Package 'dsFedAteClient'

November 14, 2026

Title Federated Average Treatment Effect Estimation Across Studies

Version 0.1.0

Description Client-side 'DataSHIELD' package that orchestrates the federated estimation of the average treatment effect (ATE) across multiple independent randomised controlled trials (RCTs), each held on a separate data server. Implements in a federated way the g-formula estimator from Khellaf, Bellet and Josse (2025) ``Federated Causal Inference: Multi-Study ATE Estimation beyond Meta-Analysis'' <doi:10.48550/arXiv.2410.16870>, including variance estimation and optional adjustment for study-specific intercept shifts.

License AGPL (>= 3)

URL https://github.com/FranciscoAndrade90/dsFedAteClient

BugReports https://github.com/FranciscoAndrade90/dsFedAteClient/issues

Encoding UTF-8

Imports cli, DSI, dsBaseClient, methods

Suggests devtools, dsFedAte, DSLite, knitr, MASS, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Francisco Andrade [aut, cre], Rémi Khellaf [aut], Julie Josse [aut]

Maintainer Francisco Andrade <francisco.de-lima-andrade@inria.fr>

Repository CRAN

Date/Publication 2026-11-13 23:30:02 UTC

Contents

ds.EstimateAte 2

Index **6**

`ds.EstimateAte` *Estimate ATE from multiple RCTs*

Description

Client-side function that orchestrates the federated estimation of an average treatment effect (ATE) – and, optionally, its variance and confidence interval – from multiple independent RCTs, each held on a separate data server. Randomisation holds within each study; the combined dataset is not necessarily itself a single RCT.

Usage

```
ds.EstimateAte(
  datasources = NULL,
  base_name,
  outcome_name,
  treatment_name,
  covariate_names,
  compute_variance = TRUE,
  study_effects = FALSE,
  heterogeneous = TRUE,
  heteroscedastic_arms = FALSE,
  quiet = FALSE
)
```

Arguments

`datasources`	A list of `DSConnection-class` objects. If `NULL`, the default connections are used.
`base_name`	A character string naming the server-side data frame (must be identical across all servers).
`outcome_name`	A character string naming the outcome variable.
`treatment_name`	A character string naming the binary treatment indicator.
`covariate_names`	A character vector of covariate names.
`compute_variance`	Logical. If `TRUE` (default), the function returns a variance estimate and a 95% confidence interval in addition to the ATE point estimate. To compute these, the per-arm Gram inverses are obtained by calling `ds.glm` with `viewVarCov = TRUE`, and each server additionally returns the within-site projected scatter scalar and – when `heterogeneous = TRUE` – covariate column sums. All of this constitutes an extra disclosure step. Set to `FALSE` to return only the ATE point estimate and minimise the information shared by each server.

`study_effects` Logical. If `FALSE` (default), the no-study-effects model is assumed (potential outcome model defined in equation (2) of Khellaf et al., 2025): there is no arrow H -> Y in the DAG, meaning study membership does not directly shift the outcome. If `TRUE`, the study-effects model is used (potential outcome model (16) of Khellaf et al., 2025): an arrow H -> Y is present, so site-specific intercept shifts confound the estimate when treatment probabilities also vary across sites. K-1 integer dummy columns (`study_id2`, ..., `study_idK`) are appended to each server's data frame by `addStudyIdDS()`, and the federated GLMs are fitted on the augmented design matrix, absorbing the shifts. Has no effect and emits a warning when only one datasource is present.

`heterogeneous` Logical. Controls which variance formula is used. `TRUE` (default) makes no assumption about the participating RCTs: each site may have a different covariate distribution and a different treatment probability. The variance requires the per-arm Gram inverses (always returned via `ds.glm(viewVarCov = TRUE)` when `compute_variance = TRUE`) and the global covariate mean assembled from per-server column sums returned by `localAteDS()`. `FALSE` assumes that all RCTs share the same covariate distribution and uses a simpler closed-form variance that is exact only under this homogeneity assumption. Has no effect when `compute_variance = FALSE`.

`heteroscedastic_arms`
Logical. Only has effect when `compute_variance = TRUE`. If `FALSE` (default), the noise variance σ^2 is assumed identical in both arms and estimated by pooling the arm-specific residual sums of squares (rss) from `ds.glm` weighted by their residual degrees of freedom (df). If `TRUE`, arm-specific variances σ_1^2 and σ_0^2 are estimated from each arm's rss and residual df, and the variance formula is adjusted accordingly.

`quiet` Logical. If `TRUE`, suppresses all messages and printed output from the federated calls (`ds.glm`, `datashield.aggregate`, etc.). Default `FALSE`.

Details

The function fits separate arm-specific linear models via `ds.glm`, broadcasts the pooled coefficients back to the servers, and collects local statistics via a single aggregate call to `localAteDS`. When `compute_variance = FALSE`, three scalars are returned per server (n_k, $n_k^{(1)}$, $\hat{\tau}_k$): $\hat{\tau}_k$ and n_k suffice for the point estimate; $n_k^{(1)}$ is additionally returned so that arm sizes can be checked for asymptotic reliability. The residual sum of squares and projected scatter are omitted, as these constitute additional disclosure.

Server-side objects created. Two symbols are always created per server: `base_name_treated` (treated-arm subset) and `base_name_control` (control-arm subset), both used by `ds.glm`. When `study_effects = TRUE`, a third symbol `base_name_study_adj` is also created, holding the data frame augmented with the K-1 study dummy columns. All three persist in the server workspace after the function returns, as is typical for DataSHIELD assign operations. The caller's original `base_name` object is never modified.

When `compute_variance = TRUE` (the default), a delta-method variance estimate and 95% confidence interval are also returned. The formula used depends on the `heterogeneous` argument; see below.

Value

A list. When `compute_variance = TRUE` (the default), seven elements:

ate Federated ATE estimate $\hat{\tau}$.

variance Variance estimate $\hat{V}(\hat{\tau})$.

ci_lower Lower 95-percent confidence interval bound.

ci_upper Upper 95-percent confidence interval bound.

n_total Total sample size across all servers.

n_k Integer vector of per-server sample sizes.

ate_k Numeric vector of per-server local ATE estimates.

When `compute_variance = FALSE`, four elements: `ate`, `n_total`, `n_k`, `ate_k`. The structure is the same regardless of `study_effects`.

References

Khellaf, R., Bellet, A., and Josse, J. (2025). Federated Causal Inference: Multi-Study ATE Estimation beyond Meta-Analysis. *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS 2025)*.

Examples

```
if (requireNamespace("DSLite", quietly = TRUE) &&
    requireNamespace("dsFedAte", quietly = TRUE)) {
  library(dsFedAte)
  library(DSLite)

  set.seed(1)
  n <- 200
  study_data <- data.frame(W = rbinom(n, 1, 0.5), X1 = rnorm(n))
  study_data$Y <- 0.5 * study_data$W + study_data$X1 + rnorm(n)

  datashield_env <- new.env(parent = emptyenv())
  options(datashield.env = datashield_env,
          datashield.privacyControlLevel = "banana",
          nfilter.glm = 0.33)

  srv <- DSLite::newDSLiteServer(tables = list(study_data = study_data))
  srv$config(DSLite::defaultDSConfiguration(include = c("dsBase", "dsFedAte")))
  srv$aggregateMethod("localAteDS", "localAteDS")
  srv$aggregateMethod("listDisclosureSettingsDS", "listDisclosureSettingsDS")
  srv$aggregateMethod("checkPermissivePrivacyControlLevel",
                      "checkPermissivePrivacyControlLevel")
  assign("dslite.server", srv, envir = datashield_env)

  builder <- DSI::newDSLoginBuilder()
  builder$append(server = "s1", url = "dslite.server",
                 table = "study_data", driver = "DSLiteDriver")
  conns <- DSI::datashield.login(logins = builder$build(), assign = FALSE)
  DSI::datashield.assign.table(conns, symbol = "study_data", table = "study_data")
```

```
  result <- ds.EstimateAte(
    datasources     = conns,
    base_name       = "study_data",
    outcome_name    = "Y",
    treatment_name  = "W",
    covariate_names = "X1",
    quiet           = TRUE
  )
  print(result$ate)

  DSI::datashield.logout(conns)
}
```

Index

ds.EstimateAte, 2
ds.glm, *3*