Package 'ClusterRandSSAdj'

June 26, 2026

Title Small Sample Adjustment of Cluster Randomized Trial

Version 1.0.0

Description A set of functions to apply 'HC3' (FIRORES) and 'HC2' (ROOT) sandwich estimators to make small-sample adjustments to standard errors of Generalized Linear Model statistics used to analyze cluster randomized trial data. The functions in the 'ClusterRandSSAdj' package make small-sample adjustments to Generalized Linear Model parameter estimates, least squares means and pairwise comparison of least squares means, Type III tests, and estimates from linear combinations of Generalized Linear Model parameters. For more details see Ford (2017) <doi:10.1002/bimj.201600182> and Westgate (2022) <doi:10.1177/17407745211063479>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, car, emmeans, lmtest, Matrix, multcomp, sandwich

Depends R (>= 4.1)

LazyData true

Suggests MASS, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Barry Eggleston [aut, cre],
Rouba Chahine [aut],
Phil Westgate [aut],
Nathan Vandergrift [fnd]

Maintainer Barry Eggleston <beggleston@rti.org>

Repository CRAN

Date/Publication 2026-06-26 09:50:15 UTC

Contents

`cluster_rct_data` *A simulated dataset,* `cluster_rct_data`

Description

The `cluster_rct_data` dataset is a simulated dataset representing a cluster randomized clinical trial.

Usage

```
cluster_rct_data
```

Format

A data frame with 50 rows and 9 variables:

ClusterID A cluster ID variable, which is simply the row number (integer)

Population_size Simulated population size for represented cluster (numeric)

Log_population_size Natural log of simulated population size for represented cluster (numeric)

Treatment A 0/1 treatment indicator (factor, reference level = 0)

Baserate Simulated baseline outcome rate for each cluster (numeric)

Log_baserate Natural log of simulated baseline outcome rate for each cluster (numeric)

Covar1 A simulated 0/1 covariate (factor, reference level = 0)

Covar2 A simulated 1/2/3 covariate (factor, reference level = 1)

Outcome A simulated outcome count for each cluster (integer)

Source

Generated for demonstration purposes.

`EstimateAdj` *Small-sample adjustment for a set of linear contrasts from Generalized Linear Models used in analysis of cluster randomized trial data*

Description

`EstimateAdj()` takes a Generalized Linear Model, a matrix of contrasts, and applies a small-sample adjustment to the standard errors of the model parameter estimates, then uses `multcomp::glht()` to estimate the contrast estimates (point estimate and 100*(1-alpha)% confidence interval) and complete hypothesis tests (t-tests) on the contrasts.

Usage

```
EstimateAdj(model_obj, contrast_mat, alpha_val = 0.05, df_val = NULL)
```

Arguments

`model_obj`	A fitted `glm()` model or a model that inherits from `glm()`.
`contrast_mat`	A matrix of contrast, analogous to a set of estimate statements in SAS, but based on the model parameterization used in the model object.
`alpha_val`	The desired Type I alpha level for confidence intervals of lsmeans. Must be between 0 and 1.
`df_val`	Specifies the error degrees of freedom, if a value different from that used in the original model fit is desired. This argument defaults to NULL, in which case the degrees of freedom from the original model fit are used in the t-tests.

Details

The model object passed into `EstimateAdj()` must be a fitted `glm()` model or a model that inherits from a `glm()`. Point estimates and confidence intervals returned by `EstimateAdj()` are on the scale of the link function, so unless the distribution is Gaussian and the link is the identity function, the returned point estimates and confidence intervals need to be transformed using the inverse link function to get meaningful results. Small sample adjustment of standard errors is made by averaging the standard errors calculated from `sandwich::vcovHC()` using `type = "HC3"` and `type = "HC2"`.

Value

A data.frame of small-sample adjusted contrast estimates on the scale of the link function. The data frame will have the following columns:

label A text identifier taken from the contrast matrix

SSAdj_Estimate Contrast estimates (estimates for HC3 adjusted model, which are the same as unadjusted and HC2 adjusted models)

SSAdj_StdErr Small-sample adjusted standard error of the contrast estimates

SSAdj_tValue T-statistics for test of contrast estimates equal to 0

SSAdj_DF Degrees of freedom used in t-test of the contrast estimates equal to 0

SSAdj_Probt Two-sided p-value from t-test of the contrast estimates equal to 0

SSAdj_EstLCL Lower 100*(1-alpha)% confidence limit of the contrast estimates

SSAdj_EstUCL Upper 100*(1-alpha)% confidence limit of the contrast estimates

References

Ford WP, Westgate PM. Improved standard error estimator for maintaining the validity of inference in cluster randomized trials with a small number of clusters. Biom J. 2017 May;59(3):478-495. doi: 10.1002/bimj.201600182. Epub 2017 Jan 27. PMID: 28128854.

Westgate PM, Cheng DM, Feaster DJ, Fernández S, Shoben AB, Vandergrift N. Marginal modeling in community randomized trials with rare events: Utilization of the negative binomial regression model. Clin Trials. 2022 Apr;19(2):162-171. doi: 10.1177/17407745211063479. Epub 2022 Jan 6. PMID: 34991359; PMCID: PMC9038610.

Kauermann G and Carroll RJ. A note on the efficiency of sandwich covariance matrix estimation. Journal of the American Statistical Association 2001; 96: 1387–1396.

Mancl LA and DeRouen TA. A covariance estimator for GEE with improved small-sample properties. Biometrics 2001; 57: 126–134.

Examples

```
# If small-sample adjustment is needed for user defined linear combinations
# of GLM model parameters, then `EstimateAdj()` is the function to use.
# Consider the simulated cluster randomized trial dataset, `cluster_rct_data`,
# which is a part of the `ClusterRandSSAdj` package.  This dataset has variables:
#   clusterID,
#   Population_size,
#   Log_population_size,
#   Treatment,
#   Baserate,
#   Log_baserate,
#   Covar1 (0/1),
#   Covar2 (1/2/3), and
#   Outcome.
# This simulated data can be modeled using a Negative binomial model.
# In order to complete a small-sample adjustment on user defined linear
# combinations of model parameters, one first needs to fit the model:

nb_model_test <- MASS::glm.nb(Outcome ~ Treatment + Log_baserate + Covar1 +
   Covar2 + Treatment:Covar2 + Treatment:Covar1 +
   offset(Log_population_size), data = cluster_rct_data)

# Once the model is fit, the small-sample adjusted user defined linear
# combinations can be calculated by passing the fitted model, a matrix of
# user defined linear contrasts, and a value for `alpha` to the `EstimateAdj()`
# function. The default value of `alpha` is 0.05, so the `alpha`
# function parameter can be ignored unless a value different than
# 0.05 is desired.
#
# Consider the lsmeans for the Treatment by Covar2 term.  Lsmeans can be
# estimated using the following code:
```

```
LSMeansAdj(nb_model_test, ~Treatment:Covar2, 0.05)

# Consider the following:
contrast_matrix <- rbind(
  "logT0_1" = c(1, 0,1.937217, 0.5, 0, 0, 0, 0, 0),
  "logT1_1" = c(1, 1,1.937217, 0.5, 0, 0, 0, 0, 0.5),
  "logT0_2" = c(1, 0,1.937217, 0.5, 1, 0, 0, 0, 0),
  "logT1_2" = c(1, 1,1.937217, 0.5, 1, 0, 1, 0, 0.5),
  "logT0_3" = c(1, 0,1.937217, 0.5, 0, 1, 0, 0, 0),
  "logT1_3" = c(1, 1,1.937217, 0.5, 0, 1, 0, 1, 0.5)
)
# The following call to `EstimateAdj()` will produce small-sample adjusted
# estimates of the linear contrasts represented in `contrast_matrix`.
lsmean_replicate <- EstimateAdj(nb_model_test, contrast_matrix, 0.05)
lsmean_replicate

# The values in `lsmean_replicate` are the lsmeans for Treatment by
# Covar2 combinations.  The call to `EstimateAdj()` replicated the call
# to `LSMeansAdj()`.  At first glance, the might suggest that `EstimateAdj()`
# is redundant; however, let us continue exploring user defined
# linear combinations of model parameters. Specifically, we will examine
# pairwise comparisons of Treatment by Covar2 combinations, focusing on the
# Treatment = 1 minus Treatment = 0 comparisons within each level of
# Covar2:

lsmean_comps <- LSMeansPairwiseCompAdj(nb_model_test,
                 ~Treatment:Covar2, 0.10, reverse=TRUE)
lsmean_comps_sub <- lsmean_comps |>
                    dplyr::filter(contrast %in%
                          c("Treatment1 Covar21 - Treatment0 Covar21",
                            "Treatment1 Covar22 - Treatment0 Covar22",
                            "Treatment1 Covar23 - Treatment0 Covar23"))
lsmean_comps_sub

# Now, consider the following contrast matrix and call to `EstimateAdj()`:

contrast_matrix2 <- rbind(
  "logT1vT0_1" = c(0, 1, 0, 0, 0, 0, 0, 0, 0.5),
  "logT1vT0_2" = c(0, 1, 0, 0, 0, 0, 1, 0, 0.5),
  "logT1vT0_3" = c(0, 1, 0, 0, 0, 0, 0, 1, 0.5)
)

# The rows in `contrast_matrix2` represent the expected difference in
# Treatment = 1 minus Treatment = 0 when Covar2 = 1 (row1), the expected
# difference in Treatment = 1 minus Treatment = 0 when Covar2 = 2 (row1),
# and the expected difference in Treatment = 1 minus Treatment = 0 when
# Covar2 = 3 (row3).

# The following call to `EstimateAdj()` will produce small-sample adjusted
# estimates of the linear contrasts represented in `contrast_matrix2`.

lsmeanpwc_replicate <- EstimateAdj(nb_model_test, contrast_matrix2, 0.10)
```

```
lsmeanpwc_replicate

# Given `lsmeanpwc_replicate` contains estimates of differences in
# log(expected count) between Treatment 1 and Treatment 0 within each
# level of Covar2, which is the differences in lsmeans, consequently
# `lsmeanpwc_replicate` has the same small-sample adjusted results
# as `lsmean_comps_sub`.
#
# Finally, in the above paragraph we talked about treatment differences
# within each level of Covar2.  What about comparisons between levels
# of Covar2 with respect to treatment differences?  Stated another way,
# what about Treatment 1 minus Treatment 0 differences in
# log(expected count) between levels of Covar2?  Such estimates are
# estimates of difference in differences.  With 3 levels in Covar2, we
# have a set of three difference in treatment group differences.  The
# following is the contrast matrix for the 3 possible difference in
# differences:

contrast_matrix3 <- rbind(
"logT1vT0_1mlogT1vT0_2" = c(0, 0, 0, 0, 0, 0, -1,  0, 0),
"logT1vT0_1mlogT1vT0_3" = c(0, 0, 0, 0, 0, 0,  0, -1, 0),
"logT1vT0_2mlogT1vT0_3" = c(0, 0, 0, 0, 0, 0,  1, -1, 0)
)
dodEst <- EstimateAdj(nb_model_test, contrast_matrix3, 0.10)
dodEst

# This final call to `EstimateAdj()` demonstrates one of the ways
# `EstimateAdj()` can be used to make inference from small-sample
# adjusted linear functions of GLM model parameters.  These differences in
# differences are not pairwise comparisons of lsmeans, but pairwise
# comparisons of differences in lsmeans, as such these estimates are
# only obtainable via a call to `EstimateAdj()`.
#
# One final step could be to transform these differences in differences
# which are on the log scale, to ratios of expected count ratios.  In
# other words, if the quantity, exp(differences in log(expected count)),
# is a ratio of expected counts, then
# exp(differences in log(expected count1) - differences in log(expected count2))
# will equal a ratio of expected count ratios.
new_lab <- c("T1vT0_1_2", "T1vT0_1_3", "T1vT0_2_3")
cbind(new_lab, exp(dodEst[,c(2, 7, 8)]))
```

`Lincom_FtestAdj` *Small-sample adjustment of SAS like F-tests for linear contrasts*

Description

Small-sample adjustment of SAS like F-tests for linear contrasts

Usage

```
Lincom_FtestAdj(model_obj, L_value, denomdf_val = NULL)
```

Arguments

model_obj	A glm() model or a model that inherits from glm().
L_value	A user defined matrix of contrasts representing linear contrasts of model parameters.
denomdf_val	The denominator degrees of freedom desired for the F-test. If left unspecified, the residual degrees of freedom from the fitted model are used.

Details

This function can be used to make small-sample adjustments of F-tests for linear contrasts of model parameters. The model object passed into Lincom_FtestAdj() must be a fitted glm() model or a model that inherits from glm(). Small sample adjustment of standard errors is made by averaging the standard errors calculated from sandwich::vcovHC() using type = "HC3" and type = "HC2".

Value

A data.frame of small-sample adjusted F-test results for a contrast matrix. The data frame will have the following columns:

Numerator_df Numerator degrees of freedom for the F-test

Denominator_df Denominator degrees of freedom for the F-test)

SSAdjF_statistic Small-sample adjusted F-statistic for the test that all rows of the contrast matrix equal 0

SSAdjF_pvalue P-value for the small-sample adjusted F-statistic that all rows of the contrast matrix equal 0

Examples

```
# If an F-test of a linear combination of model parameters and
# small-sample adjustment is needed, then `Lincom_FtestAdj()` is a
# useful function.  Consider the simulated cluster randomized trial
# dataset, `cluster_rct_data`, which is a part of the `ClusterRandSSAdj`
# package.  This dataset has variables:
#   clusterID,
#   Population_size,
#   Log_population_size,
#   Treatment,
#   Baserate,
#   Log_baserate,
#   Covar1 (0/1),
#   Covar2 (1/2/3), and
#   Outcome.
# This simulated data can be modeled using a Negative binomial model.
# In order to perform a small-sample adjustment of an F-test for a linear
# combination of model parameters, the necessary model needs to be fitted,
```

```
# which in this case is a Negative Binomial model:

nb_model_test <- MASS::glm.nb(Outcome ~ Treatment + Log_baserate + Covar1 +
   Covar2 + Treatment:Covar2 + Treatment:Covar1 +
   offset(Log_population_size), data = cluster_rct_data)

# For illustration of how to use `Lincom_FtestAdj()`, let's apply the
# small-sample adjusted SAS Type-III test for the treatment by covar2 interaction.

# Type III F-test for treatment by covar2 interaction
L_trt_by_covar2 <- rbind(
 c(0, 0, 0, 0, 0, 0, 1, 0, 0),
 c(0, 0, 0, 0, 0, 0, 0, 1, 0)
)
Lincom_FtestAdj(nb_model_test, L_trt_by_covar2)
```

`LSMeansAdj` *Small-sample adjustment of least squares means from a glm used for analysis of cluster randomized trial*

Description

`LSMeansAdj()` takes a fitted `glm()` model or a model that inherits from a `glm()`, applies a small-sample adjustment to the standard errors of the model parameter estimates, then calculates SAS like least squares means. Once the least squares means are calculated, `LSMeansAdj()` performs t-tests on the least squares means and calculates 100*(1-alpha)% confidence intervals for the least squares means.

Usage

```
LSMeansAdj(model_obj, formula_prt, alpha_val = 0.05, df_val = NULL)
```

Arguments

`model_obj`	A fitted `glm()` model or a model that inherits from `glm()`.
`formula_prt`	A right sided formula, which indicates the type of least squares means requested.
`alpha_val`	The desired Type I alpha level for confidence intervals of least squares means. Must be between 0 and 1. Default is 0.05.
`df_val`	Specifies the error degrees of freedom, if a value different from that used in the original model fit is desired. This argument defaults to NULL, in which case the degrees of freedom from the original model fit are used in the t-tests.

Details

The model object passed into `LSMeansAdj()` must be a fitted `glm()` model or a model that inherits from a `glm()`. The small-sample adjustment process averages least squares mean (lsmean) standard errors after adjusting the original model parameter estimate standard errors using the HC3 (FIRORES) and HC2 (ROOT) sandwich variance estimators calculated from `sandwich::vcovHC()`

using `type = "HC3"` and `type = "HC2"`. Once the HC3- and HC2-based lsmean standard errors are calculated, for each lsmean, the two standard errors are averaged, a t-test is completed using the averaged standard error, and a 100*(1-alpha)% confidence interval is constructed using the averaged standard error. Note, point estimates and confidence intervals returned by `LSMeansAdj()` are on the scale of the link function, so unless the distribution is Gaussian and the link is the identity function, the returned point estimates and confidence intervals need to be transformed using the inverse link function to get meaningful results. For example, if the model object passed to `LSMeansAdj()` is a Poisson model or a Negative binomial model then exponentiating the estimates and confidence interval limits returned by `LSMeansAdj()` will transform the lsmeans into estimates of expected mean counts and their corresponding confidence intervals. If the model object passed to `LSMeansAdj()` is a Logistic model then using the inverse logit transformation on the estimates and confidence interval limits returned by `LSMeansAdj()` will transform the lsmeans into estimates of expected probabilities and their corresponding confidence intervals. The right sided formula will use the expression `~{variable construct}`. For example, to calculate the lsmeans for `variable1` by `variable2`, the right sided formula would be `~variable1:variable2`. For the lsmeans for `variable1` only, the right sided formula would be `~variable1`.

Value

A data frame of small-sample adjusted lsmeans containing estimates, standard errors, degrees of freedom, t-values, p-value, and 100*(1-alpha)% confidence intervals on scale of the lsmean. The data frame will have the following columns:

First initial columns Columns will have names equal to the variables involved in the lsmean estimates

SSAdj_DF Degrees of freedom used in t-test of lsmean estimate

SSAdj_Estimate Lsmean estimates (estimate for HC3 adjusted model, which are the same as unadjusted and HC2 adjusted models)

SSAdj_StdErr Small-sample adjusted standard error of lsmean estimate

SSAdj_tValue T-statistics for test of lsmean estimate equal to 0

SSAdj_Probt Two-sided p-value from t-test of lsmean estimate equal to 0

SSAdj_EstLCL Lower 100*(1-alpha)% confidence limit of lsmean estimate

SSAdj_EstUCL Upper 100*(1-alpha)% confidence limit of lsmean estimate

References

Ford WP, Westgate PM. Improved standard error estimator for maintaining the validity of inference in cluster randomized trials with a small number of clusters. Biom J. 2017 May;59(3):478-495. doi: 10.1002/bimj.201600182. Epub 2017 Jan 27. PMID: 28128854.

Westgate PM, Cheng DM, Feaster DJ, Fernández S, Shoben AB, Vandergrift N. Marginal modeling in community randomized trials with rare events: Utilization of the negative binomial regression model. Clin Trials. 2022 Apr;19(2):162-171. doi: 10.1177/17407745211063479. Epub 2022 Jan 6. PMID: 34991359; PMCID: PMC9038610.

Kauermann G and Carroll RJ. A note on the efficiency of sandwich covariance matrix estimation. Journal of the American Statistical Association 2001; 96: 1387–1396.

Mancl LA and DeRouen TA. A covariance estimator for GEE with improved small-sample properties. Biometrics 2001; 57: 126–134.

Examples

```
# If small-sample adjustment is needed for lsmeans which are generated from a GLM
# model, then `LSMeansAdj()` is the function to use.  Consider the simulated
# cluster randomized trial dataset, `cluster_rct_data`, which is a part
# of the `ClusterRandSSAdj` package.  This dataset has variables:
#   clusterID,
#   Population_size,
#   Log_population_size,
#   Treatment,
#   Baserate,
#   Log_baserate,
#   Covar1 (0/1),
#   Covar2 (1/2/3), and
#   Outcome.
# This simulated data can be modeled using a Negative binomial model.
# In order to complete a small-sample adjustment on lsmeans from the
# model, one first needs to fit the model:

nb_model_test <- MASS::glm.nb(Outcome ~ Treatment + Log_baserate + Covar1 +
   Covar2 + Treatment:Covar2 + Treatment:Covar1 +
   offset(Log_population_size), data = cluster_rct_data)

# Once the model is fit, the small-sample adjusted lsmeans can be
# calculated by passing the fitted model, a reference to the
# desired lsmeans, and a value for `alpha` to the `LSMeansAdj()`
# function. The default value of `alpha` is 0.05, so the `alpha`
# function parameter can be ignored unless a value different from
# 0.05 is desired.

# The following code will estimate lsmeans for all combinations of
# Treatment by Covar1.  The result will contain small-sample
# adjustments to the lsmeans standard errors, t-test, and calculate
# small-sample adjusted 90% confidence of the estimated lsmeans:

LSMeansAdj(nb_model_test, ~Treatment:Covar1, 0.10)

# Since the fitted model is a Negative binomial model. The resulting
# lsmeans will be on the log scale, log(expected count). In order to
# calculate expected counts for each combination of Treatment by
# Covar 1, the log(expected count) values need to be exponentiated.
# A simple way to transform the log(expected count) to
# expected counts per person is demonstrated in the following code:

rates <- LSMeansAdj(nb_model_test, ~Treatment:Covar1, 0.05)[, c(1,2,4,8,9)]
cbind(rates[,c(1,2)], exp(rates[,c(3,4,5)]))

# If one wants expected rates per some specified number of people,
# such as per 10,000, then the following code will get the needed results:

cbind(rates[,c(1,2)], 10000*exp(rates[,c(3,4,5)]))
```

`LSMeansPairwiseCompAdj` *Small-sample adjustment of pairwise comparisons of least squares means from a Generalized Linear Model used in analysis of cluster randomized trial data*

Description

`LSMeansPairwiseCompAdj()` takes a fitted `glm()` model or a model that inherits from a `glm()`, and applies a small-sample adjustment to the standard errors of the model parameter estimates, then calculates pairwise comparisons of SAS like least squares means. Once the pairwise comparisons of least squares means are calculated, `LSMeansPairwiseCompAdj()` performs t-tests on the pairwise least squares means comparisons and calculates 100*(1-alpha)% confidence intervals for the least squares means comparisons.

Usage

```
LSMeansPairwiseCompAdj(
  model_obj,
  formula_prt,
  alpha_val = 0.05,
  reverse_val = FALSE,
  df_val = NULL
)
```

Arguments

`model_obj`	A fitted `glm()` model or a model that inherits from `glm()`.
`formula_prt`	A right sided formula, which indicates the type of least squares means that will be compared.
`alpha_val`	The desired Type I alpha level for confidence intervals of least squares means. Must be between 0 and 1.
`reverse_val`	A TRUE/FALSE value indicating if the order of least squares means pairs should be reversed
`df_val`	Specifies the error degrees of freedom, if a value different from that used in the original model fit is desired. This argument defaults to NULL, in which case the degrees of freedom from the original model fit are used in the t-tests.

Details

The model object passed into `LSMeansPairwiseCompAdj()` must be a fitted `glm()` model or a model that inherits from `glm()`. The small-sample adjustment process averages pairwise least squares mean (lsmean) comparison standard errors after adjusting the original model parameter estimate standard errors using the HC3 (FIRORES) and HC2 (ROOT) sandwich variance estimators calculated from `sandwich::vcovHC()` using `type = "HC3"` and `type = "HC2"`. Once the HC3- and HC2-based pairwise lsmean comparison standard errors are calculated, the two standard errors

are averaged. Using the averaged standard error, a t-test is then conducted and a 100*(1-alpha)% confidence interval is constructed. Note, the pairwise lsmean comparisons and corresponding confidence intervals returned by `LSMeansPairwiseCompAdj()` are on the scale of the link function, so unless the distribution is Gaussian and the link is the identity function, the returned point estimates and confidence intervals need to be transformed using the inverse link function to get meaningful results. For example, if the model object passed to `LSMeansPairwiseCompAdj()` is a Poisson model or a Negative binomial model, then exponentiating the estimates and confidence interval limits returned by `LSMeansPairwiseCompAdj()` will transform the pairwise lsmean comparison into estimates of expected mean count ratios and their confidence intervals. If the model object passed to `LSMeansAdj()` is a Logistic model then exponentiating the estimates and confidence interval limits returned by `LSMeansPairwiseCompAdj()` will transform the pairwise lsmean comparisons to estimates of estimated odds ratios and their confidence intervals. The right sided formula will use the expression `~{variable construct}`. For example, to calculate the pairwise comparison of lsmeans for `variable1` by `variable2`, the right sided formula would be `~variable1:variable2`. For pairwise comparisons of lsmeans for `variable1` only, the right sided formula would be `~variable1`.

Value

Returns a data frame of all possible pairwise comparisons adjusted for small samples. The data frame will have the following columns:

contrast Label for the pairwise contrast represented by the row

SSAdj_DF Degrees of freedom used in t-test of the contrast estimate

SSAdj_Estimate Contrast estimates (estimate for HC3 adjusted model, which are the same as unadjusted and HC2 adjusted models)

SSAdj_StdErr Small-sample adjusted standard error of contrast estimate

SSAdj_tValue T-statistics for test of contrast estimate equal to 0

SSAdj_Probt Two-sided p-value from t-test of contrast estimate equal to 0

SSAdj_EstLCL Lower 100*(1-alpha)% confidence limit of contrast estimate

SSAdj_EstUCL Upper 100*(1-alpha)% confidence limit of contrast estimate

References

Ford WP, Westgate PM. Improved standard error estimator for maintaining the validity of inference in cluster randomized trials with a small number of clusters. Biom J. 2017 May;59(3):478-495. doi: 10.1002/bimj.201600182. Epub 2017 Jan 27. PMID: 28128854.

Westgate PM, Cheng DM, Feaster DJ, Fernández S, Shoben AB, Vandergrift N. Marginal modeling in community randomized trials with rare events: Utilization of the negative binomial regression model. Clin Trials. 2022 Apr;19(2):162-171. doi: 10.1177/17407745211063479. Epub 2022 Jan 6. PMID: 34991359; PMCID: PMC9038610.

Kauermann G and Carroll RJ. A note on the efficiency of sandwich covariance matrix estimation. Journal of the American Statistical Association 2001; 96: 1387–1396.

Mancl LA and DeRouen TA. A covariance estimator for GEE with improved small-sample properties. Biometrics 2001; 57: 126–134.

Examples

```
# If small-sample adjustment is needed for pairwise comparisons of lsmeans,
# which are generated from a GLM model, then `LSMeansPairwiseCompAdj()` is
# the function to use.  Consider the simulated cluster randomized trial
# dataset, `cluster_rct_data`, which is a part of the `ClusterRandSSAdj`
# package.  This dataset has variables:
#   clusterID,
#   Population_size,
#   Log_population_size,
#   Treatment,
#   Baserate,
#   Log_baserate,
#   Covar1 (0/1),
#   Covar2 (1/2/3), and
#   Outcome.
# This simulated data can be modeled using a Negative binomial model.
# In order to complete a small-sample adjustment on pairwise comparisons
# of lsmeans from the model, first fit the model:

nb_model_test <- MASS::glm.nb(Outcome ~ Treatment + Log_baserate + Covar1 +
   Covar2 + Treatment:Covar2 + Treatment:Covar1 +
   offset(Log_population_size), data = cluster_rct_data)

# Once the model is fit, the small-sample adjusted pairwise comparisons
# of lsmeans can be calculated by passing the fitted model, a reference to the
# desired lsmeans, a value for `alpha` to the ` LSMeansPairwiseCompAdj()`
# function, and a TRUE/FALSE value to a parameter called `reverse_val`. The
# default value of `alpha` is 0.05, so the `alpha` function parameter can be
# ignored unless a value other than 0.05 is desired.  The default value
# for `reverse_val` is FALSE.  `reverse_val` is used to reverse to order
# of pairwise comparisons, if necessary, to ensure the differences are calculated
# in the desired direction.

# The following code will estimate pairwise comparisons of lsmeans for
# all combinations of Treatment by Covar1.  The result will contain small
# sample adjustments to the pairwise comparison standard errors, t-test,
# and calculate small-sample adjusted 90% confidence of the estimated
# pairwise comparisons.  Also, the order will be reversed, so the results
# of Treatment = 1 minus Treatment = 0 will be generated in the set of
# pairwise comparisons.

LSMeansPairwiseCompAdj(nb_model_test, ~Treatment:Covar1, 0.10, reverse_val=TRUE)

# Since the fitted model is a Negative binomial model. The resulting
# pairwise comparisons of lsmeans will be on the log scale, log(expected count)
# minus log(expected count). In order to calculate ratios of expected counts
# for each pairwise comparisons in Treatment by Covar 1, the pairwise
# comparison estimates will need to be exponentiated.
# A simple way to transform the differences in log(expected count) to
# ratios of expected counts is demonstrated in the following code:

lsmean_comps <- LSMeansPairwiseCompAdj(nb_model_test, ~Treatment:Covar1, 0.10, reverse=TRUE)
```

```
cbind(lsmean_comps[,1], exp(lsmean_comps[,c(3, 7, 8)]))

# To calculate the ratio of expected counts expected rates for Treatment 1 minus
# Treatment 0 in the Covar1 = 0 level and the ratio of expected counts expected rates for
# Treatment 1 minus Treatment 0 in the Covar1 = 1 level, the following code will get
# the needed results:

lsmean_comps_sub <- lsmean_comps |>
             dplyr::filter(contrast %in% c("Treatment1 Covar10 - Treatment0 Covar10",
                                           "Treatment1 Covar11 - Treatment0 Covar11"))
cbind(lsmean_comps_sub[,1], exp(lsmean_comps_sub[,c(3, 7, 8)]))
```

`ModelParmAdjEst` *Small-sample adjustment of Generalized Linear Model parameters used in analysis of cluster randomized trial data*

Description

`ModelParmAdjEst()` takes a fitted `glm()` model or a model that inherits from `glm()` and applies a small-sample adjustment to the standard errors of the model parameter estimates, then performs t-tests on the model parameter estimates and calculates 100*(1-alpha)% confidence intervals for the model parameter estimates.

Usage

```
ModelParmAdjEst(model_obj, alpha_val = 0.05, df_val = NULL)
```

Arguments

`model_obj`	A fitted `glm()` model or a model that inherits from `glm()`.
`alpha_val`	The desired Type I alpha level for confidence intervals of model parameters. Must be between 0 and 1. Default is 0.05.
`df_val`	Specifies the error degrees of freedom, if a value different from that used in the original model fit is desired. This argument defaults to NULL, in which case the degrees of freedom from the original model fit are used in the t-tests.

Details

The model object passed into `ModelParmAdjEst()` must be a fitted `glm()` model or a model that inherits from `glm()`. The small-sample adjustment process averages the model parameter standard errors after modifying the original standard errors using the HC3 (FIRORES) and HC2 (ROOT) sandwich variance estimators. The HC3-based and HC2-based standard errors are calculated using `sandwich::vcovHC()` where `type = "HC3"` and `type = "HC2"`. Once the HC3- and HC2-based standard errors are calculated and averaged for each model parameter estimate, t-tests are completed using the averaged standard errors, and 100*(1-alpha)% confidence intervals are constructed using the averaged standard errors. Note, if model_obj is a negative binomial model fit using `MASS::glm.nb()`, then the equivalent value of the SAS scale parameter can be calculated using `1/model_obj$theta`.

Value

A data frame of model parameters containing parameter estimates, standard errors, t-statistics, p-value, and 100*(1-alpha)% confidence interval on scale of model parameter. The data frame will have the following columns:

Variable Model parameter term

SSAdj_DF Degrees of freedom used in t-test of parameter estimate

SSAdj_Estimate Model parameter estimate (estimate for HC3 adjusted model, which are the same as unadjusted and HC2 adjusted models)

SSAdj_StdErr Small-sample adjusted standard error of model parameter estimate

SSAdj_tValue T-statistic for test of model parameter estimate equal to 0

SSAdj_Probt Two-sided p-value from t-test of model parameter estimate equal to 0

SSAdj_EstLCL Lower 100*(1-alpha)% confidence limit of model parameter estimate

SSAdj_EstUCL Upper 100*(1-alpha)% confidence limit of model parameter estimate

References

Ford WP, Westgate PM. Improved standard error estimator for maintaining the validity of inference in cluster randomized trials with a small number of clusters. Biom J. 2017 May;59(3):478-495. doi: 10.1002/bimj.201600182. Epub 2017 Jan 27. PMID: 28128854.

Westgate PM, Cheng DM, Feaster DJ, Fernández S, Shoben AB, Vandergrift N. Marginal modeling in community randomized trials with rare events: Utilization of the negative binomial regression model. Clin Trials. 2022 Apr;19(2):162-171. doi: 10.1177/17407745211063479. Epub 2022 Jan 6. PMID: 34991359; PMCID: PMC9038610.

Kauermann G and Carroll RJ. A note on the efficiency of sandwich covariance matrix estimation. Journal of the American Statistical Association 2001; 96: 1387–1396.

Mancl LA and DeRouen TA. A covariance estimator for GEE with improved small-sample properties. Biometrics 2001; 57: 126–134.

Examples

```
# If small-sample adjustment is needed for GLM model parameters, then
# `ModelParmAdjEst()` is the function to use.  Consider the simulated
# cluster randomized trial dataset, `cluster_rct_data`, which is a part
# of the `ClusterRandSSAdj` package.  This dataset has variables:
#   clusterID,
#   Population_size,
#   Log_population_size,
#   Treatment,
#   Baserate,
#   Log_baserate,
#   Covar1 (0/1),
#   Covar2 (1/2/3), and
#   Outcome.
# This simulated data can be modeled using a Negative binomial model.
# In order to complete a small-sample adjustment on the model
# parameters, one first needs to fit the model:
```

```
nb_model_test <- MASS::glm.nb(Outcome ~ Treatment + Log_baserate + Covar1 +
   Covar2 + Treatment:Covar2 + Treatment:Covar1 +
   offset(Log_population_size), data = cluster_rct_data)

# Once the model is fit, the small-sample adjustment can be applied to
# the model parameters, by passing the fitted model and a value for
# `alpha` to the `ModelParmAdjEst()` function.  The default value of
# `alpha` is 0.05, so the `alpha` function parameter can be ignored
# unless a value different from 0.05 is desired.  The following code
# will complete the small-sample adjustment to the standard errors of
# the model parameter estimates and calculate small-sample adjusted
# 90% confidence intervals of the model parameter estimates:

ModelParmAdjEst(nb_model_test, 0.10)
```

`SAS_Lincom_Ftest` *SAS like F-tests for linear contrasts*

Description

SAS like F-tests for linear contrasts

Usage

```
SAS_Lincom_Ftest(model_obj, L_value, adjust_type = NULL, denomdf_val = NULL)
```

Arguments

`model_obj`	A fitted `glm()` model or a model that inherits from a `glm()`.
`L_value`	A user defined matrix of contrasts representing linear contrasts of model parameters.
`adjust_type`	Type of adjustment applied to standard errors of model parameters. Possible values are "HC3" and "HC2". Default is NULL resulting in no sandwich estimator adjustment.
`denomdf_val`	The denominator degrees of freedom desired for the F-test. If left unspecified, the residual degrees of freedom from the fitted model are used.

Details

This utility function can be used to replicate SAS contrast statements and type III test results; however, it requires the user to translate the SAS contrast statements into an equivalent R contrast matrix. The model object passed into `SAS_Lincom_Ftest()` must be a fitted 'glm()`model or a model that inherits fromglm wich::vcovHC()usingtype = "HC3"andtype = "HC2"`'.

Value

A data.frame of F-test results for a contrast matrix. The data frame will have the following columns:

Numerator_df Numerator degrees of freedom for the F-test

Denominator_df Denominator degrees of freedom for the F-test)

F_statistic F-statistic for the test that all rows of the contrast matrix equal 0

F_pvalue P-value for the F-statistic that all rows of the contrast matrix equal 0

Examples

```
# If an F-test of a linear combination of model parameters is needed, then
# `SAS_Lincom_Ftest()` is a useful function.  Consider the simulated
# cluster randomized trial dataset, `cluster_rct_data`, which is a part
# of the `ClusterRandSSAdj` package.  This dataset has variables:
#   clusterID,
#   Population_size,
#   Log_population_size,
#   Treatment,
#   Baserate,
#   Log_baserate,
#   Covar1 (0/1),
#   Covar2 (1/2/3), and
#   Outcome.
# This simulated data can be modeled using a Negative binomial model.
# In order to perform an F-test of a linear combination of model parameters, first
# fit the necessary model, which in this example is a Negative Binomial:

nb_model_test <- MASS::glm.nb(Outcome ~ Treatment + Log_baserate + Covar1 +
   Covar2 + Treatment:Covar2 + Treatment:Covar1 +
   offset(Log_population_size), data = cluster_rct_data)

# For illustration of how to use `SAS_Lincom_Ftest()`, let's apply
# the HC3 (FIRORES) sandwich estimator and replicate the SAS Type-III tests for the
# variables in the model.

# Type III F-test for treatment parameter
L_treatment <- matrix(c(0, 1, 0, 0, 0, 0, 0.333333333333, 0.333333333333, 0.5),
                      nrow=1, ncol=9)
SAS_Lincom_Ftest(nb_model_test, L_treatment, "HC3")

# Type III F-test Log_baserate parameter
L_log_baserate <- matrix(c(0, 0, 1, 0, 0, 0, 0, 0, 0), nrow=1, ncol=9)
SAS_Lincom_Ftest(nb_model_test, L_log_baserate, "HC3")

# Type III F-test for covar1 parameter
L_covar1 <- matrix(c(0, 0, 0, 1, 0, 0, 0, 0, 0.5), nrow=1, ncol=9)
SAS_Lincom_Ftest(nb_model_test, L_covar1, "HC3")

# Type III F-test for covar2 parameter
L_value <- rbind(
  c(0, 0, 0, 0, 1, 0, 0.5,   0, 0),
```

```
  c(0, 0, 0, 0, 0, 1, 0,   0.5, 0)
)
SAS_Lincom_Ftest(nb_model_test, L_value, "HC3")

# Type III F-test for treatment by covar2 interaction
L_trt_by_covar2 <- rbind(
 c(0, 0, 0, 0, 0, 0, 1, 0, 0),
 c(0, 0, 0, 0, 0, 0, 0, 1, 0)
)
SAS_Lincom_Ftest(nb_model_test, L_trt_by_covar2, "HC3")

# Type III F-test for treatment by covar1 interaction
L_trt_by_covar1 <- matrix(c(0, 0, 0, 0, 0, 0, 0, 0, 1), nrow=1, ncol=9)
SAS_Lincom_Ftest(nb_model_test, L_trt_by_covar1, "HC3")
```

`TypeIIIAdj` *Small-sample adjustment for Type III test from Generalized Linear Models used in analysis of cluster randomized trial data*

Description

`TypeIIIAdj()` completes a small-sample adjustment on the Type III test from a `glm()` model or a model that inherits from `glm()`.

Usage

```
TypeIIIAdj(model_obj, type3_vec = NULL)
```

Arguments

`model_obj`	A model which is a `glm()` or inherits from `glm()`
`type3_vec`	A list of "variable = contrast form" pairs, one pair for each factor used in the glm model.

Details

The model object passed into `TypeIIIAdj()` must be a fitted `glm(0` model or a model that inherits from `glm()`. The small-sample adjustment process averages two Chi-square statistics for each parameter in the model and then calculates the p-value for the average Chi-square statistic. The first Chi-square statistic is taken after modifying the original model parameter estimate standard errors using the HC3 (FIRORES) sandwich variance estimator and the second Chi-square statistic is taken after modifying the original model parameter estimate standard errors using the HC2 (ROOT) sandwich variance estimator. These sandwich variance estimators are calculated via `sandwich::vcovHC()` using `type = "HC3"` and `type = "HC2"`. Type III Chi-square tests are completed using the `Anova()` function within the `car` package. The Anova argument settings are `type="III"`, `test.statistic="Wald"`,

Value

A data.frame of small-sample adjusted Type III test results from a glm. The data frame will have the following columns:

Variable Model parameter term

Df Degrees of freedom used in Type III Chi-square test of parameter estimates

ChiSq_firores Chi-square statistic for HC3 adjusted Type III test that model parameter terms associated with the variables equal to 0

ProbChisq_firores P-value from Wald type HC3 adjusted Chi-square Type III test

ChiSq_root Chi-square statistic for HC2 adjusted Type III test that model parameter terms associated with the variables equal to 0

ProbChisq_root P-value from Wald type HC2 adjusted Chi-square Type III test

SSAdjChisq Chi-square statistic for small-sample adjusted Type III test (average of ChiSq_firors and ChiSq_root) that model parameter terms associated with the Variable are equal to 0

SSAdjChisq_pval P-value from small-sample adjusted Chi-square Type III test (p-value when using SSAdjChisq)

Examples

```
# If small-sample adjustment is needed for a GLM model, then small-sample
# adjustment may need to be applied to the Type III tests. `TypeIIIAdj()`
# is the function that allows for this application.  Let's consider the
# simulated cluster randomized trial dataset, `cluster_rct_data`, which
# is a part of the `ClusterRandSSAdj` package.  This dataset has the
# following variables:
#   clusterID,
#   Population_size,
#   Log_population_size,
#   Treatment,
#   Baserate,
#   Log_baserate,
#   Covar1 (0/1),
#   Covar2 (1/2/3), and
#   Outcome.
# This simulated data can be modeled using a Negative binomial model.
# Before applying the small-sample adjustment on the Type III test,
# a model needs to be fitted.  Let's fit a Negative Binomial model
# using `glm.nb` from the `MASS` package:

nb_model_test <- MASS::glm.nb(Outcome ~ Treatment + Log_baserate + Covar1 +
   Covar2 + Treatment:Covar2 + Treatment:Covar1 +
   offset(Log_population_size), data = cluster_rct_data)

# Once the model is fit, `TypeIIIAdj()` can be applied to perform Type III
# tests. For Type III tests similar to those in SAS, a list of `contr.sum`
# contrast structures needs to be created, one element for each categorical
# variable. In general, this list of contrast structures needs to be
# defined according to user specifications.  If SAS like Type III tests
# are not desired, then the user can specify other available contrast
```

```
# structures such as `contr.treatment` or `contr.helmert`.  Next,
# `TypeIIIAdj()` can be used to perform the small-sample adjustment and
# obtain results similar to Type III test in SAS but based on chi-square
#tests instead of F-tests.
type3_lst <- list(Treatment = contr.sum, Covar1 = contr.sum, Covar2 = contr.sum)
TypeIIIAdj(nb_model_test, type3_lst)
```

Index