arguments (currently ignored).

Value

A matrix of the estimated covariances between the cell probability estimates. The row and column names correspond to the cells of the table in row-major order (e.g., "(Row1, Col1)", "(Row1, Col2)", ...).

Examples

```
set.seed(125)
n <- 300

# 1. Simulate true categorical data with dependency
exposure <- sample(c("low", "high"), n, replace = TRUE)

# Induce dependency - High exposure -> higher disease probability
prob_disease <- ifelse(exposure == "high", 0.7, 0.3)
true_disease <- ifelse(runif(n) < prob_disease, "yes", "no")

# 2. Induce 15% linkage error
mis_idx <- sample(1:n, size = floor(0.15 * n))
obs_disease <- true_disease
obs_disease[mis_idx] <- sample(obs_disease[mis_idx])

linked_df <- data.frame(exposure = exposure, disease = obs_disease)

# 3. Fit the adjusted contingency table model
adj <- adjMixture(linked.data = linked_df, m.rate = 0.15)
fit <- plctable(~ exposure + disease, adjustment = adj)

# 4. Extract the variance-covariance matrix of the cell probabilities
vmat <- vcov(fit)
print(vmat)
```

---

vcov.glmELE *Extract Variance-Covariance Matrix from a* glmELE *Object*

Description

Extracts the variance-covariance matrix of the main parameters for a specific weighting method.

Usage

```
## S3 method for class 'glmELE'
vcov(object, weight.matrix = NULL, ...)
```

Arguments

object	An object of class "glmELE".
weight.matrix	Character string specifying which weighting method to return. Defaults to the first method found in the object.
...	Additional arguments passed to methods.

Value

A matrix of the estimated covariances between the parameter estimates.

Examples

```
data(brfss, package = "postlink")

adj_object <- adjELE(linked.data = brfss,
                     m.rate = unique(brfss$m.rate),
                     blocks = imonth,
                     weight.matrix = "BLUE")

fit <- plglm(Weight ~ Height + Physhlth + Menthlth + Exerany,
             family = "gaussian", adjustment = adj_object)

vcov(fit)
```

---

vcov.glmMixBayes *Posterior Covariance Matrix for glmMixBayes Coefficients*

Description

Posterior Covariance Matrix for glmMixBayes Coefficients

Usage

```
## S3 method for class 'glmMixBayes'
vcov(object, ...)
```

Arguments

object	A glmMixBayes model object.
...	Not used.

Value

Posterior covariance matrix of the regression coefficients for component 1 (the correct-match component).

Examples

```
data(lifem)

# lifem data preprocessing
# For computational efficiency in the example, we work with a subset of the lifem data.
lifem <- lifem[order(-(lifem$commf + lifem$comml)), ]
lifem_small <- rbind(
  head(subset(lifem, hndlnk == 1), 100),
  head(subset(lifem, hndlnk == 0), 20)
)

x <- cbind(1, poly(lifem_small$unit_yob, 3, raw = TRUE))
y <- lifem_small$age_at_death

adj <- adjMixBayes(
  linked.data = lifem_small,
  priors = list(theta = "beta(2, 2)")
)

fit <- plglm(
  age_at_death ~ poly(unit_yob, 3, raw = TRUE),
  family = "gaussian",
  adjustment = adj,
  control = list(
    iterations = 200,
    burnin.iterations = 100,
    seed = 123
  )
)

vcov(fit)
```

---

vcov.glmMixture *Extract Variance-Covariance Matrix from a glmMixture Object*

Description

Returns the variance-covariance matrix of the main parameters of a fitted `glmMixture` object. The matrix is estimated using a sandwich estimator to account for the mixture structure.

Usage

```
## S3 method for class 'glmMixture'
vcov(object, ...)
```

Arguments

object	An object of class glmMixture.
...	Additional arguments (currently ignored).

Value

A matrix of the estimated covariances between the parameter estimates. Row and column names correspond to the parameter names (coefficients, dispersion, etc.).

Examples

```
# Load the LIFE-M demo dataset
data(lifem)

# Phase 1: Adjustment Specification
# We model the correct match indicator via logistic regression using
# name commonness scores (commf, comml) and a 5% expected mismatch rate.
adj_object <- adjMixture(
 linked.data = lifem,
 m.formula = ~ commf + comml,
 m.rate = 0.05,
 safe.matches = hndlnk
)

# Phase 2: Estimation & Inference
# Fit a Gaussian regression model utilizing a cubic polynomial for year of birth.
fit <- plglm(
 age_at_death ~ poly(unit_yob, 3, raw = TRUE),
 family = "gaussian",
 adjustment = adj_object
)

vcov(fit)
```

---

vcov.survMixBayes *Posterior Covariance Matrix for survMixBayes Coefficients*

Description

Returns the empirical posterior covariance matrix of the regression coefficients for component 1 of a fitted survMixBayes model. In this package, component 1 is interpreted as the correct-match component.

Usage

```
## S3 method for class 'survMixBayes'
vcov(object, ...)
```

Arguments

object	A survMixBayes model object.
...	Further arguments (unused).

Value

Posterior covariance matrix of the regression coefficients for component 1, interpreted as the correct-match component.

Examples

```
set.seed(301)
n <- 150
trt <- rbinom(n, 1, 0.5)

# Simulate Weibull AFT data
true_time <- rweibull(n, shape = 1.5, scale = exp(1 + 0.8 * trt))
cens_time <- rexp(n, rate = 0.1)
true_obs_time <- pmin(true_time, cens_time)
true_status <- as.integer(true_time <= cens_time)

# Induce linkage mismatch errors in approximately 20% of records
is_mismatch <- rbinom(n, 1, 0.2)
obs_time <- true_obs_time
obs_status <- true_status
mismatch_idx <- which(is_mismatch == 1)

shuffled <- sample(mismatch_idx)
obs_time[mismatch_idx] <- obs_time[shuffled]
obs_status[mismatch_idx] <- obs_status[shuffled]

linked_df <- data.frame(time = obs_time, status = obs_status, trt = trt)

adj <- adjMixBayes(linked.data = linked_df)

fit <- plsurvreg(
  survival::Surv(time, status) ~ trt,
  dist = "weibull",
  adjustment = adj,
  control = list(iterations = 200, burnin.iterations = 100, seed = 123)
)

# Extract the empirical posterior covariance matrix for component 1
vcov_mat <- vcov(fit)
print(vcov_mat)
```

Index

∗ **datasets**
    brfss, 11
    LD1000, 36
    lifem, 38

adjELE, *4*, 6, *44*, *48*, *61*
adjMixBayes, *4*, 8, *48*, *50*, *61*
adjMixture, *4*, 9, *44*, *46*, *48*, *62*
anova.plmodel *(*unsupported_generics*)*, 87

brfss, 11

coef.glmELE, 12
coef.glmMixBayes, 13
coef.glmMixture, 13
coef.survMixBayes, 14
confint.coxphELE, 14
confint.coxphMixture, 15
confint.ctableMixture, 17
confint.glmELE, 18
confint.glmMixBayes, 19
confint.glmMixture, 20
confint.survMixBayes, 21
cooks.distance.plglm
        *(*unsupported_generics*)*, 87
coxph, *43*
coxphELE, *5*, 23, *44*
coxphMixture, 25, *44*
ctableMixture, 27, *46*, *67*

df.residual.plglm, 29

extractAIC.plmodel
        *(*unsupported_generics*)*, 87

glm, *47*
glmELE, 30, *48*
glmMixBayes, 32, *48*
glmMixture, *5*, 34, *48*

LD1000, 36
lifem, 38
logLik.plmodel *(*unsupported_generics*)*,
        87

mi_with, 39
mi_with.glmMixBayes, 39
mi_with.survMixBayes, 41

na.fail, *47*

plcoxph, *4*, 43
plcoxph(), *7*, *10*
plctable, *4*, 45, *67*
plctable(), *10*
plglm, *4*, 47
plglm(), *7*, *8*, *10*
plsurvreg, *4*, 49
plsurvreg(), *8*
postlink *(*postlink-package*)*, 4
postlink-package, 4
predict.coxphELE, 50
predict.coxphMixture, 52
predict.glmELE, 54
predict.glmMixBayes, 55
predict.glmMixture, 57
predict.survMixBayes, 59
print.adjELE, 60
print.adjMixBayes, 61
print.adjMixture, 62
print.adjustment, 63
print.coxphELE, 64
print.coxphMixture, 65
print.ctableMixture, 66
print.default, *64*, *66*
print.glmELE, 67
print.glmMixBayes, 68
print.glmMixture, 69
print.mi_link_pool_glm, 70
print.mi_link_pool_survreg, 72
print.survMixBayes, 73

profile.plmodel *(*unsupported_generics*)*, 87

rstudent.plglm *(*unsupported_generics*)*, 87

summary.coxphELE, 74
summary.coxphMixture, 76
summary.ctableMixture, 77
summary.glmELE, 79
summary.glmMixBayes, 80
summary.glmMixture, 81
summary.survMixBayes, 82
Surv, *43*, *49*
survreg, *49*
survregMixBayes, *5*, *49*, *50*, 83

unsupported_generics, 87

vcov.coxphELE, 88
vcov.coxphMixture, *16*, 89
vcov.ctableMixture, *17*, 90
vcov.glmELE, 91
vcov.glmMixBayes, 92
vcov.glmMixture, 93
vcov.survMixBayes, 94