

# Package ‘clubSandwich’

April 1, 2025

**Title** Cluster-Robust (Sandwich) Variance Estimators with Small-Sample Corrections

**Version** 0.6.0

**Description** Provides several cluster-robust variance estimators (i.e., sandwich estimators) for ordinary and weighted least squares linear regression models, including the bias-reduced linearization estimator introduced by Bell and McCaffrey (2002) <<https://www150.statcan.gc.ca/n1/pub/12-001-x/2002002/article/9058-eng.pdf>> and developed further by Pustejovsky and Tipton (2017) <[DOI:10.1080/07350015.2016.1247004](https://doi.org/10.1080/07350015.2016.1247004)>. The package includes functions for estimating the variance-covariance matrix and for testing single- and multiple-contrast hypotheses based on Wald test statistics. Tests of single regression coefficients use Satterthwaite or saddle-point corrections. Tests of multiple-contrast hypotheses use an approximation to Hotelling's T-squared distribution. Methods are provided for a variety of fitted models, including `lm()` and `mlm` objects, `glm()`, `geeglm()` (from package 'geepack'), `ivreg()` (from package 'AER'), `ivreg()` (from package 'ivreg' when estimated by ordinary least squares), `plm()` (from package 'plm'), `gls()` and `lme()` (from 'nlme'), `lmer()` (from 'lme4'), `robu()` (from 'robumeta'), and `rma.uni()` and `rma.mv()` (from 'metafor').

**URL** <http://jepusto.github.io/clubSandwich/>

**BugReports** <https://github.com/jepusto/clubSandwich/issues>

**Depends** R (>= 3.0.0)

**License** GPL-3

**VignetteBuilder** knitr

**LazyData** true

**Imports** stats, sandwich, lifecycle

**Suggests** Formula, knitr, carData, geepack, metafor, metadat, robumeta, nlme, mlmRev, AER, plm (>= 1.6-4), Matrix, lme4, zoo, testthat, rmarkdown, covr, ivreg

**RoxygenNote** 7.3.2

**Encoding** UTF-8

**Language** en-US

**NeedsCompilation** no

**Author** James E. Pustejovsky [aut, cre]  
 (<<https://orcid.org/0000-0003-0591-9465>>),  
 Samuel Pekofsky [ctb],  
 Jingru Zhang [ctb]

**Maintainer** James E. Pustejovsky <jepusto@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-04-01 07:10:07 UTC

## Contents

AchievementAwardsRCT . . . . .	3
coef_test . . . . .	4
conf_int . . . . .	5
constraint_matrices . . . . .	7
dropoutPrevention . . . . .	8
findCluster.rma.mv . . . . .	10
impute_covariance_matrix . . . . .	10
linear_contrast . . . . .	13
MortalityRates . . . . .	15
pattern_covariance_matrix . . . . .	16
SATcoaching . . . . .	18
vcovCR . . . . .	19
vcovCR.geeglm . . . . .	22
vcovCR.glm . . . . .	23
vcovCR.gls . . . . .	25
vcovCR.ivreg . . . . .	26
vcovCR.lm . . . . .	28
vcovCR.lme . . . . .	29
vcovCR.lmerMod . . . . .	31
vcovCR.mlm . . . . .	32
vcovCR.plm . . . . .	34
vcovCR.rma.mv . . . . .	36
vcovCR.rma.uni . . . . .	37
vcovCR.robust . . . . .	39
Wald_test . . . . .	40

**Index** 43

---

AchievementAwardsRCT *Achievement Awards Demonstration program*

---

### Description

Data from a randomized trial of the Achievement Awards Demonstration program, reported in Angrist & Lavy (2009).

### Usage

AchievementAwardsRCT

### Format

A data frame with 16526 rows and 21 variables:

**school\_id** Fictitious school identification number

**school\_type** Factor identifying the school type (Arab religious, Jewish religious, Jewish secular)

**pair** Number of treatment pair. Note that 7 is a triple.

**treated** Indicator for whether school was in treatment group

**year** Cohort year

**student\_id** Fictitious student identification number

**sex** Factor identifying student sex

**siblings** Number of siblings

**immigrant** Indicator for immigrant status

**father\_ed** Father's level of education

**mother\_ed** Mother's level of education

**Bagrut\_status** Indicator for Bagrut attainment

**attempted** Number of Bagrut units attempted

**awarded** Number of Bagrut units awarded

**achv\_math** Indicator for satisfaction of math requirement

**achv\_english** Indicator for satisfaction of English requirement

**achv\_hebrew** Indicator for satisfaction of Hebrew requirement

**lagscore** Lagged Bagrut score

**qrtl** Quartile within distribution of lagscore, calculated by cohort and sex

**half** Lower or upper half within distribution of lagscore, calculated by cohort and sex

### Source

[Angrist Data Archive](#)

## References

Angrist, J. D., & Lavy, V. (2009). The effects of high stakes high school achievement awards : Evidence from a randomized trial. *American Economic Review*, 99(4), 1384-1414. doi:10.1257/aer.99.4.1384

---

 coef\_test

*Test all or selected regression coefficients in a fitted model*


---

## Description

coef\_test reports one- or two-sided t-tests for each coefficient estimate in a fitted linear regression model, using a sandwich estimator for the standard errors and (optionally) a small sample correction for the p-value. Available small-sample corrections include Satterthwaite approximation or a saddlepoint approximation. Coefficients can be tested against non-zero null values by specifying null\_constants.

## Usage

```
coef_test(
  obj,
  vcov,
  test = "Satterthwaite",
  alternative = c("two-sided", "greater", "less"),
  coefs = "All",
  null_constants = 0,
  p_values = TRUE,
  ...
)
```

## Arguments

obj	Fitted model for which to calculate t-tests.
vcov	Variance covariance matrix estimated using vcovCR or a character string specifying which small-sample adjustment should be used to calculate the variance-covariance.
test	Character vector specifying which small-sample corrections to calculate. "z" returns a z test (i.e., using a standard normal reference distribution). "naive-t" returns a t test with $m - 1$ degrees of freedom, where $m$ is the number of unique clusters. "naive-tp" returns a t test with $m - p$ degrees of freedom, where $p$ is the number of regression coefficients in obj. "Satterthwaite" returns a Satterthwaite correction. "saddlepoint" returns a saddlepoint correction. Default is "Satterthwaite".
alternative	Character string specifying the alternative hypothesis, with options "two-sided" (the default), "greater" or "less".
coefs	Character, integer, or logical vector specifying which coefficients should be tested. The default value "All" will test all estimated coefficients.

`null_constants` vector of null values for each coefficient to test. Must have length equal to the number of coefficients specified in `coefs`. Default is 0, in which case the null values are taken to be zero.

`p_values` Logical indicating whether to report p-values. The default value is TRUE.

... Further arguments passed to `vcovCR`, which are only needed if `vcov` is a character string.

### Value

A data frame containing estimated regression coefficients, standard errors, specified values of null hypotheses, and test results. For the Satterthwaite approximation, degrees of freedom and a p-value are reported. For the saddlepoint approximation, the saddlepoint and a p-value are reported.

### See Also

[vcovCR](#)

### Examples

```
data("ChickWeight", package = "datasets")
lm_fit <- lm(weight ~ Diet * Time, data = ChickWeight)
diet_index <- grepl("Diet.:Time", names(coef(lm_fit)))
coef_test(lm_fit, vcov = "CR2", cluster = ChickWeight$Chick, coefs = diet_index)

V_CR2 <- vcovCR(lm_fit, cluster = ChickWeight$Chick, type = "CR2")
coef_test(lm_fit, vcov = V_CR2, coefs = diet_index)

# non-inferiority test whether time-by-diet interaction effects are 2 or greater
coef_test(lm_fit, vcov = V_CR2, coefs = diet_index, null_constants = 2, alternative = "greater")
```

---

conf_int	<i>Calculate confidence intervals for all or selected regression coefficients in a fitted model</i>
----------	---

---

### Description

`conf_int` reports confidence intervals for each coefficient estimate in a fitted linear regression model, using a sandwich estimator for the standard errors and a small sample correction for the critical values. The small-sample correction is based on a Satterthwaite approximation.

### Usage

```
conf_int(
  obj,
  vcov,
  level = 0.95,
  test = "Satterthwaite",
```

```

  coefs = "All",
  ...,
  p_values = FALSE
)

```

### Arguments

obj	Fitted model for which to calculate confidence intervals.
vcov	Variance covariance matrix estimated using <code>vcovCR</code> or a character string specifying which small-sample adjustment should be used to calculate the variance-covariance.
level	Desired coverage level for confidence intervals.
test	Character vector specifying which small-sample corrections to calculate. "z" returns a z test (i.e., using a standard normal reference distribution). "naive-t" returns a t test with $m - 1$ degrees of freedom, where $m$ is the number of unique clusters. "naive-tp" returns a t test with $m - p$ degrees of freedom, where $p$ is the number of regression coefficients in <code>obj</code> . "Satterthwaite" returns a Satterthwaite correction. Unlike in <code>coef_test()</code> , "saddlepoint" is not currently supported in <code>conf_int()</code> because saddlepoint confidence intervals do not have a closed-form solution.
coefs	Character, integer, or logical vector specifying which coefficients should be tested. The default value "All" will test all estimated coefficients.
...	Further arguments passed to <code>vcovCR</code> , which are only needed if <code>vcov</code> is a character string.
p_values	Logical indicating whether to report p-values. The default value is TRUE.

### Value

A data frame containing estimated regression coefficients, standard errors, confidence intervals, and (optionally) p-values.

### See Also

[vcovCR](#)

### Examples

```

data("ChickWeight", package = "datasets")
lm_fit <- lm(weight ~ Diet * Time, data = ChickWeight)
diet_index <- grepl("Diet.:Time", names(coef(lm_fit)))
conf_int(lm_fit, vcov = "CR2", cluster = ChickWeight$Chick, coefs = diet_index)

V_CR2 <- vcovCR(lm_fit, cluster = ChickWeight$Chick, type = "CR2")
conf_int(lm_fit, vcov = V_CR2, level = .99, coefs = diet_index)

```

---

constraint\_matrices    *Create constraint matrices*

---

### Description

Helper functions to create common types of constraint matrices, for use with [Wald\\_test](#) to conduct Wald-type tests of linear contrasts from a fitted regression model.

### Usage

```
constrain_zero(constraints, coefs, reg_ex = FALSE)

constrain_equal(constraints, coefs, reg_ex = FALSE)

constrain_pairwise(constraints, coefs, reg_ex = FALSE, with_zero = FALSE)
```

### Arguments

constraints	Set of constraints to test. Can be logical (using TRUE to specify which coefficients to constrain), integer (specify the index of coefficients to constrain), character (specify the names of the coefficients to constrain), or a regular expression.
coefs	Vector of coefficient estimates, used to determine the column dimension of the constraint matrix. Can be omitted if the function is called inside <code>Wald_test()</code> .
reg_ex	Logical indicating whether constraints should be interpreted as a regular expression. Defaults to FALSE.
with_zero	Logical indicating whether coefficients should also be compared to zero. Defaults to FALSE.

### Details

Constraints can be specified as character vectors, regular expressions (with `reg_ex = TRUE`), integer vectors, or logical vectors.

`constrain_zero()` Creates a matrix that constrains a specified set of coefficients to all be equal to zero.

`constrain_equal()` Creates a matrix that constrains a specified set of coefficients to all be equal.

`constrain_pairwise()` Creates a list of constraint matrices consisting of all pairwise comparisons between a specified set of coefficients. If `with_zero = TRUE`, then the list will also include a set of constraint matrices comparing each coefficient to zero.

### Value

A matrix or list of matrices encoding the specified set of constraints.

### See Also

[Wald\\_test](#)

**Examples**

```

if (requireNamespace("carData", quietly = TRUE)) withAutoprint({

  data(Duncan, package = "carData")
  Duncan$cluster <- sample(LETTERS[1:8], size = nrow(Duncan), replace = TRUE)

  Duncan_fit <- lm(prestige ~ 0 + type + income + type:income + type:education, data=Duncan)
  # Note that type:income terms are interactions because main effect of income is included
  # but type:education terms are separate slopes for each unique level of type

  Duncan_coefs <- coef(Duncan_fit)

  # The following are all equivalent
  constrain_zero(constraints = c("typeprof:income", "typewc:income"),
                 coefs = Duncan_coefs)
  constrain_zero(constraints = ":income", coefs = Duncan_coefs,
                 reg_ex = TRUE)
  constrain_zero(constraints = 5:6, coefs = Duncan_coefs)
  constrain_zero(constraints = c(FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, FALSE, FALSE, FALSE),
                 coefs = Duncan_coefs)

  # The following are all equivalent
  constrain_equal(c("typebc:education", "typeprof:education", "typewc:education"),
                 Duncan_coefs)
  constrain_equal(":education", Duncan_coefs, reg_ex = TRUE)
  constrain_equal(7:9, Duncan_coefs)
  constrain_equal(c(FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE),
                 Duncan_coefs)

  # Test pairwise equality of the education slopes
  constrain_pairwise(":education", Duncan_coefs,
                    reg_ex = TRUE)

  # Test pairwise equality of the income slopes, plus compare against zero
  constrain_pairwise(":income", Duncan_coefs,
                    reg_ex = TRUE, with_zero = TRUE)

})

```

---

 dropoutPrevention

*Dropout prevention/intervention program effects*


---

**Description**

A dataset containing estimated effect sizes, variances, and covariates from a meta-analysis of dropout prevention/intervention program effects, conducted by Wilson et al. (2011). Missing observations were imputed.



**Usage**

dropoutPrevention

**Format**

A data frame with 385 rows and 18 variables:

**LOR1** log-odds ratio measuring the intervention effect

**varLOR** estimated sampling variance of the log-odds ratio

**studyID** unique identifier for each study

**studySample** unique identifier for each sample within a study

**study\_design** study design (randomized, matched, or non-randomized and unmatched)

**outcome** outcome measure for the intervention effect is estimated (school dropout, school enrollment, graduation, graduation or GED receipt)

**evaluator\_independence** degree of evaluator independence (independent, indirect but influential, involved in planning but not delivery, involved in delivery)

**implementation\_quality** level of implementation quality (clear problems, possible problems, no apparent problems)

**program\_site** Program delivery site (community, mixed, school classroom, school but outside of classroom)

**attrition** Overall attrition (proportion)

**group\_equivalence** pretest group-equivalence log-odds ratio

**adjusted** adjusted or unadjusted data used to calculate intervention effect

**male\_pct** proportion of the sample that is male

**white\_pct** proportion of the sample that is white

**average\_age** average age of the sample

**duration** program duration (in weeks)

**service\_hrs** program contact hours per week

**big\_study** indicator for the 32 studies with 3 or more effect sizes

**Source**

Wilson, S. J., Lipsey, M. W., Tanner-Smith, E., Huang, C. H., & Steinka-Fry, K. T. (2011). Dropout prevention and intervention programs: Effects on school completion and dropout Among school-aged children and youth: A systematic review. *Campbell Systematic Reviews*, 7\_(1), 1-61. doi:10.4073/csr.2011.8

**References**

Wilson, S. J., Lipsey, M. W., Tanner-Smith, E., Huang, C. H., & Steinka-Fry, K. T. (2011). Dropout prevention and intervention programs: Effects on school completion and dropout Among school-aged children and youth: A systematic review. *Campbell Systematic Reviews*, 7\_(1), 1-61. doi:10.4073/csr.2011.8

Tipton, E., & Pustejovsky, J. E. (2015). Small-sample adjustments for tests of moderators and model fit using robust variance estimation in meta-regression. *Journal of Educational and Behavioral Statistics*, 40\_(6), 604-634. doi:10.3102/1076998615606099

---

`findCluster.rma.mv`      *Detect cluster structure of an rma.mv object*

---

### Description

`findCluster.rma.mv` returns a vector of ID variables for the highest level of clustering in a fitted `rma.mv` model.

### Usage

```
findCluster.rma.mv(obj)
```

### Arguments

`obj`                    A fitted `rma.mv` object.

### Value

A vector of ID variables for the highest level of clustering in `obj`.

### Examples

```
if (requireNamespace("metafor", quietly = TRUE)) {  
  
  library(metafor)  
  data(dat.assink2016, package = "metadat")  
  
  mfor_fit <- rma.mv(yi ~ year + deltype,  
                   V = vi, random = ~ 1 | study / esid,  
                   data = dat.assink2016)  
  
  findCluster.rma.mv(mfor_fit)  
  
}
```

---

`impute_covariance_matrix`

*Impute a block-diagonal covariance matrix*

---

**Description**

‘r lifecycle::badge("superseded")‘

This function is superseded by the `vcalc` provided by the `metafor` package. Compared to `impute_covariance_matrix`, `vcalc` provides many further features, includes a `data` argument, and uses syntax that is consistent with other functions in `metafor`.

`impute_covariance_matrix` calculates a block-diagonal covariance matrix, given the marginal variances, the block structure, and an assumed correlation structure. Can be used to create compound-symmetric structures, AR(1) auto-correlated structures, or combinations thereof.

**Usage**

```
impute_covariance_matrix(
  vi,
  cluster,
  r,
  ti,
  ar1,
  smooth_vi = FALSE,
  subgroup = NULL,
  return_list = identical(as.factor(cluster), sort(as.factor(cluster))),
  check_PD = TRUE
)
```

**Arguments**

<code>vi</code>	Vector of variances
<code>cluster</code>	Vector indicating which effects belong to the same cluster. Effects with the same value of ‘cluster’ will be treated as correlated.
<code>r</code>	Vector or numeric value of assumed constant correlation(s) between effect size estimates from each study.
<code>ti</code>	Vector of time-points describing temporal spacing of effects, for use with autoregressive correlation structures.
<code>ar1</code>	Vector or numeric value of assumed AR(1) auto-correlation(s) between effect size estimates from each study. If specified, then <code>ti</code> argument must be specified.
<code>smooth_vi</code>	Logical indicating whether to smooth the marginal variances by taking the average <code>vi</code> within each cluster. Defaults to <code>FALSE</code> .
<code>subgroup</code>	Vector of category labels describing sub-groups of effects. If non-null, effects that share the same category label and the same cluster will be treated as correlated, but effects with different category labels will be treated as uncorrelated, even if they come from the same cluster.
<code>return_list</code>	Optional logical indicating whether to return a list of matrices (with one entry per block) or the full variance-covariance matrix.
<code>check_PD</code>	Optional logical indicating whether to check whether each covariance matrix is positive definite. If <code>TRUE</code> (the default), the function will display a warning if any covariance matrix is not positive definite.

## Details

A block-diagonal variance-covariance matrix (possibly represented as a list of matrices) with a specified structure. The structure depends on whether the `r` argument, `ar1` argument, or both arguments are specified. Let  $v_{ij}$  denote the specified variance for effect  $i$  in cluster  $j$  and  $C_{hij}$  be the covariance between effects  $h$  and  $i$  in cluster  $j$ .

- If only `r` is specified, each block of the variance-covariance matrix will have a constant (compound symmetric) correlation, so that

$$C_{hij} = r_j \sqrt{v_{hj} v_{ij}},$$

where  $r_j$  is the specified correlation for cluster  $j$ . If only a single value is given in `r`, then it will be used for every cluster.

- If only `ar1` is specified, each block of the variance-covariance matrix will have an AR(1) auto-correlation structure, so that

$$C_{hij} = \phi_j^{|t_{hj} - t_{ij}|} \sqrt{v_{hj} v_{ij}},$$

where  $\phi_j$  is the specified auto-correlation for cluster  $j$  and  $t_{hj}$  and  $t_{ij}$  are specified time-points corresponding to effects  $h$  and  $i$  in cluster  $j$ . If only a single value is given in `ar1`, then it will be used for every cluster.

- If both `r` and `ar1` are specified, each block of the variance-covariance matrix will have combination of compound symmetric and an AR(1) auto-correlation structures, so that

$$C_{hij} = \left[ r_j + (1 - r_j) \phi_j^{|t_{hj} - t_{ij}|} \right] \sqrt{v_{hj} v_{ij}},$$

where  $r_j$  is the specified constant correlation for cluster  $j$ ,  $\phi_j$  is the specified auto-correlation for cluster  $j$  and  $t_{hj}$  and  $t_{ij}$  are specified time-points corresponding to effects  $h$  and  $i$  in cluster  $j$ . If only single values are given in `r` or `ar1`, they will be used for every cluster.

If `smooth_vi = TRUE`, then all of the variances within cluster  $j$  will be set equal to the average variance of cluster  $j$ , i.e.,

$$v'_{ij} = \frac{1}{n_j} \sum_{i=1}^{n_j} v_{ij}$$

for  $i = 1, \dots, n_j$  and  $j = 1, \dots, k$ .

## Value

If `cluster` is appropriately sorted, then a list of matrices, with one entry per cluster, will be returned by default. If `cluster` is out of order, then the full variance-covariance matrix will be returned by default. The output structure can be controlled with the optional `return_list` argument.

## Examples

```
if (requireNamespace("metafor", quietly = TRUE)) {
  library(metafor)

  # Constant correlation
```

```

data(SATcoaching)
V_list <- impute_covariance_matrix(vi = SATcoaching$V, cluster = SATcoaching$study, r = 0.66)
MVFE <- rma.mv(d ~ 0 + test, V = V_list, data = SATcoaching)
conf_int(MVFE, vcov = "CR2", cluster = SATcoaching$study)

}

```

---

linear_contrast	<i>Calculate confidence intervals and p-values for linear contrasts of regression coefficients in a fitted model</i>
-----------------	--

---

### Description

linear\_contrast reports confidence intervals and (optionally) p-values for linear contrasts of regression coefficients from a fitted model, using a sandwich estimator for the standard errors and (optionally) a small sample correction for the critical values. The default small-sample correction is based on a Satterthwaite approximation.

### Usage

```

linear_contrast(
  obj,
  vcov,
  contrasts,
  level = 0.95,
  test = "Satterthwaite",
  ...,
  p_values = FALSE,
  adjustment_method = "none"
)

```

### Arguments

obj	Fitted model for which to calculate confidence intervals.
vcov	Variance covariance matrix estimated using vcovCR or a character string specifying which small-sample adjustment should be used to calculate the variance-covariance.
contrasts	A contrast matrix, or a list of multiple contrast matrices to test. See details and examples.
level	Desired coverage level for confidence intervals.
test	Character vector specifying which small-sample corrections to calculate. "z" returns a z test (i.e., using a standard normal reference distribution). "naive-t" returns a t test with $m-1$ degrees of freedom, where $m$ is the number of unique clusters. "naive-tp" returns a t test with $m-p$ degrees of freedom, where $p$ is the number of regression coefficients in obj. "Satterthwaite" returns a Satterthwaite correction. Unlike in coef_test(), "saddlepoint" is not currently

supported in `conf_int()` because saddlepoint confidence intervals do not have a closed-form solution.

... Further arguments passed to `vcovCR`, which are only needed if `vcov` is a character string.

`p_values` Logical indicating whether to report p-values. The default value is `FALSE`.

`adjustment_method` Correction method, a `character` string from `p.adjust.methods`, which is passed to `p.adjust` to correct p-values in the case of multiple comparisons. Defaults to `"none"`. Any value given will be ignored if `p_values = FALSE`.

### Details

Constraints can be specified directly as  $q \times p$  matrices or indirectly through `constrain_pairwise`, `constrain_equal`, or `constrain_zero`.

### Value

A data frame containing estimated contrasts, standard errors, confidence intervals, and (optionally) p-values.

### See Also

[vcovCR](#)

### Examples

```
data("ChickWeight", package = "datasets")
lm_fit <- lm(weight ~ 0 + Diet + Time:Diet, data = ChickWeight)

# Pairwise comparisons of diet-by-time slopes
linear_contrast(lm_fit, vcov = "CR2", cluster = ChickWeight$Chick,
               contrasts = constrain_pairwise("Diet.:Time", reg_ex = TRUE))

if (requireNamespace("carData", quietly = TRUE)) withAutoprint({
  data(Duncan, package = "carData")
  Duncan$cluster <- sample(LETTERS[1:8], size = nrow(Duncan), replace = TRUE)

  Duncan_fit <- lm(prestige ~ 0 + type + income + type:income + type:education, data=Duncan)
  # Note that type:income terms are interactions because main effect of income is included
  # but type:education terms are separate slopes for each unique level of type

  # Pairwise comparisons of type-by-education slopes
  linear_contrast(Duncan_fit, vcov = "CR2", cluster = Duncan$cluster,
                contrasts = constrain_pairwise(":education", reg_ex = TRUE),
                test = "Satterthwaite")

  # Repeat the above, but with p-values displayed and multiple comparisons adjustment applied.
  linear_contrast(Duncan_fit, vcov = "CR2", cluster = Duncan$cluster,
                contrasts = constrain_pairwise(":education", reg_ex = TRUE),
```

```

        test = "Satterthwaite", p_values = TRUE, adjustment_method = "hochberg")

# Pairwise comparisons of type-by-income interactions
linear_contrast(Duncan_fit, vcov = "CR2", cluster = Duncan$cluster,
                contrasts = constrain_pairwise(":income", reg_ex = TRUE, with_zero = TRUE),
                test = "Satterthwaite")

})

```

---

MortalityRates

*State-level annual mortality rates by cause among 18-20 year-olds*


---

### Description

A dataset containing state-level annual mortality rates for select causes of death, as well as data related to the minimum legal drinking age and alcohol consumption.

### Usage

```
MortalityRates
```

### Format

A data frame with 5508 rows and 12 variables:

**year** Year of observation

**state** identifier for state

**count** Number of deaths

**pop** Population size

**legal** Proportion of 18-20 year-old population that is legally allowed to drink

**beertaxa** Beer taxation rate

**beerpercap** Beer consumption per capita

**winepercap** Wine consumption per capita

**spiritpercap** Spirits consumption per capita

**totpercap** Total alcohol consumption per capita

**mrate** Mortality rate per 10,000

**cause** Cause of death

### Source

[Mastering 'Metrics data archive](#)

## References

- Angrist, J. D., and Pischke, J. S. (2014). *Mastering metrics: the path from cause to effect*. Princeton University Press, 2014.
- Carpenter, C., & Dobkin, C. (2011). The minimum legal drinking age and public health. *Journal of Economic Perspectives*, 25\_(2), 133-156. doi:10.1257/jep.25.2.133

---

pattern\_covariance\_matrix

*Impute a patterned block-diagonal covariance matrix*

---

## Description

‘r lifecycle::badge("superseded")’

This function is superseded by the `vcalc` provided by the `metafor` package. Compared to `pattern_covariance_matrix`, `vcalc` provides many further features, includes a `data` argument, and uses syntax that is consistent with other functions in `metafor`.

`pattern_covariance_matrix` calculates a block-diagonal covariance matrix, given the marginal variances, the block structure, and an assumed correlation structure defined by a patterned correlation matrix.

## Usage

```
pattern_covariance_matrix(
  vi,
  cluster,
  pattern_level,
  r_pattern,
  r,
  smooth_vi = FALSE,
  subgroup = NULL,
  return_list = identical(as.factor(cluster), sort(as.factor(cluster))),
  check_PD = TRUE
)
```

## Arguments

- |                            |  |
|----------------------------|--|
| <code>vi</code>            | Vector of variances  |
| <code>cluster</code>       | Vector indicating which effects belong to the same cluster. Effects with the same value of ‘cluster’ will be treated as correlated.  |
| <code>pattern_level</code> | Vector of categories for each effect size, used to determine which entry of the pattern matrix will be used to impute a correlation. |
| <code>r_pattern</code>     | Patterned correlation matrix with row and column names corresponding to the levels of pattern.                                       |
| <code>r</code>             | Vector or numeric value of assumed constant correlation(s) between effect size estimates from each study.                            |



smooth_vi	Logical indicating whether to smooth the marginal variances by taking the average $v_i$ within each cluster. Defaults to FALSE.
subgroup	Vector of category labels describing sub-groups of effects. If non-null, effects that share the same category label and the same cluster will be treated as correlated, but effects with different category labels will be treated as uncorrelated, even if they come from the same cluster.
return_list	Optional logical indicating whether to return a list of matrices (with one entry per block) or the full variance-covariance matrix.
check_PD	Optional logical indicating whether to check whether each covariance matrix is positive definite. If TRUE (the default), the function will display a warning if any covariance matrix is not positive definite.

## Details

A block-diagonal variance-covariance matrix (possibly represented as a list of matrices) with a specified correlation structure, defined by a patterned correlation matrix. Let  $v_{ij}$  denote the specified variance for effect  $i$  in cluster  $j$  and  $C_{hij}$  be the covariance between effects  $h$  and  $i$  in cluster  $j$ . Let  $p_{ij}$  be the level of the pattern variable for effect  $i$  in cluster  $j$ , taking a value in  $1, \dots, C$ . A patterned correlation matrix is defined as a set of correlations between pairs of effects taking each possible combination of patterns. Formally, let  $r_{cd}$  be the correlation between effects in categories  $c$  and  $d$ , respectively, where  $r_{cd} = r_{dc}$ . Then the covariance between effects  $h$  and  $i$  in cluster  $j$  is taken to be

$$C_{hij} = \sqrt{v_{hj}v_{ij}} \times r_{p_{hj}p_{ij}}.$$

Correlations between effect sizes within the same category are defined by the diagonal values of the pattern matrix, which may take values less than one.

Combinations of pattern levels that do not occur in the patterned correlation matrix will be set equal to  $r$ .

If `smooth_vi = TRUE`, then all of the variances within cluster  $j$  will be set equal to the average variance of cluster  $j$ , i.e.,

$$v'_{ij} = \frac{1}{n_j} \sum_{i=1}^{n_j} v_{ij}$$

for  $i = 1, \dots, n_j$  and  $j = 1, \dots, k$ .

## Value

If `cluster` is appropriately sorted, then a list of matrices, with one entry per cluster, will be returned by default. If `cluster` is out of order, then the full variance-covariance matrix will be returned by default. The output structure can be controlled with the optional `return_list` argument.

## Examples

```
pkgs_available <-
  requireNamespace("metafor", quietly = TRUE) &
  requireNamespace("robumeta", quietly = TRUE)

if (pkgs_available) {
```

```

library(metafor)

data(oswald2013, package = "robumeta")
dat <- escalc(data = oswald2013, measure = "ZCOR", ri = R, ni = N)
subset_ids <- unique(dat$Study)[1:20]
dat <- subset(dat, Study %in% subset_ids)

# make a patterned correlation matrix

p_levels <- levels(dat$Crit.Cat)
r_pattern <- 0.7^as.matrix(dist(1:length(p_levels)))
diag(r_pattern) <- seq(0.75, 0.95, length.out = 6)
rownames(r_pattern) <- colnames(r_pattern) <- p_levels

# impute the covariance matrix using patterned correlations
V_list <- pattern_covariance_matrix(vi = dat$vi,
                                   cluster = dat$Study,
                                   pattern_level = dat$Crit.Cat,
                                   r_pattern = r_pattern,
                                   smooth_vi = TRUE)

# fit a model using imputed covariance matrix

MVFE <- rma.mv(yi ~ 0 + Crit.Cat, V = V_list,
              random = ~ Crit.Cat | Study,
              data = dat)

conf_int(MVFE, vcov = "CR2")

}

```

---

 SATcoaching

*Randomized experiments on SAT coaching*


---

### Description

Effect sizes from studies on the effects of SAT coaching, reported in Kalaian and Raudenbush (1996)

### Usage

SATcoaching

### Format

A data frame with 67 rows and 11 variables:

**study** Study identifier

**year** Year of publication

- test** Character string indicating whether effect size corresponds to outcome on verbal (SATV) or math (SATM) test
- d** Effect size estimate (Standardized mean difference)
- V** Variance of effect size estimate
- nT** Sample size in treatment condition
- nC** Sample size in control condition
- study\_type** Character string indicating whether study design used a matched, non-equivalent, or randomized control group
- hrs** Hours of coaching
- ETS** Indicator variable for Educational Testing Service
- homework** Indicator variable for homework

## References

Kalaian, H. A. & Raudenbush, S. W. (1996). A multivariate mixed linear model for meta-analysis. *Psychological Methods*, 1(3), 227-235. doi:[10.1037/1082989X.1.3.227](https://doi.org/10.1037/1082989X.1.3.227)

---

vcovCR

*Cluster-robust variance-covariance matrix*

---

## Description

This is a generic function, with specific methods defined for `lm`, `plm`, `glm`, `gls`, `lme`, `robu`, `rma.uni`, and `rma.mv` objects.

`vcovCR` returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates.

## Usage

```
vcovCR(obj, cluster, type, target, inverse_var, form, ...)
```

```
## Default S3 method:
vcovCR(
  obj,
  cluster,
  type,
  target = NULL,
  inverse_var = FALSE,
  form = "sandwich",
  ...
)
```

**Arguments**

<code>obj</code>	Fitted model for which to calculate the variance-covariance matrix
<code>cluster</code>	Expression or vector indicating which observations belong to the same cluster. For some classes, the cluster will be detected automatically if not specified.
<code>type</code>	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
<code>target</code>	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If a vector, the target matrix is assumed to be diagonal. If not specified, <code>vcovCR</code> will attempt to infer a value.
<code>inverse_var</code>	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, <code>vcovCR</code> will attempt to infer a value.
<code>form</code>	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting <code>form = "meat"</code> will return only the meat of the sandwich and setting <code>form = B</code> , where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. <code>form = "estfun"</code> will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.
<code>...</code>	Additional arguments available for some classes of objects.

**Details**

`vcovCR` returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates.

Several different small sample corrections are available, which run parallel with the "HC" corrections for heteroskedasticity-consistent variance estimators, as implemented in [vcovHC](#). The "CR2" adjustment is recommended (Pustejovsky & Tipton, 2017; Imbens & Kolesar, 2016). See Pustejovsky and Tipton (2017) and Cameron and Miller (2015) for further technical details. Available options include:

"**CR0**" is the original form of the sandwich estimator (Liang & Zeger, 1986), which does not make any small-sample correction.

"**CR1**" multiplies CR0 by  $m / (m - 1)$ , where  $m$  is the number of clusters.

"**CR1p**" multiplies CR0 by  $m / (m - p)$ , where  $m$  is the number of clusters and  $p$  is the number of covariates.

"**CR1S**" multiplies CR0 by  $(m(N-1)) / [(m-1)(N-p)]$ , where  $m$  is the number of clusters,  $N$  is the total number of observations, and  $p$  is the number of covariates. Some Stata commands use this correction by default.

"**CR2**" is the "bias-reduced linearization" adjustment proposed by Bell and McCaffrey (2002) and further developed in Pustejovsky and Tipton (2017). The adjustment is chosen so that the variance-covariance estimator is exactly unbiased under a user-specified working model.

"**CR3**" approximates the leave-one-cluster-out jackknife variance estimator (Bell & McCaffrey, 2002).

## Value

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates. The matrix has several attributes:

**type** indicates which small-sample adjustment was used

**cluster** contains the factor vector that defines independent clusters

**bread** contains the bread matrix

**v\_scale** constant used in scaling the sandwich estimator

**est\_mats** contains a list of estimating matrices used to calculate the sandwich estimator

**adjustments** contains a list of adjustment matrices used to calculate the sandwich estimator

**target** contains the working variance-covariance model used to calculate the adjustment matrices. This is needed for calculating small-sample corrections for Wald tests.

## References

Bell, R. M., & McCaffrey, D. F. (2002). Bias reduction in standard errors for linear regression with multi-stage samples. *Survey Methodology*, 28(2), 169-181.

Cameron, A. C., & Miller, D. L. (2015). A Practitioner's Guide to Cluster-Robust Inference. *Journal of Human Resources*, 50(2), 317-372. doi:10.3368/jhr.50.2.317

Imbens, G. W., & Kolesar, M. (2016). Robust standard errors in small samples: Some practical advice. *Review of Economics and Statistics*, 98(4), 701-712. doi:10.1162/rest\_a\_00552

Liang, K.-Y., & Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1), 13-22. doi:10.1093/biomet/73.1.13

Pustejovsky, J. E. & Tipton, E. (2018). Small sample methods for cluster-robust variance estimation and hypothesis testing in fixed effects models. *Journal of Business and Economic Statistics*, 36(4), 672-683. doi:10.1080/07350015.2016.1247004

## See Also

`vcovCR.lm`, `vcovCR.plm`, `vcovCR.glm`, `vcovCR.gls`, `vcovCR.lme`, `vcovCR.lmerMod`, `vcovCR.robust`, `vcovCR.rma.uni`, `vcovCR.rma.mv`

## Examples

```
# simulate design with cluster-dependence
m <- 8
cluster <- factor(rep(LETTERS[1:m], 3 + rpois(m, 5)))
n <- length(cluster)
X <- matrix(rnorm(3 * n), n, 3)
nu <- rnorm(m)[cluster]
e <- rnorm(n)
y <- X %*% c(.4, .3, -.3) + nu + e
dat <- data.frame(y, X, cluster, row = 1:n)

# fit linear model
lm_fit <- lm(y ~ X1 + X2 + X3, data = dat)
```

```
vcov(lm_fit)

# cluster-robust variance estimator with CR2 small-sample correction
vcovCR(lm_fit, cluster = dat$cluster, type = "CR2")

# compare small-sample adjustments
CR_types <- paste0("CR",c("0", "1", "1S", "2", "3"))
sapply(CR_types, function(type)
  sqrt(diag(vcovCR(lm_fit, cluster = dat$cluster, type = type))))
```

---

vcovCR.geeglm

*Cluster-robust variance-covariance matrix for a geeglm object.*


---

### Description

vcovCR returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from an [geeglm](#) object.

### Usage

```
## S3 method for class 'geeglm'
vcovCR(
  obj,
  cluster,
  type,
  target = NULL,
  inverse_var = NULL,
  form = "sandwich",
  ...
)
```

### Arguments

obj	Fitted model for which to calculate the variance-covariance matrix
cluster	Expression or vector indicating which observations belong to the same cluster. Required for geeglm objects.
type	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
target	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If a vector, the target matrix is assumed to be diagonal. If not specified, the target is taken to be the estimated variance function.
inverse_var	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, vcovCR will attempt to infer a value.

`form` Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting `form = "meat"` will return only the meat of the sandwich and setting `form = B`, where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. `form = "estfun"` will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.

`...` Additional arguments available for some classes of objects.

**Value**

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

**See Also**

[vcovCR](#)

**Examples**

```
if (requireNamespace("geepack", quietly = TRUE)) {
  library(geepack)
  data(dietox, package = "geepack")
  dietox$Cu <- as.factor(dietox$Cu)
  mf <- formula(Weight ~ Cu * (Time + I(Time^2) + I(Time^3)))
  gee1 <- geeglm(mf, data=dietox, id=Pig, family=poisson("identity"), corstr="ar1")
  V_CR <- vcovCR(gee1, cluster = dietox$Pig, type = "CR2")
  coef_test(gee1, vcov = V_CR, test = "Satterthwaite")
}
```

---

vcovCR.glm

*Cluster-robust variance-covariance matrix for a glm object.*

---

**Description**

`vcovCR` returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from an `glm` object.

**Usage**

```
## S3 method for class 'glm'
vcovCR(
  obj,
  cluster,
  type,
```

```

    target = NULL,
    inverse_var = NULL,
    form = "sandwich",
    ...
  )

```

## Arguments

obj	Fitted model for which to calculate the variance-covariance matrix
cluster	Expression or vector indicating which observations belong to the same cluster. Required for glm objects.
type	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
target	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If a vector, the target matrix is assumed to be diagonal. If not specified, the target is taken to be the estimated variance function.
inverse_var	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, vcovCR will attempt to infer a value.
form	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting form = "meat" will return only the meat of the sandwich and setting form = B, where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. form = "estfun" will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.
...	Additional arguments available for some classes of objects.

## Value

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

## See Also

[vcovCR](#)

## Examples

```

if (requireNamespace("geepack", quietly = TRUE)) {
  data(dietox, package = "geepack")
  dietox$Cu <- as.factor(dietox$Cu)
  weight_fit <- glm(Weight ~ Cu * poly(Time, 3), data=dietox, family = "quasipoisson")
  V_CR <- vcovCR(weight_fit, cluster = dietox$Pig, type = "CR2")
  coef_test(weight_fit, vcov = V_CR, test = "Satterthwaite")
}

```



```
}

```

---

vcovCR.gls

*Cluster-robust variance-covariance matrix for a gls object.*


---

## Description

vcovCR returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from a [gls](#) object.

## Usage

```
## S3 method for class 'gls'
vcovCR(obj, cluster, type, target, inverse_var, form = "sandwich", ...)
```

## Arguments

obj	Fitted model for which to calculate the variance-covariance matrix
cluster	Optional expression or vector indicating which observations belong to the same cluster. If not specified, will be set to <code>getGroups(obj)</code> .
type	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
target	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If not specified, the target is taken to be the estimated variance-covariance structure of the <code>gls</code> object.
inverse_var	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, <code>vcovCR</code> will attempt to infer a value.
form	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting <code>form = "meat"</code> will return only the meat of the sandwich and setting <code>form = B</code> , where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. <code>form = "estfun"</code> will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.
...	Additional arguments available for some classes of objects.

## Value

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

## See Also

[vcovCR](#)

## Examples

```
if (requireNamespace("nlme", quietly = TRUE)) {

  library(nlme)
  data(Ovary, package = "nlme")
  Ovary$time_int <- 1:nrow(Ovary)
  lm_AR1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), data = Ovary,
               correlation = corAR1(form = ~ time_int | Mare))
  vcovCR(lm_AR1, type = "CR2")

}
```

---

 vcovCR.ivreg

---

*Cluster-robust variance-covariance matrix for an ivreg object.*


---

## Description

vcovCR returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from an ivreg object fitted from the **AER** package or the **ivreg** package.

## Usage

```
## S3 method for class 'ivreg'
vcovCR(
  obj,
  cluster,
  type,
  target = NULL,
  inverse_var = FALSE,
  form = "sandwich",
  ...
)
```

## Arguments

obj	Fitted model for which to calculate the variance-covariance matrix
cluster	Expression or vector indicating which observations belong to the same cluster. Required for ivreg objects.
type	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
target	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If a vector, the target matrix is assumed to be diagonal. If not specified, the target is taken to be an identity matrix.

inverse_var	Not used for ivreg objects.
form	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting form = "meat" will return only the meat of the sandwich and setting form = B, where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. form = "estfun" will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.
...	Additional arguments available for some classes of objects.

### Details

For any "ivreg" objects fitted via the `ivreg` function from the `ivreg` package, only traditional 2SLS regression method (method = "OLS") is supported. `clubSandwich` currently cannot support robust-regression methods such as M-estimation (method = "M") or MM-estimation (method = "MM").

### Value

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

### See Also

[vcovCR](#)

### Examples

```
if (requireNamespace("AER", quietly = TRUE)) withAutoprint({

  library(AER)
  data("CigarettesSW")
  Cigs <- within(CigarettesSW, {
    rprice <- price/cpi
    rincome <- income/population/cpi
    tdiff <- (taxs - tax)/cpi
  })

  iv_fit_AER <- AER::ivreg(log(packs) ~ log(rprice) + log(rincome) |
    log(rincome) + tdiff + I(tax/cpi), data = Cigs)
  vcovCR(iv_fit_AER, cluster = Cigs$state, type = "CR2")
  coef_test(iv_fit_AER, vcov = "CR2", cluster = Cigs$state)

})

pkgs_available <-
  requireNamespace("AER", quietly = TRUE) &
  requireNamespace("ivreg", quietly = TRUE)

if (pkgs_available) withAutoprint ({

  data("CigarettesSW")
```

```

Cigs <- within(CigarettesSW, {
  rprice <- price/cpi
  rincome <- income/population/cpi
  tdiff <- (taxes - tax)/cpi
})
iv_fit_ivreg <- ivreg::ivreg(log(packs) ~ log(rprice) + log(rincome) |
  log(rincome) + tdiff + I(tax/cpi), data = Cigs)
vcovCR(iv_fit_ivreg, cluster = Cigs$state, type = "CR2")
coef_test(iv_fit_ivreg, vcov = "CR2", cluster = Cigs$state)
})

```

---

vcovCR.lm

*Cluster-robust variance-covariance matrix for an lm object.*


---

### Description

vcovCR returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from an `lm` object.

### Usage

```

## S3 method for class 'lm'
vcovCR(
  obj,
  cluster,
  type,
  target = NULL,
  inverse_var = NULL,
  form = "sandwich",
  ...
)

```

### Arguments

<code>obj</code>	Fitted model for which to calculate the variance-covariance matrix
<code>cluster</code>	Expression or vector indicating which observations belong to the same cluster. Required for <code>lm</code> objects.
<code>type</code>	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <code>vcovCR</code> for further information.
<code>target</code>	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If a vector, the target matrix is assumed to be diagonal. If not specified, the target is taken to be an identity matrix.
<code>inverse_var</code>	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, <code>vcovCR</code> will attempt to infer a value.

`form` Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting `form = "meat"` will return only the meat of the sandwich and setting `form = B`, where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. `form = "estfun"` will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.

... Additional arguments available for some classes of objects.

### Value

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

### See Also

[vcovCR](#)

### Examples

```
data("ChickWeight", package = "datasets")
lm_fit <- lm(weight ~ Time + Diet:Time, data = ChickWeight)
vcovCR(lm_fit, cluster = ChickWeight$Chick, type = "CR2")

if (requireNamespace("plm", quietly = TRUE)) withAutoprint({

  data("Produc", package = "plm")
  lm_individual <- lm(log(gsp) ~ 0 + state + log(pcap) + log(pc) + log(emp) + unemp, data = Produc)
  individual_index <- !grepl("state", names(coef(lm_individual)))
  vcovCR(lm_individual, cluster = Produc$state, type = "CR2")[individual_index, individual_index]

  # compare to plm()
  plm_FE <- plm::plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
                    data = Produc, index = c("state", "year"),
                    effect = "individual", model = "within")
  vcovCR(plm_FE, type="CR2")

})
```

---

vcovCR.lme

*Cluster-robust variance-covariance matrix for an lme object.*

---

### Description

`vcovCR` returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from a [lme](#) object.

**Usage**

```
## S3 method for class 'lme'
vcovCR(obj, cluster, type, target, inverse_var, form = "sandwich", ...)
```

**Arguments**

obj	Fitted model for which to calculate the variance-covariance matrix
cluster	Optional expression or vector indicating which observations belong to the same cluster. If not specified, will be set to <code>getGroups(obj)</code> .
type	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
target	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If not specified, the target is taken to be the estimated variance-covariance structure of the <code>lme</code> object.
inverse_var	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, <code>vcovCR</code> will attempt to infer a value.
form	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting <code>form = "meat"</code> will return only the meat of the sandwich and setting <code>form = B</code> , where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. <code>form = "estfun"</code> will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.
...	Additional arguments available for some classes of objects.

**Value**

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

**See Also**

[vcovCR](#)

**Examples**

```
if (requireNamespace("nlme", quietly = TRUE)) {
  library(nlme)
  rat_weight <- lme(weight ~ Time * Diet, data=BodyWeight, ~ Time | Rat)
  vcovCR(rat_weight, type = "CR2")
}

pkgs_available <-
  requireNamespace("nlme", quietly = TRUE) &
  requireNamespace("mlmRev", quietly = TRUE)
```

```

if (pkgs_available) {

  data(egsingle, package = "mlmRev")
  subset_ids <- levels(egsingle$schoolid)[1:10]
  egsingle_subset <- subset(egsingle, schoolid %in% subset_ids)

  math_model <- lme(math ~ year * size + female + black + hispanic,
                    random = list(~ year | schoolid, ~ 1 | childid),
                    data = egsingle_subset)

  vcovCR(math_model, type = "CR2")

}

```

---

vcovCR.lmerMod

*Cluster-robust variance-covariance matrix for an lmerMod object.*


---

### Description

vcovCR returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from [merMod](#) object.

### Usage

```

## S3 method for class 'lmerMod'
vcovCR(obj, cluster, type, target, inverse_var, form = "sandwich", ...)

```

### Arguments

obj	Fitted model for which to calculate the variance-covariance matrix
cluster	Optional expression or vector indicating which observations belong to the same cluster. If not specified, will be set to <code>getGroups(obj)</code> .
type	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
target	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If not specified, the target is taken to be the estimated variance-covariance structure of the <code>lmerMod</code> object.
inverse_var	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, <code>vcovCR</code> will attempt to infer a value.
form	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting <code>form = "meat"</code> will return only the meat of the sandwich and setting <code>form = B</code> , where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix

calculated using B as the bread. `form = "estfun"` will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.

... Additional arguments available for some classes of objects.

### Value

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

### See Also

[vcovCR](#)

### Examples

```
if (requireNamespace("lme4", quietly = TRUE)) {
  library(lme4)
  sleep_fit <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
  vcovCR(sleep_fit, type = "CR2")
}

pkgs_available <-
  requireNamespace("lme4", quietly = TRUE) &
  requireNamespace("mlmRev", quietly = TRUE)

if (pkgs_available) {
  data(egsingle, package = "mlmRev")
  subset_ids <- levels(egsingle$schoolid)[1:10]
  math_model <- lmer(math ~ year * size + female + black + hispanic
    + (1 | schoolid) + (1 | childid),
    data = egsingle, subset = schoolid %in% subset_ids)
  vcovCR(math_model, type = "CR2")
}
```

---

vcovCR.mlm

*Cluster-robust variance-covariance matrix for an mlm object.*

---

### Description

`vcovCR` returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from an `mlm` object.



**Usage**

```
## S3 method for class 'mlm'
vcovCR(obj, cluster, type, target, inverse_var, form = "sandwich", ...)
```

**Arguments**

obj	Fitted model for which to calculate the variance-covariance matrix
cluster	Optional expression or vector indicating which observations belong to the same cluster. If not specified, each row of the data will be treated as a separate cluster.
type	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
target	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If not specified, the target is taken to be an identity matrix.
inverse_var	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, vcovCR will attempt to infer a value.
form	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting form = "meat" will return only the meat of the sandwich and setting form = B, where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. form = "estfun" will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.
...	Additional arguments available for some classes of objects.

**Value**

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

**See Also**

[vcovCR](#)

**Examples**

```
iris_fit <- lm(cbind(Sepal.Length, Sepal.Width) ~ Species +
              Petal.Length + Petal.Width, data = iris)
Vcluster <- vcovCR(iris_fit, type = "CR2")
Vcluster
```

vcovCR.plm

*Cluster-robust variance-covariance matrix for a plm object.***Description**

vcovCR returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from a `plm` object.

**Usage**

```
## S3 method for class 'plm'
vcovCR(
  obj,
  cluster,
  type,
  target,
  inverse_var,
  form = "sandwich",
  ignore_FE = FALSE,
  ...
)
```

**Arguments**

<code>obj</code>	Fitted model for which to calculate the variance-covariance matrix
<code>cluster</code>	Optional character string, expression, or vector indicating which observations belong to the same cluster. For fixed-effect models that include individual effects or time effects (but not both), the cluster will be taken equal to the included fixed effects if not otherwise specified. Clustering on individuals can also be obtained by specifying the name of the individual index (e.g., <code>cluster = "state"</code> ) or <code>cluster = "individual"</code> ; clustering on time periods can be obtained by specifying the name of the time index (e.g., <code>cluster = "year"</code> ) or <code>cluster = "time"</code> ; if a group index is specified, clustering on groups (in which individuals are nested) can be obtained by specifying the name of the group index or <code>cluster = "group"</code> . For random-effects models, the cluster will be taken equal to the included random effect identifier if not otherwise specified.
<code>type</code>	Character string specifying which small-sample adjustment should be used, with available options <code>"CR0"</code> , <code>"CR1"</code> , <code>"CR1p"</code> , <code>"CR1S"</code> , <code>"CR2"</code> , or <code>"CR3"</code> . See "Details" section of <code>vcovCR</code> for further information.
<code>target</code>	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. By default, the target is taken to be an identity matrix for fixed effect models or the estimated compound-symmetric covariance matrix for random effects models.
<code>inverse_var</code>	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, <code>vcovCR</code> will attempt to infer a value.

form	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting form = "meat" will return only the meat of the sandwich and setting form = B, where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. form = "estfun" will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.
ignore_FE	Optional logical controlling whether fixed effects are ignored when calculating small-sample adjustments in models where fixed effects are estimated through absorption.
...	Additional arguments available for some classes of objects.

**Value**

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

**See Also**

[vcovCR](#)

**Examples**

```
if (requireNamespace("plm", quietly = TRUE)) withAutoprint({

  library(plm)
  # fixed effects
  data("Produc", package = "plm")
  plm_FE <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
               data = Produc, index = c("state", "year", "region"),
               effect = "individual", model = "within")
  vcovCR(plm_FE, type="CR2")
  vcovCR(plm_FE, type = "CR2", cluster = Produc$region) # clustering on region

  # random effects
  plm_RE <- update(plm_FE, model = "random")
  vcovCR(plm_RE, type = "CR2")
  vcovCR(plm_RE, type = "CR2", cluster = Produc$region) # clustering on region

  # nested random effects
  plm_nested <- update(plm_FE, effect = "nested", model = "random")
  vcovCR(plm_nested, type = "CR2") # clustering on region
})

pkgs_available <- requireNamespace("plm", quietly = TRUE) & requireNamespace("AER", quietly = TRUE)

if (pkgs_available) withAutoprint({
  # first differencing
  data(Fatalities, package = "AER")
  Fatalities <- within(Fatalities, {
    frate <- 10000 * fatal / pop
  })
})
```

```

    drinkagec <- cut(drinkage, breaks = 18:22, include.lowest = TRUE, right = FALSE)
    drinkagec <- relevel(drinkagec, ref = 4)
  })

  plm_FD <- plm(frate ~ beertax + drinkagec + miles + unemp + log(income),
               data = Fatalities, index = c("state", "year"),
               model = "fd")
  vcovHC(plm_FD, method="arellano", type = "sss", cluster = "group")
  vcovCR(plm_FD, type = "CR1S")
  vcovCR(plm_FD, type = "CR2")

})

```

---

vcovCR.rma.mv

*Cluster-robust variance-covariance matrix for a rma.mv object.*


---

## Description

vcovCR returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from a `rma.mv` object.

## Usage

```

## S3 method for class 'rma.mv'
vcovCR(obj, cluster, type, target, inverse_var, form = "sandwich", ...)

```

## Arguments

<code>obj</code>	Fitted model for which to calculate the variance-covariance matrix
<code>cluster</code>	Optional expression or vector indicating which observations belong to the same cluster. If not specified, will be set to the factor in the random-effects structure with the fewest distinct levels. Caveat emptor: the function does not check that the random effects are nested.
<code>type</code>	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <code>vcovCR</code> for further information.
<code>target</code>	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If not specified, the target is taken to be the estimated variance-covariance structure of the <code>rma.mv</code> object.
<code>inverse_var</code>	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, <code>vcovCR</code> will attempt to infer a value.
<code>form</code>	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting <code>form = "meat"</code> will return only the meat of the sandwich and setting <code>form = B</code> , where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix

calculated using  $B$  as the bread. `form = "estfun"` will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.

... Additional arguments available for some classes of objects.

### Value

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

### See Also

[vcovCR](#)

### Examples

```
pkgs_available <-
  requireNamespace("metafor", quietly = TRUE) &
  requireNamespace("metadat", quietly = TRUE)

if (pkgs_available) withAutoprint({

  library(metafor)
  data(dat.assink2016, package = "metadat")

  mfor_fit <- rma.mv(yi ~ year + deltype,
                   V = vi, random = ~ 1 | study / esid,
                   data = dat.assink2016)

  mfor_fit

  mfor_CR2 <- vcovCR(mfor_fit, type = "CR2")
  mfor_CR2

  coef_test(mfor_fit, vcov = mfor_CR2, test = c("Satterthwaite", "saddlepoint"))
  Wald_test(mfor_fit, constraints = constrain_zero(3:4), vcov = mfor_CR2)

})
```

---

vcovCR.rma.uni

*Cluster-robust variance-covariance matrix for a rma.uni object.*

---

### Description

`vcovCR` returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from a [rma.uni](#) object.

**Usage**

```
## S3 method for class 'rma.uni'
vcovCR(obj, cluster, type, target, inverse_var, form = "sandwich", ...)
```

**Arguments**

obj	Fitted model for which to calculate the variance-covariance matrix
cluster	Expression or vector indicating which observations belong to the same cluster. Required for <code>rma.uni</code> objects.
type	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
target	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If not specified, the target is taken to be diagonal with entries equal to the estimated marginal variance of the effect sizes.
inverse_var	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, <code>vcovCR</code> will attempt to infer a value.
form	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting <code>form = "meat"</code> will return only the meat of the sandwich and setting <code>form = B</code> , where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. <code>form = "estfun"</code> will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.
...	Additional arguments available for some classes of objects.

**Value**

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

**See Also**

[vcovCR](#)

**Examples**

```
pkgs_available <-
  requireNamespace("metafor", quietly = TRUE) &
  requireNamespace("metadat", quietly = TRUE)

if (pkgs_available) withAutoprint({

  library(metafor)
  data(dat.assink2016, package = "metadat")

  mfor_fit <- rma.uni(yi ~ year + deltype, vi = vi,
```

```

                                data = dat.assink2016)
mfor_fit

mfor_CR2 <- vcovCR(mfor_fit, type = "CR2", cluster = dat.assink2016$study)
mfor_CR2
coef_test(mfor_fit, vcov = mfor_CR2, test = c("Satterthwaite", "saddlepoint"))
Wald_test(mfor_fit, constraints = constrain_zero(2:4), vcov = mfor_CR2)

})

```

vcovCR.robust

*Cluster-robust variance-covariance matrix for a robust object.***Description**

vcovCR returns a sandwich estimate of the variance-covariance matrix of a set of regression coefficient estimates from a [robust](#) object.

**Usage**

```

## S3 method for class 'robust'
vcovCR(obj, cluster, type, target, inverse_var, form = "sandwich", ...)

```

**Arguments**

obj	Fitted model for which to calculate the variance-covariance matrix
cluster	Optional expression or vector indicating which observations belong to the same cluster. If not specified, will be set to the studynum used in fitting the <a href="#">robust</a> object.
type	Character string specifying which small-sample adjustment should be used, with available options "CR0", "CR1", "CR1p", "CR1S", "CR2", or "CR3". See "Details" section of <a href="#">vcovCR</a> for further information.
target	Optional matrix or vector describing the working variance-covariance model used to calculate the CR2 and CR4 adjustment matrices. If not specified, the target is taken to be the inverse of the estimated weights used in fitting the <a href="#">robust</a> object.
inverse_var	Optional logical indicating whether the weights used in fitting the model are inverse-variance. If not specified, vcovCR will attempt to infer a value.
form	Controls the form of the returned matrix. The default "sandwich" will return the sandwich variance-covariance matrix. Alternately, setting form = "meat" will return only the meat of the sandwich and setting form = B, where B is a matrix of appropriate dimension, will return the sandwich variance-covariance matrix calculated using B as the bread. form = "estfun" will return the (appropriately scaled) estimating function, the transposed crossproduct of which is equal to the sandwich variance-covariance matrix.
...	Additional arguments available for some classes of objects.

**Value**

An object of class `c("vcovCR", "clubSandwich")`, which consists of a matrix of the estimated variance of and covariances between the regression coefficient estimates.

**See Also**

[vcovCR](#)

**Examples**

```
if (requireNamespace("robumeta", quietly = TRUE)) withAutoprint({
  library(robumeta)
  data(hierdat)

  robu_fit <- robu(effectsize ~ binge + followup + sreport + age,
                 data = hierdat, studynum = studyid,
                 var.eff.size = var, modelweights = "HIER")
  robu_fit

  robu_CR2 <- vcovCR(robu_fit, type = "CR2")
  robu_CR2
  coef_test(robu_fit, vcov = robu_CR2, test = c("Satterthwaite", "saddlepoint"))

  Wald_test(robu_fit, constraints = constrain_zero(c(2,4)), vcov = robu_CR2)
  Wald_test(robu_fit, constraints = constrain_zero(2:5), vcov = robu_CR2)

})
```

---

Wald\_test

*Test parameter constraints in a fitted linear regression model*

---

**Description**

Wald\_test reports Wald-type tests of linear contrasts from a fitted linear regression model, using a sandwich estimator for the variance-covariance matrix and a small sample correction for the p-value. Several different small-sample corrections are available.

**Usage**

```
Wald_test(
  obj,
  constraints,
  vcov,
  null_constant = 0,
  test = "HTZ",
  tidy = FALSE,
  adjustment_method = "none",
  ...
)
```



**Arguments**

<code>obj</code>	Fitted model for which to calculate Wald tests.
<code>constraints</code>	constraint or list of multiple constraints to test. See details and examples.
<code>vcov</code>	Variance covariance matrix estimated using <code>vcovCR</code> or a character string specifying which small-sample adjustment should be used to calculate the variance-covariance.
<code>null_constant</code>	vector of null values or list of such vectors for each set of constraints to test. For a single constraint, the null values must have length equal to the number of rows in the constraint. For lists of null values, each entry must have length equal to the number of rows in the corresponding entry of <code>constraints</code> . Default is <code>0</code> , in which case the null values are taken to be zero (for every entry, if <code>constraints</code> is a list).
<code>test</code>	Character vector specifying which small-sample correction(s) to calculate. The following corrections are available: "chi-sq", "Naive-F", "Naive-Fp", "HTA", "HTB", "HTZ", "EDF", "EDT". Default is "HTZ".
<code>tidy</code>	Logical value controlling whether to tidy the test results. If <code>constraints</code> is a list with multiple constraints, the result will be coerced into a data frame when <code>tidy = TRUE</code> .
<code>adjustment_method</code>	A character string indicating a multiple comparisons correction to apply to p-values in instances where multiple tests are run. Possible options are from <a href="#">p.adjust.methods</a> , which is passed to <code>p.adjust</code> to correct p-values for multiple comparisons. Defaults to "none".
<code>...</code>	Further arguments passed to <code>vcovCR</code> , which are only needed if <code>vcov</code> is a character string.

**Details**

Constraints can be specified directly as  $q \times p$  matrices or indirectly through [constrain\\_equal](#), [constrain\\_zero](#), or [constrain\\_pairwise](#). By default, each constraint will be tested against the null hypothesis that it equal to a zero vector. Non-zero values for null-hypotheses can be specified using the `null_constant` argument.

**Value**

A list of test results.

**See Also**

[vcovCR](#), [constrain\\_equal](#), [constrain\\_zero](#), [constrain\\_pairwise](#)

**Examples**

```
if (requireNamespace("carData", quietly = TRUE)) withAutoprint({
  data(Duncan, package = "carData")
```

```
Duncan$cluster <- sample(LETTERS[1:8], size = nrow(Duncan), replace = TRUE)

Duncan_fit <- lm(prestige ~ 0 + type + income + type:income + type:education, data=Duncan)
# Note that type:income terms are interactions because main effect of income is included
# but type:education terms are separate slopes for each unique level of type

# Test equality of intercepts
Wald_test(Duncan_fit,
          constraints = constrain_equal(1:3),
          vcov = "CR2", cluster = Duncan$cluster)

# Test equality of type-by-education slopes
Wald_test(Duncan_fit,
          constraints = constrain_equal(":education", reg_ex = TRUE),
          vcov = "CR2", cluster = Duncan$cluster)

# Pairwise comparisons of type-by-education slopes
Wald_test(Duncan_fit,
          constraints = constrain_pairwise(":education", reg_ex = TRUE),
          vcov = "CR2", cluster = Duncan$cluster)

# Test type-by-income interactions
Wald_test(Duncan_fit,
          constraints = constrain_zero(":income", reg_ex = TRUE),
          vcov = "CR2", cluster = Duncan$cluster)

# Pairwise comparisons of type-by-income interactions
Wald_test(Duncan_fit,
          constraints = constrain_pairwise(":income", reg_ex = TRUE, with_zero = TRUE),
          vcov = "CR2", cluster = Duncan$cluster)

# Pairwise comparisons of type-by-education slopes, with two tests and
# multiple comparisons p-value adjustment
Wald_test(Duncan_fit,
          constraints = constrain_pairwise(":education", reg_ex = TRUE),
          vcov = "CR2",
          cluster = Duncan$cluster,
          test = c("HTZ", "chi-sq"),
          adjustment_method = "holm")

})
```

# Index

## \* datasets

AchievementAwardsRCT, 3  
dropoutPrevention, 8  
MortalityRates, 15  
SATcoaching, 18

AchievementAwardsRCT, 3

character, 14  
coef\_test, 4  
conf\_int, 5  
constrain\_equal, 14, 41  
constrain\_equal (constraint\_matrices), 7  
constrain\_pairwise, 14, 41  
constrain\_pairwise  
    (constraint\_matrices), 7  
constrain\_zero, 14, 41  
constrain\_zero (constraint\_matrices), 7  
constraint\_matrices, 7

dropoutPrevention, 8

findCluster.rma.mv, 10

geeglm, 22  
glm, 19, 23  
gls, 19, 25

impute\_covariance\_matrix, 10  
ivreg, 27

linear\_contrast, 13  
lm, 19, 28  
lme, 19, 29

merMod, 31  
MortalityRates, 15

p.adjust, 14, 41  
p.adjust.methods, 41  
pattern\_covariance\_matrix, 16

plm, 19, 34

rma.mv, 19, 36  
rma.uni, 19, 37  
robu, 19, 39

SATcoaching, 18

vcalc, 11, 16  
vcovCR, 5, 6, 14, 19, 20, 22–41  
vcovCR.geeglm, 22  
vcovCR.glm, 21, 23  
vcovCR.gls, 21, 25  
vcovCR.ivreg, 26  
vcovCR.lm, 21, 28  
vcovCR.lme, 21, 29  
vcovCR.lmerMod, 21, 31  
vcovCR.mlm, 32  
vcovCR.plm, 21, 34  
vcovCR.rma.mv, 21, 36  
vcovCR.rma.uni, 21, 37  
vcovCR.robu, 21, 39  
vcovHC, 20

Wald\_test, 7, 40