Package: pcdreg (via r-universe)

November 10, 2026

Type Package

Title Semiparametric Regression for Panel Count Data

Version 0.1.0

Description Fits semiparametric regression models to panel count data, that is, to recurrent event processes observed only at intermittent examination times, allowing covariates to be time-varying. The proportional rate model is fitted by nonparametric maximum likelihood through an expectation-maximization algorithm under a working Poisson assumption, while inference does not require that assumption to hold; two computationally inexpensive covariance estimators are provided, a robust sandwich estimator that remains consistent when the Poisson assumption fails and an efficient information estimator that is a fast substitute for profile likelihood when it holds, with the profile likelihood estimator available for comparison. The proportional means model is fitted by the estimating equation of Hu, Sun and Wei (2003) <doi:10.1111/1467-9469.00316>. Methods for the rate model are described in Sun, Guo, Li, Tu and Sun (2024) <doi:10.3150/23-BEJ1713>.

License GPL (>= 2)

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

Imports Rcpp, cli, generics, ggplot2, rlang, stats, tibble

LinkingTo Rcpp, RcppArmadillo

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

URL https://github.com/dayusun/pcdreg, https://www.sundayu.me/pcdreg/

BugReports https://github.com/dayusun/pcdreg/issues

Config/testthat/edition 3

NeedsCompilation yes

Author Dayu Sun [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7149-3303>), Yuanyuan Guo [ctb]
(co-author of the method paper), Yang Li [ctb] (co-author of
the method paper), Wanzhu Tu [ctb] (co-author of the method
paper), Jianguo Sun [ctb] (co-author of the method paper)

Maintainer Dayu Sun <dayusun@iu.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-10 18:01:17 UTC

RemoteUrl https://github.com/cran/pcdreg

RemoteRef HEAD

RemoteSha ac3c90d9573a08de28b75debf0510a40ddc2e8b0

Contents

autoplot.pcd 2
autoplot.pcdfit 4
baseline 5
is.pcd 5
pcd 6
pcdreg 8
pcdreg-tidiers 10
pcdreg_control 12
predict.pcdfit 13
sim_pcd 14
summary.pcdfit 15
vcov.pcdfit 16

Index **18**

`autoplot.pcd` *Plot panel count data*

Description

Draws the observation pattern as tiles: one row per subject, one tile per examination interval spanning the time it covers, shaded by the number of events recorded in it.

Usage

```
## S3 method for class 'pcd'
autoplot(
  object,
  order_by = c("followup", "events", "none"),
  max_subjects = 60L,
  palette = NULL,
  ...
)

## S3 method for class 'pcd'
plot(x, ...)
```

Arguments

`object`, `x`	A `pcd()` object.
`order_by`	How to order subjects up the vertical axis. `"followup"` sorts by the time of the last examination, which makes the censoring pattern legible; `"events"` sorts by the total number of events; `"none"` keeps the order of the identifiers.
`max_subjects`	Draw at most this many subjects, taken as an evenly spaced sample so the picture stays readable for large studies. A message reports how many were shown.
`palette`	Fill colours, lightest first. The default is a single-hue sequential ramp, which is where a magnitude belongs.
`...`	Ignored.

Details

The tile is the unit of observation. A panel count is attributed to the interval $(T_{i,j-1}, T_{ij}]$ rather than to an instant, so drawing the intervals rather than points at the examination times is what the data actually says: a wide pale tile and a narrow dark one can carry the same count while meaning very different things about the rate.

Two features decide how an analysis will go and both are visible here. Whether examination times line up across subjects sets the size of the pooled grid and so the cost of fitting the rate model; and how ragged the right hand edge is shows how much of the tail of the baseline rests on a handful of subjects.

Value

A `ggplot2::ggplot()` object, so it can be titled, faceted or themed like any other. `autoplot()` returns it; `plot()` draws it and returns it invisibly, so it can be used for its side effect like any other `plot()` method.

See Also

`autoplot.pcdfit()` for the fitted baseline function.

Examples

```
set.seed(1)
d <- sim_pcd(40)
autoplot(with(d, pcd(id, tstart, tstop, count)))
```

`autoplot.pcdfit` *Plot the estimated baseline function*

Description

Plot the estimated baseline function

Usage

```
## S3 method for class 'pcdfit'
autoplot(object, ...)

## S3 method for class 'pcdfit'
plot(x, ...)
```

Arguments

`object`, `x`	A fitted `pcdreg()` model.
`...`	Ignored.

Details

The rate model's $\hat{\Lambda}$ increases by construction. The means model's $\hat{\mu}$ need not, and with covariates that fluctuate it generally does not; it is drawn as estimated rather than forced upwards, because that behaviour is the point of the comparison between the two models.

Value

A `ggplot2::ggplot()` object: a step function of the cumulative baseline rate for the rate model, or of the baseline mean for the means model. `autoplot()` returns it; `plot()` draws it and returns it invisibly, so it can be used for its side effect like any other `plot()` method.

Examples

```
set.seed(1)
d <- sim_pcd(80, beta = c(1, -1), lambda = function(t) 8 / (1 + t))
autoplot(pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, d))
```

baseline *Estimated baseline function*

Description

Estimated baseline function

Usage

```
baseline(object, ...)

## S3 method for class 'pcdfit'
baseline(object, ...)
```

Arguments

object	A fitted pcdreg() model.
...	Ignored.

Value

A tibble with one row per examination time on the pooled grid. For pcdreg() it gives the jump sizes and the cumulative baseline rate $\Lambda(t)$; for pcdreg() it gives the baseline mean $\mu(t)$, which is not constrained to increase.

Examples

```
set.seed(1)
d <- sim_pcd(80, beta = c(1, -1), lambda = function(t) 8 / (1 + t))
head(baseline(pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, d)))
head(baseline(pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, d, model = "mean")))
```

is.pcd *Test for a panel count response*

Description

Test for a panel count response

Usage

```
is.pcd(x)
```

Arguments

x	An object.

Value

TRUE if x was created by pcd().

Examples

```
is.pcd(pcd(c(1, 2), c(1, 1), c(0, 2)))
```

---

pcd *Describe panel count observations*

Description

Creates the response object used on the left hand side of a pcdreg() formula, in the same spirit as survival::Surv().

Usage

```
pcd(id, time, time2, count)
```

Arguments

id	Subject identifier. May be numeric, character or a factor; rows sharing an identifier are treated as one subject.
time	Examination time in the three argument form, or the start of the interval in the four argument form.
time2	The end of the interval in the four argument form; the event count in the three argument form.
count	Number of events observed since the previous examination. Use NA on rows that only record a change in a covariate rather than an examination. Omitted in the three argument form.

Details

There are two forms, distinguished by the number of arguments. Which one you need depends on a single question: does any covariate change value during follow-up?

pcd(id, time, count) is for data with one row per examination, which is all that time-invariant covariates need. Interval starts are filled in as the previous examination time of the same subject, with the first interval starting at zero.

pcd(id, tstart, tstop, count) is the counting process form, needed as soon as a covariate moves. Each row gives an interval (tstart, tstop] over which the covariates in that row are constant, exactly as for a time-varying survival::coxph() fit. The count column distinguishes the two kinds of row:

- a number means the row ends at an **examination**, and that many events occurred since the previous examination;
- NA means the row ends at a **covariate change** rather than an examination.

There is no separate event indicator; the presence or absence of a count is the indicator.

Value

An object of class "pcd": a numeric matrix with columns id, tstart, tstop, count and exam, carrying the original identifier labels in the "labels" attribute. Missing counts are stored as count = 0 with exam = 0, so that the response contains no NA and na.action handling applies to the covariates alone.

Counts are increments, not totals

count must be the number of events since the previous examination, ΔN_{ij}, not the running total $N_i(T_{ij})$. Data recorded cumulatively must be differenced within subject first. This is not something the package can detect for you, and getting it wrong inflates the event count and biases the coefficients rather than raising an error. A quick check after fitting is that fit$nevent matches the number of events you believe you have. See vignette("data-preparation") for the conversion.

What the data must satisfy

Checked before any arithmetic, naming the offending subject when a check fails:

- follow-up starts at time zero, since the models integrate from there;
- intervals within a subject are contiguous, with no gaps or overlaps;
- no subject has two rows ending at the same time;
- every subject has at least one examination;
- counts are non-negative whole numbers.

Row order is irrelevant: the data are sorted internally. Identifiers may be numeric, character or factor, and need not be consecutive.

A subject's follow-up ends at their last examination, because the likelihood runs only over $j = 1, \ldots, J_i$. Rows lying entirely beyond it carry no information and are dropped, with a message saying how many. A censoring row appended after the final visit is the usual reason for seeing it.

See Also

pcdreg() and vignette("data-preparation") for converting data into this shape.

Examples

```
# One row per examination: subject 1 seen at 0.5 and 1.2 with 2 then 0
# events, subject 2 seen once at 0.8 with 3.
pcd(c(1, 1, 2), c(0.5, 1.2, 0.8), c(2, 0, 3))

# Counting process form. Subject 1's covariates change at t = 0.9, part way
# through the examination interval (0.5, 1.2], so that interval needs two
# rows: the first ends at the change and carries NA, the second ends at the
# examination and carries its count.
pcd(
  id     = c(1, 1, 1, 2),
  time   = c(0.0, 0.5, 0.9, 0.0),
  time2  = c(0.5, 0.9, 1.2, 0.8),
  count  = c(2, NA, 0, 3)
```

```
)

# Used on the left hand side of a formula, which is how you will normally
# meet it.
set.seed(1)
d <- sim_pcd(40)
fit <- pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, data = d)
c(subjects = fit$n, examinations = fit$nexam, events = fit$nevent)
```

`pcdreg` *Semiparametric regression for panel count data*

Description

Fits either of the two standard semiparametric models to panel count data, that is, to recurrent events observed only at intermittent examination times. Covariates may vary over time.

Usage

```
pcdreg(
  formula,
  data,
  model = c("rate", "mean"),
  subset,
  na.action,
  control = pcdreg_control(),
  profile = FALSE,
  init = NULL
)
```

Arguments

`formula`	A formula whose left hand side is a `pcd()` object, for example `pcd(id, tstart, tstop, count) ~ x1 + x2`.
`data`	A data frame containing the variables in `formula`.
`model`	Which model to fit. `"rate"` is the proportional rate model and the default; `"mean"` is the proportional means model.
`subset, na.action`	Handled as in `stats::lm()`. Note that missing covariate values delete rows, which can break the contiguity of a subject's intervals; resolve them before fitting.
`control`	A list from `pcdreg_control()`.
`profile`	Rate model only: whether to also compute the profile likelihood covariance. It is provided for comparison with the literature and costs one extra baseline-only fit per coefficient, so it is off by default.
`init`	Optional starting values for the coefficients.

Value

An object of class `"pcdfit"`, with components including `coefficients`, `model` (which model was fitted), `baseline` (a data frame of the estimated baseline function), `Omega` and `S` (the two matrices behind the sandwich), `vcov` (a list of the available covariance estimates), and convergence information. Rate model fits also carry `loglik`.

The two models

$$\text{rate:} \quad E[dN(t) \mid X(t)] = \exp(\beta' X(t))\, d\Lambda(t)$$

$$\text{mean:} \quad E[N(t) \mid X(t)] = \mu(t) \exp(\beta' X(t))$$

The **rate model** is fitted by nonparametric maximum likelihood, treating Λ as a step function with a jump at each observed examination time. The likelihood maximised is the one a nonhomogeneous Poisson process would give, but that is a working device rather than an assumption about the data: the estimator stays consistent and asymptotically normal when it fails, and the default covariance estimator stays valid.

The **means model** is fitted by the estimating equation of Hu, Sun and Wei (2003). Only the examination times enter, so no covariate values between them are needed and it is much cheaper. There is no likelihood behind it, so it reports no log likelihood, `profile = TRUE` is unavailable, and the only covariance is the sandwich.

The two are not reparametrisations of each other when covariates vary over time, so their coefficients answer different questions rather than estimating a common quantity: β acts on the instantaneous rate in one and on the cumulative mean in the other. A further practical difference is that nothing constrains the fitted $\hat{\mu}$ to increase, and with fluctuating covariates it generally does not, which is the drawback of the means model that motivates the rate model.

Specifying the model

The left hand side is always a `pcd()` object, which carries the subject, the interval and the event count. The right hand side is an ordinary model formula, so interactions, transformations, factors and `.` all behave as usual:

```
pcd(id, tstart, tstop, count) ~ x1 + x2
pcd(id, tstart, tstop, count) ~ x1 * x2 + log(x3)
pcd(id, time, count)          ~ .
```

The dot expands to every column that is not part of the response, so the identifier, times and counts are excluded automatically.

No intercept is fitted. A constant column cannot be distinguished from a rescaling of the baseline, so it is dropped and the baseline is reported separately by `baseline()`. Adding `- 1` changes nothing. One consequence is that shifting a covariate by a constant c leaves β alone and rescales the baseline by $e^{-c\beta}$: centring covariates moves the baseline, not the coefficients.

`vignette("data-preparation")` covers the data layouts in full, including the conversion from cumulative counts, which is the mistake most worth avoiding.

Covariance estimation

Available through `vcov.pcdfit()`. `"robust"` is the sandwich $\Omega^{-1} S \Omega^{-1}/n$, which does not rely on the Poisson assumption, is the default, and is the only option for the means model.

For the rate model two more are available. `"information"` is S^{-1}/n, valid only under the Poisson assumption but a fast substitute for the profile likelihood when it holds. `"profile"` is the numerical profile likelihood estimator of Murphy and van der Vaart (2000), computed only when `profile = TRUE`; it is there for comparison with the literature rather than for routine use, since it rests on a numerical derivative with a step of order $n^{-1/2}$ that is coarse at small sample sizes.

When the counts are overdispersed relative to Poisson, which is the usual situation, the latter two understate the standard errors, sometimes severely.

References

Sun, D., Guo, Y., Li, Y., Tu, W. and Sun, J. (2024). A robust approach for regression analysis of panel count data. *Bernoulli* **30**(4), 3251–3275. doi:10.3150/23BEJ1713

Hu, X. J., Sun, J. and Wei, L. J. (2003). Regression parameter estimation from panel counts. *Scandinavian Journal of Statistics* **30**(1), 25–43. doi:10.1111/14679469.00316

Murphy, S. A. and van der Vaart, A. W. (2000). On profile likelihood. *Journal of the American Statistical Association* **95**, 449–465.

See Also

`pcd()`, `vcov.pcdfit()`, `baseline()`, `sim_pcd()`

Examples

```
set.seed(1)
d <- sim_pcd(80, beta = c(1, -1), lambda = function(t) 8 / (1 + t))

fit <- pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, data = d)
summary(fit)

# The same data under the means model. The coefficients are not comparable
# term by term, because the two models act on different quantities.
mfit <- pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, data = d,
               model = "mean")
cbind(rate = coef(fit), mean = coef(mfit))
```

---

pcdreg-tidiers *Tidy a panel count model fit*

Description

Methods for the `generics::tidy()`, `generics::glance()` and `generics::augment()` generics, so that a `pcdreg()` fit can be handled the same way as any other model in a tidy workflow.

Usage

```
## S3 method for class 'pcdfit'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  exponentiate = FALSE,
  type = c("robust", "information", "profile"),
  ...
)

## S3 method for class 'pcdfit'
glance(x, ...)

## S3 method for class 'pcdfit'
augment(x, data = NULL, newdata = NULL, ...)
```

Arguments

x	A fitted pcdreg() model.
conf.int	Whether to add conf.low and conf.high columns.
conf.level	Width of the interval, if one is requested.
exponentiate	Whether to report e^{β} rather than β. For the rate model this is the multiplicative effect of a one unit change in the covariate on the rate, read the same way as a hazard ratio; for the means model it is the effect on the mean. Standard errors are left on the log scale, as elsewhere in the tidy ecosystem.
type	Which covariance estimator to base the standard errors on. See vcov.pcdfit(); the default is the robust sandwich.
...	Ignored.
data, newdata	Data to augment. Defaults to the data the model was fitted to. It must contain the pcd() response, since the fitted mean follows each subject's covariate trajectory.

Value

tidy() gives one row per coefficient, with columns term, estimate, std.error, statistic and p.value.

glance() gives a one row summary of the fit. logLik is NA for the means model, which is fitted by an estimating equation and so has none.

augment() returns data with four columns added:

.linear.predictor $\beta' X(t)$ on each row.

.fitted the predicted mean number of events by the end of the row's interval.

.observed the observed cumulative count at that time, on examination rows, and NA on rows that only record a covariate change.

.resid .observed - .fitted, the difference between the counts seen so far and the mean the model implies.

Examples

```
set.seed(1)
d <- sim_pcd(60, beta = c(1, -1), lambda = function(t) 8 / (1 + t))
fit <- pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, data = d)

tidy(fit)
tidy(fit, conf.int = TRUE, exponentiate = TRUE)
glance(fit)
augment(fit)
```

`pcdreg_control` *Tuning parameters for the fitting algorithms*

Description

Tuning parameters for the fitting algorithms

Usage

```
pcdreg_control(
  maxit = 2000L,
  reltol = 1e-07,
  accelerate = TRUE,
  profile_maxit = 20000L,
  profile_reltol = 1e-06,
  profile_h = NULL
)
```

Arguments

`maxit`	Maximum number of iterations. For the rate model these are EM iterations; for the means model, Newton iterations of the estimating equation, which converges in a handful.
`reltol`	Convergence tolerance on the largest relative change in the coefficients, and for the rate model in the baseline jump sizes too.
`accelerate`	Rate model only: whether to extrapolate the EM iterations by the SQUAREM scheme of Varadhan and Roland (2008). This reaches the same fixed point in far fewer passes; set it to `FALSE` to run plain EM.
`profile_maxit, profile_reltol`	Limits for the inner runs that maximise over the baseline with the coefficients held fixed, used only by the profile likelihood covariance. These can be looser than `reltol` without loss: the profile log likelihood is stationary in the baseline at its maximum, so an error of size ϵ in the inner solution moves it by only $O(\epsilon^2)$.
`profile_h`	Step size for the numerical derivative of the profile log likelihood. The default, `NULL`, uses $5n^{-1/2}$ as recommended by Zeng et al. (2016) and used in the paper.

Value

A list of control values.

References

Varadhan, R. and Roland, C. (2008). Simple and globally convergent methods for accelerating the convergence of any EM algorithm. *Scandinavian Journal of Statistics* **35**, 335–353.

Examples

```
pcdreg_control(reltol = 1e-9)
```

---

predict.pcdfit *Predictions from a panel count model*

Description

Predictions from a panel count model

Usage

```
## S3 method for class 'pcdfit'
predict(object, newdata, type = c("lp", "mean"), ...)
```

Arguments

`object`	A fitted `pcdreg()` model.
`newdata`	Data to predict for. Defaults to the data the model was fitted to. For `type = "mean"` it must also contain the variables in the `pcd()` response, since the prediction follows each subject's covariate trajectory; the counts themselves are ignored.
`type`	`"lp"` returns the linear predictor $\beta' X(t)$ for each row. `"mean"` returns the predicted mean number of events at every fitted examination time within each subject's follow-up: $\int_0^t \exp(\beta' X(s))\, d\hat{\Lambda}(s)$ for `pcdreg()`, and $\hat{\mu}(t)\exp(\beta' X(t))$ for `pcdreg()`.
`...`	Ignored.

Details

The rate model's predicted mean is non-decreasing in t by construction, because it integrates a positive quantity. The means model's is not: it is a fitted $\hat{\mu}(t)$ rescaled by a covariate value that may itself move up and down, so a predicted mean count that falls is possible and is the practical drawback of that model with time-varying covariates.

Value

For `type = "lp"`, a numeric vector with one element per row of `newdata`. For `type = "mean"`, a tibble with columns `id`, `time` and `mean`.

Examples

```
set.seed(1)
d <- sim_pcd(60, beta = c(1, -1), lambda = function(t) 8 / (1 + t))
fit <- pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, data = d)
head(predict(fit))
head(predict(fit, type = "mean"))
```

`sim_pcd` *Simulate panel count data with a time-varying covariate*

Description

Generates data from the design used in the simulation study of Sun et al. (2024), which is a convenient source of examples and test cases.

Usage

```
sim_pcd(
  n,
  beta = c(1, -1),
  lambda = function(t) 8/(1 + t),
  tau = 2,
  frailty = 0,
  exam_mean = 4,
  lambda_max = NULL,
  digits = 2
)
```

Arguments

`n`	Number of subjects.
`beta`	Length two coefficient vector for `x1` and `x2`.
`lambda`	Baseline rate function, vectorised over `t`.
`tau`	Maximum follow-up time.
`frailty`	Variance of a gamma frailty with mean one that multiplies each subject's rate. Zero, the default, gives a genuine Poisson process; a positive value produces overdispersed counts that violate the Poisson assumption while leaving the rate model intact.
`exam_mean`	Mean of the zero-truncated Poisson distribution from which the number of examinations per subject is drawn.

`lambda_max` Optional upper bound for `lambda` on `[0, tau]`, used by the thinning algorithm. The default takes the maximum over a fine grid, which is reliable unless `lambda` oscillates very rapidly.

`digits` Number of decimal places to round examination times to, or `NA` to leave them unrounded. Rounding follows the design of the paper and of Wellner and Zhang (2007): examinations then tie across subjects, which is realistic for scheduled visits and keeps the pooled grid, and hence the cost of fitting, from growing quadratically in the sample size.

Details

The time-varying covariate is a single step, $x_1(t) = X_{11} I(t \le U_1) + X_{12} I(t > U_1)$, with $X_{11}, X_{12} \sim U(0,1)$ and $U_1 \sim U(0,\tau)$. The time-invariant covariate `x2` is $U(0,1)$. Events follow a nonhomogeneous Poisson process with rate $r\lambda(t)\exp(\beta' X(t))$, where r is one unless `frailty` is positive. Examination times are the order statistics of a uniform sample on $(0, \bar{T})$ with $\bar{T} \sim U(0.9\tau, \tau)$.

Value

A tibble in counting process form with columns `id`, `tstart`, `tstop`, `count`, `x1` and `x2`. Rows are intervals over which the covariates are constant; `count` is the number of events since the previous examination and is `NA` on rows that only record a covariate change.

References

Sun, D., Guo, Y., Li, Y., Tu, W. and Sun, J. (2024). A robust approach for regression analysis of panel count data. *Bernoulli* **30**(4), 3251–3275. doi:10.3150/23BEJ1713

Examples

```
set.seed(42)
d <- sim_pcd(5, beta = c(1, -1), lambda = function(t) 8 / (1 + t))
head(d)

# Overdispersed counts: the rate model still holds, the Poisson one does not.
od <- sim_pcd(5, frailty = 1)
```

---

summary.pcdfit *Summarise a fitted panel count model*

Description

Summarise a fitted panel count model

Usage

```
## S3 method for class 'pcdfit'
summary(object, type = c("robust", "information", "profile"), ...)
```

Arguments

object	A fitted pcdreg() model.
type	Which covariance estimator to base the standard errors on. See vcov.pcdfit().
...	Ignored.

Value

An object of class "summary.pcdfit", whose coefficients component is the usual four column table of estimates, standard errors, Wald statistics and two sided p-values.

Examples

```
set.seed(1)
d <- sim_pcd(80, beta = c(1, -1), lambda = function(t) 8 / (1 + t))
fit <- pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, data = d)
summary(fit)
summary(fit, "information")
```

---

vcov.pcdfit *Covariance matrix of the estimated coefficients*

Description

Covariance matrix of the estimated coefficients

Usage

```
## S3 method for class 'pcdfit'
vcov(object, type = c("robust", "information", "profile"), ...)
```

Arguments

object	A fitted pcdreg() model.
type	Which estimator to return. "robust" is the sandwich estimator that does not rely on the Poisson assumption and is available for both models. "information" and "profile" apply to pcdreg() only: the first is the efficient information estimator, valid under the Poisson assumption, and the second is the profile likelihood estimator, available only if the model was fitted with profile = TRUE.
...	Ignored.

Value

A covariance matrix.

Examples

```
set.seed(1)
d <- sim_pcd(80, beta = c(1, -1), lambda = function(t) 8 / (1 + t))
fit <- pcdreg(pcd(id, tstart, tstop, count) ~ x1 + x2, data = d)
sqrt(diag(vcov(fit)))
sqrt(diag(vcov(fit, "information")))
```

Index

augment.pcdfit *(*pcdreg-tidiers*)*, 10
autoplot.pcd, 2
autoplot.pcdfit, 4
autoplot.pcdfit(), *3*

baseline, 5
baseline(), *9, 10*

generics::augment(), *10*
generics::glance(), *10*
generics::tidy(), *10*
ggplot2::ggplot(), *3, 4*
glance.pcdfit *(*pcdreg-tidiers*)*, 10

is.pcd, 5

pcd, 6
pcd(), *3*, *6*, *8–11*, *13*
pcdreg, 8
pcdreg(), *4–7*, *10, 11*, *13*, *16*
pcdreg-tidiers, 10
pcdreg_control, 12
pcdreg_control(), *8*
plot.pcd *(*autoplot.pcd*)*, 2
plot.pcdfit *(*autoplot.pcdfit*)*, 4
predict.pcdfit, 13

sim_pcd, 14
sim_pcd(), *10*
stats::lm(), *8*
summary.pcdfit, 15
survival::coxph(), *6*
survival::Surv(), *6*

tibble, *5*, *14, 15*
tidy.pcdfit *(*pcdreg-tidiers*)*, 10

vcov.pcdfit, 16
vcov.pcdfit(), *10, 11*, *16*