# Package: PanelSelect (via r-universe)

October 29, 2026

**Type** Package

**Title** Panel Sample Selection Models

**Version** 2.0.0

**Maintainer** Jing Peng <jing.peng@uconn.edu>

**Description** Fits sample selection models with continuous, binary, and count outcomes in panel and cross-sectional data. Model details are provided in Bailey and Peng (2025) <doi:10.2139/ssrn.5475626> and Peng and Van den Bulte (2024) <doi:10.1287/mnsc.2019.01897>.

**License** GPL (>= 3)

**Encoding** UTF-8

**VignetteBuilder** knitr

**Depends** R (>= 4.5.0)

**Imports** Rcpp, pbivnorm, statmod, stats, utils

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** knitr, rmarkdown, testthat (>= 3.1.7), mvtnorm, maxLik

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.1.1

**NeedsCompilation** yes

**Author** Jing Peng [aut, cre]

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

**Date/Publication** 2026-10-29 08:40:16 UTC

**RemoteUrl** https://github.com/cran/PanelSelect

**RemoteRef** HEAD

**RemoteSha** cde983046a83ada8e68941987f09d7b2c2c59ae0

# Contents

PanelSelect-package . . . 2
add_information . . . 3
check_quadrature . . . 5
check_stationarity . . . 7
coef.panelselect . . . 9
formula.panelselect . . . 11
logLik.panelselect . . . 13
panel_control . . . 14
panel_select . . . 19
panelselect-methods . . . 25
predict.panelselect . . . 26
random_effects . . . 29
residuals.panelselect . . . 30
summary.panelselect . . . 32
update.panelselect . . . 33

**Index** **36**

---

`PanelSelect-package` *PanelSelect: panel sample-selection models*

---

## Description

PanelSelect provides a unified and flexible framework for sample selection models with continuous, binary, and count outcomes in panel and cross-sectional data. In particular, it supports a broad family of panel selection models, which extend the classic Heckman framework to account for persistent unobserved heterogeneity. The package supports selection at two levels: the individual level, through correlated random effects, and the observation level, through correlated occasion-specific shocks. This allows researchers to distinguish non-random selection driven by persistent individual characteristics from selection driven by temporary shocks. Alongside model estimation, PanelSelect supports prediction of population-level outcomes.

## Main functions

- `panel_select()` fits a model from an outcome formula, a selection formula (or `NULL` for a standalone model), the data, the panel ID column (or `NULL` for cross-sectional data) and the outcome family; its arguments also set restrictions, starting values, missing-data handling and the reported covariance. `panel_control()` sets the integration rules and the optimizer.
- Standard methods (panelselect-methods) summarize, compare and update fits; `predict.panelselect()`, `residuals.panelselect()` and `random_effects()` give predictions and posterior summaries.
- `check_quadrature()` compares likelihoods and scores at retained estimates under higher orders or an independent GL rule; agreement is evidence of stability, not an integration certificate.
- `check_stationarity()` reads stored scores and information, reporting `Q_OPG` and, with valid positive-definite stored observed information and its inverse, `Q_Hessian`. The default warning is strictly `Q_OPG > 0.1`. It does not compute a Hessian or alter backend convergence.

- add_information() explicitly returns a fit with observed information stored for reuse. Requested covariance can remain unavailable; optimizer convergence, stationarity, integration and inference require distinct checks.

Use panel_select() in place of the PanelSelect 1.x and PanelCount wrappers; vignette("migration", package = "PanelSelect") maps their calls and results to panel_select().

## Quick start

```
fit = panel_select(y ~ x, observed ~ .outcome + z, data, id = "person",
                   family = "linear")
summary(fit)
check_stationarity(fit)[c("Q_OPG", "Q_Hessian")]
predict(fit, newdata, type = "selected_mean")
panel_select(count ~ x, NULL, data, "person", "poisson")  # no selection
```

Start with vignette("vignette", package = "PanelSelect"); users of earlier versions should read vignette("migration", package = "PanelSelect"). GL supports population predictions; history-conditioned predictions and random-effect summaries require a GH/AGH fit, with no prediction-time integration override. History-conditioned prediction requires explicit history; random_effects() defaults to the retained fitting rows.

## Author(s)

**Maintainer**: Jing Peng <jing.peng@uconn.edu>

Authors:

- Jing Peng <jing.peng@uconn.edu>

---

# add_information *Compute and store observed information for a fitted model*

---

## Description

Augment an OPG fit with the observed Hessian and its covariance estimators, without refitting. Assign the returned object to retain the calculation for later inference and stationarity diagnostics.

## Usage

```
add_information(object)
```

## Arguments

| | |
|---|---|
| object | A fitted model of class panelselect, returned by panel_select(). Its fitted parameters, scores, controls and data define the information calculation. |

## Details

The default `vcov = "opg"` fit does not compute a post-fit observed Hessian. `add_information()` differentiates the full score at the fitted free optimizer coordinates to obtain the negative log-likelihood Hessian. This is the observed information of the fitted numerical objective; it does not certify quadrature accuracy, a global maximum or calibrated intervals.

The returned fit stores the information, any integration-accuracy record, its computed/unavailable status and error reason, and the observed, OPG and sandwich covariances on both optimizer and natural parameter scales. Its coefficients, likelihood and default covariance choice are unchanged. `check_stationarity()` can then report `Q_Hessian` and Newton diagnostics when the stored information and its inverse are finite and positive definite. Missing or invalid curvature remains unavailable; no OPG or sandwich matrix substitutes for the observed inverse.

Explicit observed/sandwich requests in `vcov()`, `confint()`, `summary()` or predictions can calculate missing information for that request. They do not store it in the caller's original object. Use `fit = add_information(fit)` once to reuse the stored calculation in subsequent requests.

A fit whose information is already marked as computed is returned unchanged, including when its stored information is unavailable. This also applies to information stored by older versions, including before the Gaussian outcome-unit correction. Use `update(fit, vcov = "observed")` to refit and recompute information with current numerical steps.

A staged GL fit whose retained mesh cannot serve the information stencil warns and stores an unavailable result. It does not silently refine the mesh or replace the estimator. An explicit refit with `vcov = "observed"` prepares a mesh for the information calculation, subject to the requested accuracy and resource limits.

## Value

A `panelselect` fit with stored information and covariance results, or the unchanged fit when information was already computed. Assign this return value to preserve the added information.

## See Also

panelselect-methods

Other diagnostic functions: `check_quadrature()`, `check_stationarity()`

## Examples

```
set.seed(1)
dat = data.frame(person = rep(1:60, each = 4), x = rnorm(240), z = rnorm(240))
u = rnorm(60); v = 0.5 * u + rnorm(60, sd = 0.8)  # correlated person effects
dat$observed = as.integer(0.3 + 0.5 * dat$z + 0.7 * u[dat$person] + rnorm(240) > 0)
dat$y = ifelse(dat$observed == 1, 1 + 0.5 * dat$x + v[dat$person] + rnorm(240), NA)
fit = panel_select(y ~ x, observed ~ x + z, dat, "person", "linear")
fit = add_information(fit)
sqrt(diag(vcov(fit, "observed")))
check_stationarity(fit)$Q_Hessian
```

check_quadrature *Check integration sensitivity at the fitted estimates*

## Description

Re-evaluate a fitted model using higher quadrature orders or an alternative GL rule. The comparisons hold parameters fixed, so they isolate changes in the numerical likelihood and score. A higher order is not guaranteed to be more accurate, and agreement is evidence of stability rather than a certificate.

## Usage

```
check_quadrature(
  object,
  nodes = NULL,
  observation_nodes = NULL,
  tolerance = c(loglik = 1e-05, score = 1e-04),
  refit = FALSE,
  compare = c("higher_order", "gl"),
  quadrature_options = list(),
  ...
)

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

## Arguments

| | |
|---|---|
| object | A fitted panelselect model. Its retained data, restrictions and parameters define every comparison; the input object is not modified. |
| nodes | Person-effect orders for the comparison. With compare = "higher_order", the default adds the larger of 5 or half the fitted order (rounded up) to each fitted order. A scalar uses the same order on both effects; a pair follows selection then outcome. Supplied orders cannot decrease. With compare = "gl", this is the GL order per segment (default 20), not a total node budget. |
| observation_nodes | PLN observation order for the comparison: twice the fitted order by default for "higher_order", or GL order 20 for "gl". Only active PLN observation integration contributes a comparison row. |
| tolerance | Named positive numeric vector with loglik (default 1e-5) and score (default 1e-4). The log-likelihood tolerance applies to the absolute total change and the largest absolute independent-unit change; the score tolerance applies to the largest absolute total and unit optimizer-score component changes. These are comparison tolerances, not optimizer stopping controls or certified integral-error bounds. |

| | |
|---|---|
| `refit` | Logical; default `FALSE` performs only fixed-parameter checks. `TRUE` additionally calls `update()` with the compared controls and fitted natural coefficients as warm starts, retaining the resulting fit in `$refit`. Fitting errors are not silently replaced by a different method. |
| `compare` | `"higher_order"` (default) refines the fitted rule; `"gl"` compares a GH/AGH person-effect fit against a separately constructed GL rule, also using GL for an active PLN observation integral. Fits already using GL should use `"higher_order"`. See the comparison table below. |
| `quadrature_options` | Named GL options, with names and meanings from `panel_control()`, used only for `compare = "gl"`. Defaults are obtained from a GL control; increase bounds or budgets explicitly when a comparison reports the corresponding limit. |
| `...` | Unused by the print method. `check_quadrature()` rejects unknown arguments and gives replacements for earlier argument names. |
| `x` | A quadrature check to print. |
| `digits` | Significant digits for the printed comparison; default `3`. The stored discrepancies retain full precision. |

## Value

A `panelselect_quadrature_check` list with `compare`, `fitted` (a fitted-rule description), the `comparison` data frame, `tolerance`, `optimizer_converged`, `covariance_valid`, `integration`, `failures`, `refit` (`NULL` by default), and `qualification`. `print()` displays the main comparison columns and returns the object invisibly. Covariance validity describes the requested fitted estimator, not statistical calibration.

## Choose a comparison

| Option | What changes |
|---|---|
| `"higher_order"` | Orders increase within the fitted integration methods. Staged rules can require refined meshes. |
| `"gl"` | GH/AGH person-effect integration is compared with GL at the same estimates, using moving cuts or a fre |

For PLN fits, `person_effect` changes only the outer integral, `observation` only the occasion-level integral, and `both` changes both where both are active. Fixed restrictions and exact reductions can remove an integral, so only effective changes produce rows. A fully closed-form model has no numerical integral to refine and the function stops with that explanation.

## Read the results and decide what to do

| Comparison column | Meaning |
|---|---|
| `rule` | `person_effect`, `observation`, `both`, or `gl`. |
| `nodes`, `reference_nodes` | Compared and fitted person-effect orders, respectively. |
| `observation_nodes`, `observation_quadrature` | Compared PLN inner rule when applicable. |
| `mesh_changed` | Whether a staged comparison changed the retained mesh; `NA` when not ap |
| `loglik_change` | Signed change in the total log likelihood. |

| | |
|---|---|
| `max_panel_change` | Largest absolute independent-unit log-likelihood change. |
| `max_score_change`, `max_panel_score_change` | Largest absolute total and individual-unit optimizer-score changes. |
| `integration_failed`, `message` | Numerical failure and its explanation, separate from tolerance exceedance |
| `within_tolerance` | Whether all four discrepancies meet their tolerances; `NA` after numerical f |

- If discrepancies exceed tolerance, investigate larger orders or another method and assess changes in estimates/uncertainty after an explicit refit.
- If integration fails, inspect `message`, `$failures` and `$integration`; change only the named controls whose limits or accuracy require attention.
- Even small changes do not prove both rules accurate. Parameter-dependent integration error can move an optimum despite correct finite-rule scores. Compare competing endpoints under a common checked rule when needed.
- Keep optimizer status, `check_stationarity()`, covariance availability, and substantive sensitivity separate from this integration comparison.

## See Also

`panel_control()`, `check_stationarity()`, `update.panelselect()`

Other diagnostic functions: `add_information()`, `check_stationarity()`

## Examples

```
set.seed(1)
dat = data.frame(person = rep(1:60, each = 4), x = rnorm(240), z = rnorm(240))
u = rnorm(60); v = 0.5 * u + rnorm(60, sd = 0.8)  # correlated person effects
dat$observed = as.integer(0.3 + 0.5 * dat$z + 0.7 * u[dat$person] + rnorm(240) > 0)
dat$y = ifelse(dat$observed == 1, 1 + 0.5 * dat$x + v[dat$person] + rnorm(240), NA)
fit = panel_select(y ~ x, observed ~ x + z, dat, "person", "linear")
check_quadrature(fit)
check_quadrature(fit, nodes = 20, tolerance = c(loglik = 1e-6, score = 1e-5))
```

---

# `check_stationarity` *Inspect score and curvature diagnostics without refitting*

---

## Description

Calculate stationarity diagnostics from the fitted object's stored scores and covariances. The primary screen is `Q_OPG`; when valid observed information is stored, `Q_Hessian` and local Newton diagnostics are also available. No likelihood, score or Hessian is evaluated by this function.

## Usage

```
check_stationarity(object)
```

## Arguments

| | |
|---|---|
| `object` | A fitted `panelselect` model. An OPG-only fit normally has no observed information: assign `fit = add_information(fit)` if you explicitly want to calculate and retain it. Merely inspecting a fit does not add it. |

## Value

A list with top-level numeric `Q_OPG` and `Q_Hessian` (`NA` when unavailable), backend convergence/code/status, named optimizer `score`, the curvature/Newton quantities above, `information_error`, and retained rejected-trial summaries. `opg` contains `Q_OPG`, `status`, `tolerance`, `warning`, `reason`, and `basis`. Top-level `status` and `reason` concern the observed-information diagnostics; they can be unavailable while `opg$status` is available. The supplied fit is unchanged.

## Diagnostic quantities

Let $g$ be the total free-parameter score in optimizer coordinates, $B$ the sum of independent-unit score outer products, and $A = -\nabla^2 \ell$ the observed information for the fitted numerical objective.

| Quantity | Definition and interpretation |
|---|---|
| `Q_OPG` | $g^\mathsf{T} B^{-1} g$. Screens score size relative to independent-unit score vari |
| `Q_Hessian` | $g^\mathsf{T} A^{-1} g$. Local curvature diagnostic, reported only with valid posit |
| `newton_step` | $A^{-1} g$. Local quadratic estimate of the optimizer-coordinate displac |
| `standardized_step` | Newton components divided by the square roots of the correspondi |
| `max_abs_score`, `max_standardized_step` | Largest absolute score or standardized Newton component. |
| `information_eigenvalues`, `information_condition` | Curvature eigenvalues and, for positive-definite information, their r |

Optimizer coordinates are coefficients, log-SDs and Fisher-transformed correlations. The Hessian and Newton interpretation is local to these coordinates; it is not generally invariant to nonlinear reparameterization away from a stationary point. Neither Q is a p-value or a global-optimality test.

## Warning policy and unavailable results

- The default warning cutoff is `Q_OPG > 0.1`, strictly. Change `panel_control(stationarity_warning = ...)` to use another Q cutoff; `NULL` disables the warning while preserving the statistic. This is a diagnostic policy, not an optimizer stopping criterion.
- There is no additional automatic cutoff for `Q_Hessian`.
- Missing, nonfinite, singular or indefinite observed information/inverse leaves `Q_Hessian` and the Newton quantities unavailable with a reason. OPG and sandwich covariance are never substituted for observed information.
- An all-fixed model has status `"all_fixed"` and no free-parameter stationarity quantities. An unavailable OPG screen is not an automatic pass.
- Native saved fits use current Q semantics and their stored scores/covariances. Previously stored Gaussian Hessians remain historical calculations; an explicit update/refit obtains corrected outcome-scale information.

A small Q, or a backend convergence code, does not certify integration accuracy, identification, calibrated intervals or a global maximum. Near parameter boundaries the screen can overstate

remaining displacement. Use `check_quadrature()` for integration sensitivity, and explicit tighter optimizer settings or different starts when investigating optimization.

## See Also

`panel_control()`, `add_information()`, `check_quadrature()`

Other diagnostic functions: `add_information()`, `check_quadrature()`

## Examples

```
set.seed(1)
dat = data.frame(person = rep(1:60, each = 4), x = rnorm(240), z = rnorm(240))
u = rnorm(60); v = 0.5 * u + rnorm(60, sd = 0.8)  # correlated person effects
dat$observed = as.integer(0.3 + 0.5 * dat$z + 0.7 * u[dat$person] + rnorm(240) > 0)
dat$y = ifelse(dat$observed == 1, 1 + 0.5 * dat$x + v[dat$person] + rnorm(240), NA)
fit = panel_select(y ~ x, observed ~ x + z, dat, "person", "linear")
check_stationarity(fit)$opg
check_stationarity(add_information(fit))$max_standardized_step
```

---

# coef.panelselect *Extract coefficients, covariance matrices and confidence intervals*

---

## Description

Inspect the natural-scale estimates of a `panel_select()` fit, select a covariance estimator, or obtain parameter confidence intervals. These methods do not refit the model. Observed and sandwich covariance can require an additional information calculation at the fitted estimates.

## Usage

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

## S3 method for class 'panelselect'
vcov(object, type, vcov, ...)

## S3 method for class 'panelselect'
confint(object, parm, level = 0.95, vcov = object$control$vcov, ...)
```

## Arguments

| | |
|---|---|
| `object` | A fitted `panelselect` object from `panel_select()`. |
| `...` | `coef()` ignores extra arguments. `vcov()` and `confint()` reject unused arguments, including misspelled covariance or interval options. |
| `type` | Covariance estimator for `vcov()`: `"opg"`, `"observed"`, or `"sandwich"`. Omit it to use the covariance choice stored in the fit. See the table below for the calculation and when additional work is needed. |

| | |
|---|---|
| vcov | In confint(), an estimator name ("opg", "observed", or "sandwich") or a finite, named natural-scale covariance matrix; the default is the fit's choice. In vcov(), this is an alternative argument name for type and accepts only an estimator name. If both names are supplied to vcov(), their values must agree. |
| parm | Parameters to include in confint(): a character vector of names, all-positive inclusion indices, all-negative exclusion indices, or a logical vector with one entry per coefficient. Omit to include every parameter. Use names(coef(object)) to find the available names. |
| level | Confidence level strictly between zero and one; default 0.95. |

## Value

coef() returns a named numeric vector. vcov() returns a square named natural-scale covariance matrix. confint() returns a two-column matrix of lower and upper limits for the requested parameters.

## Choosing a covariance estimator

Let each independent unit contribute a score vector, let B be the sum of their outer products, and let A be the observed information. Independent units are people in a panel model and rows in a cross-sectional model.

| Choice | Calculation | Common use and computation |
|---|---|---|
| "opg" | Inverse of B | Default likelihood-based covariance; uses the stored unit scores and needs |
| "observed" | Inverse of A | Curvature-based likelihood inference; computes information if it is not alre |
| "sandwich" | solve(A) %*% B %*% solve(A) | Independent-unit sandwich inference; also requires valid observed informa |
| Named matrix | Supplied natural covariance | Available to confint(), summary(), and predict(), for externally calcul |

Assign fit = add_information(fit) to retain information for repeated observed/sandwich requests. Information that is singular, indefinite, or numerically unavailable is not replaced by a different estimator. Staged GL uses the retained mesh; if information needs a refined mesh, explicitly update/refit with vcov = "observed" or "sandwich".

## Parameter scales and intervals

- Regression coefficients are reported on the model's coefficient scale.
- SDs are positive natural SDs; intervals are constructed on the log scale and transformed back. Testing an SD against zero is a boundary problem, so summaries do not supply a usual zero-test p-value for SDs.
- Correlations are reported between -1 and 1; intervals and summary tests use the Fisher (atanh) scale and transform interval limits back.
- Fixed parameters remain in coef() and have zero covariance rows/columns. Their inferential entries, including confidence limits, are NA.

These are Wald intervals. Stored optimizer-coordinate covariance is used when available; supplied matrices and older fits without it use the natural-scale delta method. Numerical convergence alone does not establish interval coverage.

## See Also

summary.panelselect(), add_information(), check_stationarity()

## Examples

```
dat = data.frame(person = rep(1:20, each = 3), y = rep(c(0, 1, 3, 1, 2, 4), 10))
fit = panel_select(y ~ 1, NULL, dat, "person", "poisson",
                   fixed = c(sd_re_outcome = 0))
coef(fit)
vcov(fit)
confint(fit, parm = "outcome.(Intercept)")
fit = add_information(fit)
vcov(fit, type = "observed")
```

---

# formula.panelselect *Recover fitted formulas, designs and omitted rows*

---

## Description

Inspect either equation of a fitted model using its retained data, factor levels and contrasts. The methods describe the fitted design rather than constructing a design for new prediction data.

## Usage

```
## S3 method for class 'panelselect'
formula(x, equation = c("outcome", "selection"), ...)

## S3 method for class 'panelselect'
model.frame(formula, equation = c("outcome", "selection"), ...)

## S3 method for class 'panelselect'
terms(x, equation = c("outcome", "selection"), ...)

## S3 method for class 'panelselect'
na.action(object, ...)

## S3 method for class 'panelselect'
model.matrix(object, equation = c("outcome", "selection"), ...)
```

## Arguments

| | |
|---|---|
| x, object | A fitted panelselect model. formula() and terms() call this argument x; model.matrix() and na.action() call it object. |
| equation | Equation to extract: "outcome" (default) or "selection". A standalone model has no selection equation: formula() returns NULL, while terms(), model.frame() and model.matrix() stop for that request. |

| | |
|---|---|
| `...` | Extra generic arguments; these methods do not use them. In particular, they do not accept new data or a replacement missing-data policy. |
| `formula` | A fitted `panelselect` model, despite the generic argument name, when calling `model.frame()`. |

## Value

A formula, terms object, data frame, numeric design matrix, or `omit`/`exclude` row-index object, according to the generic.

## Returned designs

- `formula()` returns the fitted equation formula, with `.outcome` selection shorthand expanded to the regressors used by the fit.
- `terms()` returns its terms object, including offset information.
- `model.frame()` rebuilds the retained rows in input order, using the fitted factor levels and allowing structural missing outcome entries.
- `model.matrix()` constructs the corresponding coefficient design with the fitted contrasts. Offsets belong to the formula/model frame, not additional estimated-coefficient columns.
- `na.action()` identifies rows omitted or excluded for required missing data; it returns `NULL` when none were dropped. Rows outside `subset` are separate.

Outcome values/regressors unused on unselected rows can remain missing in the returned frame. The likelihood's internal zero placeholders are not imputed outcomes. Categories represented only among unselected rows can produce an unidentified outcome coefficient and a rank-deficiency error; levels are not silently dropped as a model-selection step. Prepare an estimable outcome design and check new-data factor support explicitly.

## See Also

`panel_select()`, `predict.panelselect()`, `update.panelselect()`

## Examples

```
dat = data.frame(person = rep(1:20, each = 3), y = rep(c(0, 1, 3, 1, 2, 4), 10))
fit = panel_select(y ~ 1, NULL, dat, "person", "poisson",
                   fixed = c(sd_re_outcome = 0))
formula(fit)
head(model.frame(fit))
head(model.matrix(fit))
na.action(fit)
```

`logLik.panelselect` *Inspect likelihood, sample size and information criteria*

## Description

Extract the fitted joint log likelihood or compare models fitted to the same sample and response definition. Within-person dependence is included in each panel's integrated likelihood contribution; observations within a person are not treated as independent likelihood units.

## Usage

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

## S3 method for class 'panelselect'
nobs(object, ...)

## S3 method for class 'panelselect'
AIC(object, ..., k = 2)

## S3 method for class 'panelselect'
BIC(object, ..., unit = c("panels", "observations"))
```

## Arguments

| | |
|---|---|
| `object` | A fitted `panelselect` model. For a multiple-model AIC/BIC table, this is the first model to compare. |
| `...` | Additional fitted models in `AIC()` and `BIC()`. They must supply compatible `logLik()` and `nobs()` methods and represent comparable likelihoods. Extra arguments are ignored by `logLik()` and `nobs()`. |
| `k` | Penalty per free parameter in `AIC()`; default `2` gives ordinary AIC. Supply one finite number. Fixed parameters do not contribute to the parameter count. |
| `unit` | Sample-size convention for `BIC()`: `"panels"` (default) uses independent people for panel models and independent rows for cross-sectional models; `"observations"` uses retained rows for every model. |

## Value

`logLik()` returns a `logLik` scalar with `df`, `nobs` and `n_rows` attributes. `nobs()` returns the retained row count. AIC/BIC return one number for one model or a data frame with `df` and the criterion for several.

## Calculations and sample counts

If `ell` is the fitted log likelihood and `p` is the number of free parameters, `AIC = -2 * ell + k * p` and `BIC = -2 * ell + log(n) * p`.

| Quantity or option | Count |
|---|---|
| `nobs(fit)` | Retained rows, including observed nonparticipation rows in a selection model. |
| `attr(logLik(fit), "nobs")` | Independent people for a panel model; retained rows for a cross-sectional model. |
| `attr(logLik(fit), "n_rows")` | Retained rows. |
| `attr(logLik(fit), "df")` | Estimated coefficients, SDs and correlations; fixed parameters excluded. |
| `BIC(fit, unit = "panels")` | Uses the independent-unit `nobs` attribute of `logLik()`. |
| `BIC(fit, unit = "observations")` | Uses `nobs(fit)` instead. |

### Comparing models responsibly

- Use the same retained sample and response definition. Equal sample counts do not establish sample identity, and criteria from different likelihood definitions need not be comparable.
- The existing guard warns for different row counts, and for default BIC also different independent-unit counts. It does not compare row identities.
- Check integration sensitivity and optimization before interpreting small criterion differences. A higher-order fit need not be a more accurate fit.
- Use likelihood-ratio tests only with the appropriate nesting, identified parameters and boundary conditions; see `panel_select()`.

### See Also

`panel_select()`, `update.panelselect()`, `check_quadrature()`

### Examples

```
dat = data.frame(person = rep(1:20, each = 3), y = rep(c(0, 1, 3, 1, 2, 4), 10))
fit = panel_select(y ~ 1, NULL, dat, "person", "poisson",
                   fixed = c(sd_re_outcome = 0))
logLik(fit)
c(rows = nobs(fit), people = attr(logLik(fit), "nobs"))
AIC(fit)
BIC(fit)
BIC(fit, unit = "observations")
```

---

## `panel_control` *Control numerical integration and optimization*

---

### Description

Collects numerical settings for a panel selection fit with `panel_select()`, also used by its cross-sectional selection and standalone models. Person effects use fixed Gauss-Hermite (GH), 10 nodes per effect, by default; the separate PLN observation integral uses fixed GH with 20 nodes. The default optimizer is `optim` BFGS with 500 iterations. These are starting settings to assess, not guarantees of accuracy. Missing-data handling, initialization and the reported covariance belong to `panel_select()` (`na_action`, `init`, `vcov`).

## Usage

```
panel_control(
  quadrature = c("gh", "agh", "gl"),
  nodes = NULL,
  quadrature_options = list(),
  observation_quadrature = c("gh", "agh", "gl"),
  observation_nodes = NULL,
  observation_options = list(),
  mode_maxit = 50L,
  mode_tol = 1e-09,
  exact_reductions = TRUE,
  optimizer = c("optim", "maxLik"),
  method = "BFGS",
  maxit = 500L,
  reltol = sqrt(.Machine$double.eps),
  factr = 1e+07,
  parscale = NULL,
  trace = 0L,
  stationarity_warning = 0.1,
  ...
)

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

## Arguments

| | |
|---|---|
| quadrature | Person-effect integration rule: "gh" (fixed GH, default), "agh" (adaptive GH), or "gl" (Gauss-Legendre on segments). See Integration rules for how they differ. Cross-sectional models have no person-effect integral. |
| nodes | Positive integer GH order per person effect, or one GL order per segment between 3 and 256. NULL chooses 10 for GH/AGH and 20 for GL. GH/AGH also accept two orders in selection/outcome order, optionally named c(selection = 8, outcome = 12); GL requires one order. |
| quadrature_options | Uniquely named list of GL person-effect options; see GL options for every name, default and meaning. Must be empty for GH/AGH. A supplied list overrides only its named GL option defaults. |
| observation_quadrature | PLN observation-effect rule: "gh" (default), "agh" or "gl", selected independently of quadrature. Other families do not use this rule; controls are still validated. |
| observation_nodes | Positive integer PLN observation order. NULL chooses 20 for every rule; for GL it is the per-segment order, from 3 to 256. |
| observation_options | Uniquely named list of GL observation options. It accepts the same names as |

quadrature_options, with a smaller default batch_nodes. Must be empty for a GH/AGH observation rule.

mode_maxit — Positive integer iteration limit for adaptive-GH person-effect mode searches (50). It does not control the model optimizer or the separate PLN observation-effect mode search.

mode_tol — Positive residual tolerance of adaptive-GH person-effect mode searches (1e-9). Non-PLN person-effect modes are polished to min(mode_tol, 1e-12), accepting a roundoff floor within mode_tol; PLN modes use mode_tol. A relative residual above 10 * mode_tol fails the evaluation.

exact_reductions — Logical, default TRUE. Use exact likelihood and integration reductions implied by the model and fixed parameters. FALSE requests the general route for numerical comparisons; it does not change the restrictions or restore an unidentified parameter.

optimizer — Backend: "optim" (default) or "maxLik". The latter requires the optional maxLik package. See Optimizer methods.

method — Backend method name, default "BFGS"; the available names differ by backend and are listed below.

maxit — Nonnegative integer optimizer iteration limit (500), passed as maxit to optim and iterlim to maxLik. Zero evaluates the starting point without claiming optimizer convergence.

reltol — Positive relative stopping tolerance (default sqrt(.Machine$double.eps)). Passed to maxLik and to optim methods other than L-BFGS-B; the automatic fixed-GH binary BFGS exception is below.

factr — Positive optim L-BFGS-B function-reduction tolerance factor (1e7), relative to machine precision. Other supported methods do not use it.

parscale — NULL for automatic BFGS scaling, 1 for an unscaled metric, or positive finite optimizer-coordinate scales. Supply one value or one per free parameter; a named vector must match the free parameter names. Native maxLik Newton/BHHH/BFGSR methods do not accept parscale.

trace — Nonnegative integer backend tracing level; 0 is silent. Passed as trace to optim and printLevel to maxLik.

stationarity_warning — Positive threshold for Q_OPG = g' B^{-1} g, where g is the total free-parameter score and B its summed independent-unit score outer products in optimizer coordinates. The default warns strictly when Q_OPG > 0.1. NULL disables this warning, while retaining the diagnostic. This is separate from backend stopping, integration checks and covariance validity; see check_stationarity().

... — Not used. Removed settings (binary_integration, checked, fixed_geometry, and GK), misplaced fit settings and earlier argument names stop with messages naming their replacements. Other names are errors.

x — A panelselect_control list to print. Printing shows the selected integration rules, optimizer settings and Q warning threshold.

## Value

A panelselect_control list of validated settings, printed by print().

### Integration rules

| Rule | Calculation |
|---|---|
| `"gh"` (default) | Fixed standard-normal nodes; order stays fixed during optimization. |
| `"agh"` | Centres and scales nodes at each panel's posterior mode and curvature; scores include node, curvature and vo |
| `"gl"` | GL on segments, with a distinct order for empirical refinement checks and separate analytic tail envelopes. |

Analytical scores differentiate the finite rule actually optimized. Correct differentiation and backend convergence do not establish accuracy of the continuous integral. Higher order is not uniformly more accurate; inspect likelihood and score sensitivity with `check_quadrature()`, especially for large scales, near-boundary correlations and rare response histories. Equal GH/GL orders are neither equal work nor matched accuracy, because GL uses that many nodes in each segment.

PLN has two integration levels. Choosing outer AGH or GL does not change the fixed-GH observation default. Coarse fixed observation nodes can make the person-effect integrand corrugated; `observation_quadrature = "agh"` differentiates its own moving observation geometry and can help in some cases, with additional computation. Select both rules explicitly for GL at both levels. Refine observation and person-effect orders separately and then jointly before interpreting changed estimates.

Exact reductions skip integrals made unnecessary by fixed zero scales or correlations. Cross-sectional selection without PLN is closed form. A selected rule can therefore do no integration work in a reduced model.

### GL fitting and limits

Standalone panel probit and selected binary/Linear panels with `cor_error` fixed at zero use GL with analytically differentiated moving cuts. Other GL routes keep a positive mesh fixed within each optimization and information stage. A failed staged gate at any trial point ends that stage; the next stage starts from its best valid point on a refined mesh. At most three stages are allowed, including stages needed for requested covariance. Unresolved preparation or exhausted limits fail explicitly and retain diagnostic records. No rule, tolerance or covariance estimator is silently substituted.

Staged GL can be substantially more expensive than GH/AGH because it prepares, checks and refines meshes. Cost depends on the model, active dimensions, restrictions and accuracy target; the moving-cut routes and staged PLN routes do not share a universal timing ratio. Physical moving-cut GL can also be difficult when the person-effect correlation approaches -1 or 1. Refinement agreement remains empirical evidence, not a certified integration-error bound.

GL observed information must pass step/order checks and, when positive definite, inverse/SE stability checks; otherwise it remains unavailable. GL supports population prediction, serialization and explicit update/refit. History-conditioned prediction and `random_effects()` are unavailable when GL is selected for person effects or the PLN observation rule. There is no prediction-time integration override; use an explicit GH/AGH refit if those targets are required.

### GL options

Both option lists accept the following positive finite scalar settings. Orders and capacity controls must be integers. These options are checked against the selected rule; unknown or misspelled names are errors.

| Option | Default | Meaning |
|---|---|---|
| reference_nodes | max(2, floor(nodes/2)) | Distinct GL comparison order, from 2 to 256; may be above or below t |
| bound | 12 | Integration-domain half-width in standard-normal coordinates. The do |
| panel_loglik_tol | 1e-7 | Per-panel log-likelihood refinement tolerance. |
| panel_score_tol | 1e-5 | Per-panel first-score refinement tolerance, with coordinate-specific scal |
| tail_tol | 1e-9 | Relative omitted probability-tail tolerance. |
| total_loglik_tol | 1e-5 | Aggregate log-likelihood discrepancy budget across panels. |
| total_score_tol | 1e-4 | Aggregate first-score discrepancy budget across panels. |
| information_tol | 2e-3 | Step/order stability tolerance for requested observed information and, v |
| seconds_per_panel | 60 | Wall-time limit for one panel's integration work in a preparation or eva |
| seconds_total | 300 | Wall-time limit across panels in a preparation or evaluation. |
| max_nodes | 1e8 | Integration-point cap for a panel rule, including tensor-product work w |
| max_cells | 4096 | Capacity limit for partition refinement on staged routes. |
| batch_nodes | 8192 outer; 128 observation | Target node batch size for streamed calculations; a complete cell can ex |

Score budgets account for coordinate units: coefficient budgets use each regressor's root mean square. A small canonical/reference discrepancy is an empirical interior-error estimate; analytic omitted-tail envelopes are separate. More interior nodes cannot resolve an omitted-tail bound outside the chosen domain. Increase the named resource or domain limit explicitly when justified, then repeat the comparison without relaxing its tolerances. Observation time limits accumulate work across a person's observed rows, including observation-mesh refinement. They are not a fresh allowance for each row. Time guards run between bounded numerical blocks, so an indivisible block can finish after its deadline before the failure is raised. With outer GH/AGH and observation GL, the overall guard is the larger of 300 seconds and the supplied observation total limit. See the staged-quadrature mathematical chapter for the full budget accounting.

## Optimizer methods

| Backend | Accepted methods |
|---|---|
| "optim" | "BFGS" (default), "Nelder-Mead", "CG", "L-BFGS-B", "SANN" |
| "maxLik" | "BFGS" (default), "NR"/"Newton-Raphson", "BFGSR"/"BFGS-R", "BHHH", "CG"/"Conjugate-Gradient", "NM"/" |

Optimizer coordinates are unrestricted coefficients, log SDs and Fisher correlations; supplied starts and reported estimates remain natural-scale. optim minimizes the negative log likelihood. The backend status reports that method's termination condition. SANN has no convergence test, and a zero-iteration run is reported as not run; neither is labelled converged. Newton methods may require curvature during optimization even with OPG covariance. No method guarantees a global maximum.

Without explicit scales, BFGS uses the reciprocal initial panel-score column norms, 1 / sqrt(colSums(score^2)), for regression coefficients with non-negligible initial scores. Other coordinates use 1 / pmax(1, sqrt(colSums(score^2))); all scales are bounded below by sqrt(.Machine$double.eps). This respects coefficient units under rescaled regressors and Linear outcomes. Fixed-GH panel binary selection with free cor_error and otherwise default optim BFGS settings instead uses parscale = 1/sqrt(n_panels) with reltol = 1e-12. Explicit parscale or reltol prevents that automatic

pair. The random-effects auxiliaries used by `init = "default"` have fresh BFGS optimizer settings; the joint fit's optimizer options do not control them. Numerical auxiliaries inherit the equation's person-effect integration settings, and PLN also inherits the observation settings. The internal Linear auxiliary uses an exact Gaussian likelihood without quadrature. See `panel_select()` for supplied-start precedence and initialization fallback.

The fit's `control` keeps requested settings and `optimizer$control` records effective backend options. `update()` retains the fitted rules and settings unless explicitly replaced; a fresh `panel_control(...)` is a complete control replacement, not a partial patch to the old control.

## See Also

`check_quadrature()`

Other fitting functions: `panel_select()`

## Examples

```
panel_control()
panel_control(nodes = c(selection = 8, outcome = 12))
panel_control(quadrature = "agh", nodes = 15)
panel_control(quadrature = "gl", quadrature_options = list(bound = 16))
panel_control(optimizer = "optim", method = "L-BFGS-B", factr = 1e6)
panel_control(stationarity_warning = NULL)
```

---

# panel_select *Fit panel and cross-sectional sample-selection models*

---

## Description

`panel_select()` fits panel and cross-sectional sample-selection models with probit selection and linear, binary, or count outcomes. It also supports standalone random-effects panel models for binary and count outcomes.

Panel selection models allow dependence between selection and outcomes through correlated individual-level random effects, correlated observation-level errors, or both. These models are described in Bailey and Peng (2025) and Peng and Van den Bulte (2024).

Estimation uses maximum likelihood with analytical gradients and numerical integration of random effects. The mathematical engine is deeply optimized using C++. The table below summarizes the available models.

## Usage

```
panel_select(
  outcome,
  selection,
  data,
  id,
  family,
  start = NULL,
```

```
  fixed = NULL,
  control = panel_control(),
  subset = NULL,
  na_action = c("fail", "omit", "exclude"),
 init = c("default", "re", "pooled", "cross_sectional", "zero", "unif", "norm"),
  vcov = c("opg", "observed", "sandwich")
)
```

## Arguments

| | |
|---|---|
| `outcome` | Outcome formula, such as `y ~ x1 + x2`, with optional `offset()` terms. Binary responses must be zero/one, counts nonnegative integers, and linear responses finite numeric values on the rows where the outcome is observed. |
| `selection` | Formula for the selection model, such as `observed ~ x1 + x2 + z` or `observed ~ .outcome + z`, where `.outcome` stands for all outcome regressors and `z` represents an optional regressor affecting selection but not outcome. Set to `NULL` for a standalone probit, Poisson, or Poisson-lognormal random-effects model without selection. `.outcome` copies the outcome regressor terms, but not its offsets or intercept choice. |
| `data` | A data frame containing the variables used by the formulas, along with a column indicating panel ID for panel models. Outcome observations on unselected rows are unused and may be missing. Selection regressors are required on every retained row; see Formulas and missing data. |
| `id` | Name of the panel ID column for a panel model. Set `id = NULL` to fit a cross-sectional sample-selection model. Supplying an ID column always fits a panel model, even if each ID occurs only once. Standalone random-effects models (`selection = NULL`) also require an ID column. |
| `family` | Outcome model: `"linear"` for linear regression with normal errors (aliases `"continuous"` and `"gaussian"`), `"binary"` for probit regression (alias `"probit"`), or `"count"` for Poisson-lognormal regression (alias `"pln"`). Panel models include person-level random effects. The Poisson-lognormal model also includes a normal observation-level effect on the log rate, allowing additional overdispersion and, in selection models, correlation with the selection error (`cor_error`). In panel data, the `"poisson"` option becomes available and fits a Poisson model with person-level random effects, but no observation-level random effects. This option still permits overdispersion through person-level heterogeneity and allows selection-outcome dependence only through correlated person effects (`cor_re`). |
| `start` | `NULL` for default starts, or a finite, uniquely named numeric vector of natural-scale values. Partial vectors replace only those starts; a complete free-parameter vector bypasses default initialization. Free SDs must start positive and correlations strictly between -1 and 1. See Models for parameter names. |
| `fixed` | Named parameter values, on their natural scales, to hold fixed during estimation. For example, `fixed = c(cor_error = 0)` sets the correlation between the selection and outcome errors within each observation to zero, which can substantially speed up estimation. Values supplied in both `start` and `fixed` must agree. See Restrictions with `fixed`. |

`control` Numerical integration and optimization settings from `panel_control()`. By default, random effects are integrated using fixed Gauss-Hermite quadrature with 10 nodes per person effect and 20 nodes for Poisson-lognormal observation effects. The default optimizer is `optim` with the BFGS method and a maximum of 500 iterations. For binary panel outcomes, Gauss-Legendre quadrature is recommended when the outcome random-effect SD (`sd_re_outcome`) is large (e.g., 5 or higher). It can improve accuracy, often at greater computational cost. For example, `control = panel_control(quadrature = "gl", nodes = 15)` uses 15 nodes per segment with Gauss-Legendre quadrature.

`subset` Optional, subset of rows to use for estimation, specified as a logical vector, row indices, or row names. Expressions are evaluated in `data`, so column names can be used directly (e.g., `age >= 18`). `NA` entries in a logical subset are handled according to `na_action`.

`na_action` Handling of missing required values: `"fail"` (default) stops and identifies rows; `"omit"` drops them; `"exclude"` drops them and restores `NA` positions in fitting-row predictions and residuals. This does not drop structurally missing unselected outcomes.

`init` Initialization method for free parameters not supplied in `start`. `"default"` (alias `"re"`) uses estimates from separate random-effects selection and outcome models for panel selection, and from pooled cross-sectional regressions for standalone and cross-sectional models. `"pooled"` (alias `"cross_sectional"`) always uses pooled cross-sectional regressions. `"zero"` sets regression coefficients and correlations to zero, and standard deviations to one. `"unif"` and `"norm"` draw initial values from Uniform(0, 1) and Normal(0, 1), respectively. Standard deviations use `exp(draw)` and correlations use `tanh(draw)`. Use `set.seed()` for reproducible random starts. See Starting values for details.

`vcov` Covariance estimator used for standard errors and inference. `"opg"` (default) uses the inverse of the summed independent-unit score outer products and avoids a post-fit Hessian calculation. `"observed"` uses the inverse observed information (the negative Hessian of the log likelihood). `"sandwich"` combines the observed information with the score outer products to provide asymptotically robust standard errors, clustered by panel ID for panel models. Selecting `"observed"` or `"sandwich"` requests a numerical Hessian calculation after optimization, which adds computation time. See Inference for assumptions and details on computing the Hessian after fitting.

## Value

An object of class `panelselect`. Use summary() to view the estimation results, including parameter estimates and standard errors. The object contains these main components:

- `coefficients`: named regression coefficients, SDs and correlations on natural scales, including fixed values. Extract them with coef().
- `covariance`: stored parameter covariance matrices. Use vcov() and summary() for uncertainty estimates with the chosen estimator.
- `loglik`: total log likelihood at the returned estimates; extract with logLik().
- `optimizer`: convergence and stationarity information; inspect with `check_stationarity()`.

- `integration`: available numerical integration diagnostics. Assess sensitivity to the integration rule and node counts with `check_quadrature()`.
- `control` and `fixed`: fitting settings and parameter restrictions. Modify them through `update()`.

See panelselect-methods for the full set of fitted-model methods.

## Models

| Model | family | selection | id | Structural parameters |
|---|---|---|---|---|
| ProbitRE_LinearRE | "linear" | Yes | Yes | sd_re_selection, sd_re_outcome, sd_error, cor_re, cor_err |
| ProbitRE_ProbitRE | "probit" | Yes | Yes | sd_re_selection, sd_re_outcome, cor_re, cor_error |
| ProbitRE_PoissonRE | "poisson" | Yes | Yes | sd_re_selection, sd_re_outcome, cor_re |
| ProbitRE_PLNRE | "pln" | Yes | Yes | sd_re_selection, sd_re_outcome, sd_error, cor_re, cor_err |
| Probit_Linear | "linear" | Yes | No | sd_error, cor_error |
| Probit_Probit | "probit" | Yes | No | cor_error |
| Probit_PLN | "pln" | Yes | No | sd_error, cor_error |
| ProbitRE | "probit" | No | Yes | sd_re_outcome |
| PoissonRE | "poisson" | No | Yes | sd_re_outcome |
| PLNRE | "pln" | No | Yes | sd_re_outcome, sd_error |

In the `selection` and `id` columns, Yes means that the argument is required, while No means it must be `NULL`.

Regression coefficients use `selection.<column>` and `outcome.<column>`. `sd_re_selection` and `sd_re_outcome` are individual-effect SDs, with correlation `cor_re`. `sd_error` is the residual SD for Linear outcomes or the observation-level random effect SD for PLN outcomes. `cor_error` measures selection-outcome error correlation within an observation. `summary()` describes the structural parameters for the fitted model.

## Formulas and missing data

In selection models, indicators and selection regressors are required on every retained row. Outcomes and outcome regressors are required only on selected rows; unused entries may be `NA`. With `selection = NULL`, outcomes are required on every retained row, and omitting missing outcomes does not correct for selection.

Use `.outcome` to copy outcome regressors into the selection formula; `observed ~ .outcome + z` adds an exclusion restriction that can aid identification. Ordinary `.` shorthand is unsupported. Formulas support factors, interactions, transformations and `offset()`.

`subset` is applied before missing-data checks. `na_action` handles missing required values: `"fail"` stops, `"omit"` drops rows, and `"exclude"` also restores `NA` positions in fitting-row predictions and residuals. See panelselect-design for design and factor-support details.

## Restrictions with `fixed`

`fixed` holds named parameters, including regression coefficients, at natural-scale values. Use the parameter names listed under Models.

- `fixed = c(cor_re = 0)` removes dependence between individual effects.

- `fixed = c(cor_error = 0)` removes same-observation error dependence and can substantially speed up estimation. Use `fixed = c(cor_re = 0, cor_error = 0)` to combine both restrictions when both correlations are present.
- Setting an SD to zero removes that random effect and automatically fixes any unidentified associated correlation to zero. In PLN models, `fixed = c(sd_error = 0)` removes observation-level heterogeneity, reducing the outcome to Poisson.

Correlations must remain strictly between -1 and 1; Linear `sd_error` must be positive. Fixed parameters remain in the estimates, have no inferential entries, and do not count toward AIC/BIC.

## Starting values

`init` supplies values not specified in `start`. The default uses separate random-effects fits for applicable panel selection equations; standalone and cross-sectional models use pooled starts. Failed, nonconvergent, or invalid auxiliary fits warn and retain pooled starts for the affected equation. Use `init = "pooled"` to bypass auxiliary fitting; see `init` for other choices.

Named partial starts override automatic values; a complete free-parameter vector bypasses initialization. Starts and fixed values use natural scales and must agree where both are supplied.

`update()` normally uses fitted coefficients as starts. To initialize afresh, use `update(fit, start = NULL, init = "pooled")`; changing `init` alone normally preserves fitted starts. See `update.panelselect()`.

## Inference

Use summary(), vcov() and confint() for standard errors and intervals. The default `vcov = "opg"` uses unit scores and avoids a Hessian calculation. `"observed"` uses observed information; `"sandwich"` combines it with unit scores. Both require an additional information calculation.

OPG and observed-information inference assume a correctly specified likelihood. Sandwich inference is asymptotic and requires independent units and consistent coefficient estimates; it does not correct coefficient bias or integration error. Independent units are panels, or rows for cross-sectional models.

`fit = add_information(fit)` stores observed information without refitting or changing the default covariance. Unavailable requested covariance is never replaced automatically. See panelselect-inference for formulas and interval construction, and `add_information()` for repeated requests.

## Integration and numerical checks

`panel_control()` describes GH, AGH and GL settings. GL orders count nodes per integration segment, so equal GH/AGH and GL orders are not directly comparable. PLN's observation-level rule is set separately by `observation_quadrature`; changing the person-effect rule leaves it unchanged.

Use `check_stationarity()` to inspect optimizer convergence and `Q_OPG`; the package warns when `Q_OPG > 0.1` by default. Stationarity does not establish integration accuracy. `check_quadrature()` compares likelihoods and scores with parameters fixed. Refit explicitly with `update()` or `refit = TRUE` to assess changes in estimates. Agreement supports numerical stability but does not guarantee accuracy or a global maximum.

GL supports population predictions, but history-conditioned predictions and `random_effects()` are unavailable when GL is used at either integration level. See `predict.panelselect()` for supported predictions.

## References

Bailey, M., & Peng, J. (2025). A Random Effects Model of Non-Ignorable Nonresponse in Panel Survey Data. Available at SSRN doi:10.2139/ssrn.5475626

Peng, J., & Van den Bulte, C. (2024). Participation vs. Effectiveness in Sponsored Tweet Campaigns: A Quality-Quantity Conundrum. *Management Science*, 70(11):7961-7983. Available at doi:10.1287/mnsc.2019.01897

## See Also

panelselect-methods, `predict.panelselect()`, `vignette("vignette", package = "PanelSelect")`

Other fitting functions: `panel_control()`

## Examples

```
# 1. Fit a panel selection model
set.seed(1)
dat = data.frame(person = rep(1:60, each = 4), x = rnorm(240), z = rnorm(240))
u = rnorm(60); v = 0.5 * u + rnorm(60, sd = 0.8)  # correlated person effects
dat$observed = as.integer(0.3 + 0.5 * dat$z + 0.7 * u[dat$person] + rnorm(240) > 0)
dat$y = ifelse(dat$observed == 1, 1 + 0.5 * dat$x + v[dat$person] + rnorm(240), NA)

# z enters the selection equation but is excluded from the outcome.
fit = panel_select(y ~ x, observed ~ .outcome + z, dat, id = "person",
                   family = "linear")
summary(fit)

# 2. Check and refine the fit
check_stationarity(fit)[c("optimizer_converged", "Q_OPG")]
# Compare calculations with more nodes, holding the estimates fixed.
check_quadrature(fit, nodes = 20)

# The initial fit uses default GH; refit with 20 nodes.
fit20 = update(fit, control = panel_control(nodes = 20))
cbind(GH10 = coef(fit), GH20 = coef(fit20))


# 3. Predict population outcome means
targets = data.frame(x = c(-1, 0, 1), z = 0)
# Mean potential outcomes before selection, averaging over person effects.
predict(fit, targets, type = "response",
        conditioning = "population", interval = "confidence")


# 4. Set both selection-outcome correlations to zero
independent = update(fit, fixed = c(cor_re = 0, cor_error = 0))
coef(independent)

# 5. Fit a cross-sectional selection model
# Use one observation per independent person.
cross_dat = dat[!duplicated(dat$person), ]
cross = panel_select(y ~ x, observed ~ .outcome + z, data = cross_dat,
```

```
                      id = NULL, family = "linear")
coef(cross)

# 6. Fit a standalone Poisson model with exposure
# Counts are observed for everyone, so there is no selection equation.
dat$exposure = runif(240, 0.5, 2)
dat$count = rpois(240, dat$exposure * exp(0.2 + 0.3 * dat$x + 0.4 * u[dat$person]))
counts = panel_select(count ~ x + offset(log(exposure)), selection = NULL,
                      data = dat, id = "person", family = "poisson")
coef(counts)
```

---

`panelselect-methods` *Work with fitted PanelSelect models*

---

## Description

Use the standard R generics below to inspect or reuse a `panel_select()` fit. Reported coefficients and covariance matrices use natural parameter scales.

## Details

| Task | Functions and help |
|---|---|
| Extract estimates and uncertainty | coef(), vcov(), confint() |
| Read a fit or coefficient table | print(), summary() |
| Inspect likelihood and compare models | logLik(), nobs(), AIC(), BIC() |
| Recover formulas, designs and omitted rows | formula(), terms(), model.frame(), model.matrix(), na.action() |
| Refit with changed settings or data | update() |
| Predict or summarize person effects | predict(), fitted(), residuals(), `random_effects()` |
| Check a fit | `check_stationarity()`, `check_quadrature()`, `add_information()` |

See `vignette("vignette", package = "PanelSelect")` for a complete analysis workflow and `vignette("migration", package = "PanelSelect")` for earlier APIs.

## See Also

`panel_select()`, `panel_control()`

`predict.panelselect` *Predict outcome means and selection probabilities*

## Description

Compute a clearly defined mean, probability or linear predictor at the fitted parameters. Population predictions average over person effects. History predictions instead update each person's effect distribution using explicitly supplied responses and selection indicators; they do not refit coefficients.

## Usage

```
## S3 method for class 'panelselect'
predict(
  object,
  newdata = NULL,
  type = c("response", "selection_probability", "joint", "selected_mean", "link"),
  equation = c("outcome", "selection"),
  conditioning = c("population", "history"),
  history = NULL,
  se.fit = FALSE,
  vcov = NULL,
  aggregate = FALSE,
  weights = NULL,
  interval = c("none", "confidence"),
  level = 0.95,
  ...
)

## S3 method for class 'panelselect'
fitted(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | A fitted `panelselect` model from `panel_select()`. |
| `newdata` | Data frame containing the regressors, factor levels and offset variables needed by the target equations; response columns may be omitted. `NULL` (default) uses the retained fitting rows. Under `na_action = "exclude"`, default row predictions restore excluded rows as `NA`. History conditioning also requires the fitted ID column with nonmissing target IDs. New factor levels unsupported by the fitted design cause an error. |
| `type` | One target: `"response"` (default potential-outcome mean), `"selection_probability"`, `"joint"`, `"selected_mean"`, or `"link"`. The table below distinguishes quantities that can differ substantially in selection models. Standalone models support `"response"` and `"link"` only. |
| `equation` | Equation for `type = "link"`: `"outcome"` (default) or `"selection"`. Other target types already determine the equations they need; setting `equation = "selection"` for them is an error. |

| | |
|---|---|
| conditioning | "population" (default) uses the prior person-effect distribution. "history" uses the posterior given the matching person's supplied history. History requires a panel model with GH/AGH integration; it is unavailable for GL, including PLN observation rules, and for links. |
| history | Data frame of conditioning rows for conditioning = "history". Supply the fitted ID column, selection indicator and selection regressors on all relevant rows, plus observed outcomes/outcome regressors on selected rows (all rows for standalone models). Include offset variables required by the formulas. NULL is an error for history prediction; no responses are taken implicitly from the target or training data. An empty history gives population predictions. With population conditioning, a supplied history is rejected rather than ignored. |
| se.fit | Logical; TRUE returns parameter-estimation standard errors and natural-parameter gradients along with predictions. Default FALSE returns just predictions or the requested confidence-interval matrix. |
| vcov | Covariance for SEs/intervals: NULL (default) uses the fit's choice; choose "opg", "observed", "sandwich", or supply a named natural-scale covariance matrix. An observed/sandwich request may calculate information. This argument is used only when SEs or confidence intervals are requested. |
| aggregate | Logical; default FALSE returns one prediction per row. TRUE averages the row predictions and their gradients, giving a single mean/probability and, if requested, its uncertainty. |
| weights | Numeric averaging weights, one per row of newdata, used only with aggregate = TRUE. They must be finite, nonnegative and sum to a positive number; they are normalized internally. NULL gives equal row weights. These are prediction-averaging weights, not likelihood weights. |
| interval | "none" (default) or "confidence" for confidence bounds on the estimated target. Bounds include parameter uncertainty only; "prediction" intervals for realized outcomes are not implemented. |
| level | Confidence level strictly between zero and one when interval = "confidence"; default 0.95. |
| ... | predict() rejects unused arguments. fitted() forwards its arguments to predict(), so the same targets and options are available. |

## Value

Normally a named numeric vector; with aggregate = TRUE, a single estimate. interval = "confidence" returns a matrix with fit, lwr and upr. With se.fit = TRUE, the result is a list containing that fit vector/matrix, a numeric se.fit, and a gradient matrix with natural parameters in columns. Fixed-parameter gradient columns do not add variance.

## Choose the target

Write D for selection and Y* for the potential outcome before selection. Means below condition on the supplied regressors and any explicit history.

| type | Quantity | Use |
|---|---|---|

| | | |
|---|---|---|
| "response" | E[Y*] | Mean outcome if every target row could be observed |
| "selection_probability" | P(D = 1) | Probability that the outcome is observed or the perso |
| "joint" | E[D * Y*] | Mean observed contribution when nonparticipation c |
| "selected_mean" | ‘E[Y* | D = 1]‘ |
| "link" | Fixed linear predictor including offsets | Inspect the chosen equation’s regression index, witho |

Binary-outcome response means are probabilities. Linear-outcome means use outcome units, while Poisson/PLN means are expected counts, including exposure offsets when specified. Each target represents a fresh within-person shock.

### Choose the conditioning information

- **Population:** integrate over the model’s person-effect distribution. This is appropriate for a new person or a population-average target. It is available after GH, AGH and GL fitting and uses analytic target formulas.
- **History:** integrate over each person’s posterior given exactly the rows supplied in `history`. IDs absent from usable history receive population predictions. The current GH/AGH history calculation uses the fit’s stored integration settings; there is no prediction-specific method override.

The function does not infer chronology, past-only windows or leave-one-out samples. For forecasting, supply only information available before the target. Including a target’s own observed response informs its person effects and produces a fresh-wave replicated prediction, not a held-out forecast or a prediction of that same realized shock. An absent history row is not an observed `D = 0` event. Missing required history values follow the fitted `na_action`: `"fail"` stops, otherwise incomplete history rows are omitted.

For selected means the target selection event also reweights the history posterior. The result is a ratio of integrated moments, not an unweighted average of node-specific selected means.

### Uncertainty and aggregation

- SEs use the delta method with uncertainty in the fitted parameters. For history targets, derivatives include posterior weights and moving nodes.
- Confidence bounds use a probit scale for probabilities, a log scale for count means, and the identity scale for Linear means and linear predictors, then transform back. Aggregation occurs before this interval calculation.
- A supplied matrix must name all natural parameters, be finite and symmetric, and be positive semidefinite (singular matrices are allowed). Symmetry and eigenvalue checks allow relative roundoff of `1e-10`.
- Unavailable covariance produces unavailable SEs; a different estimator is never substituted. These intervals do not include realized-outcome variation.

### Fitted values versus residuals

`fitted()` defaults to the potential-outcome mean, whereas `residuals()` uses the selected mean by default for selection models. Therefore adding their default results does not generally reconstruct observed selected outcomes.

## See Also

random_effects(), residuals.panelselect(), panelselect-inference

Other prediction functions: random_effects(), residuals.panelselect()

## Examples

```
set.seed(1)
dat = data.frame(person = rep(1:60, each = 4), x = rnorm(240), z = rnorm(240))
u = rnorm(60); v = 0.5 * u + rnorm(60, sd = 0.8)  # correlated person effects
dat$observed = as.integer(0.3 + 0.5 * dat$z + 0.7 * u[dat$person] + rnorm(240) > 0)
dat$y = ifelse(dat$observed == 1, 1 + 0.5 * dat$x + v[dat$person] + rnorm(240), NA)
fit = panel_select(y ~ x, observed ~ x + z, dat, "person", "linear")

targets = data.frame(person = 1:2, x = 0, z = 0)
predict(fit, targets, type = "selection_probability", interval = "confidence")
# Condition on each person's observed rows.
predict(fit, targets, type = "selected_mean", conditioning = "history",
        history = dat[dat$person %in% 1:2, ])
predict(fit, aggregate = TRUE, se.fit = TRUE)$fit
```

---

# random_effects *Summarize person effects conditional on observed history*

---

## Description

Calculate posterior means and SDs of person-specific random-intercept contributions at the fitted parameters. These summaries describe conditional heterogeneity in each equation, rather than re-estimating its coefficients.

## Usage

```
random_effects(object, history = NULL)
```

## Arguments

| | |
|---|---|
| object | A fitted panel panelselect model using GH/AGH integration. Cross-sectional models have no person effects. GL at the person-effect or PLN observation level is not supported by this function. |
| history | Data frame of conditioning rows, including the fitted ID and the response, selection, regressor and offset columns required by the original formulas. Required outcome data apply only to selected rows in selection models. NULL (default) uses the retained fitting rows, unlike predict() history conditioning, which always requires explicit history. Supply a restricted history for an earlier information set. Missing required values stop under na_action = "fail" and are omitted otherwise. |

## Value

A data frame with one row per history person and the columns above. A structurally fixed-zero effect has zero posterior mean and SD.

## Interpretation and available summaries

| Column | Meaning |
|---|---|
| Fitted ID-column name | Person whose supplied usable rows define the posterior. |
| `selection_mean`, `selection_sd` | Posterior mean and SD of the selection random intercept; included only for selection mod |
| `outcome_mean`, `outcome_sd` | Posterior mean and SD of the outcome random intercept. |

The intercepts include their fitted person-effect SD multipliers. For count models they contribute to the log rate, not directly to the expected count. The posterior SDs measure remaining uncertainty about a person's effect at fixed fitted parameters; they are not parameter-estimation SEs or confidence intervals for regression coefficients.

Exactly the supplied history is used, without automatic date filtering. Default summaries use the person's own fitting responses and are descriptive. Only IDs represented in usable history are returned; an empty history returns an empty data frame with the appropriate columns. For predictions of people without observed history, use `predict()` and its population fallback.

## See Also

`predict.panelselect()`, `panel_select()`

Other prediction functions: `predict.panelselect()`, `residuals.panelselect()`

## Examples

```
set.seed(1)
dat = data.frame(person = rep(1:60, each = 4), x = rnorm(240), z = rnorm(240))
u = rnorm(60); v = 0.5 * u + rnorm(60, sd = 0.8)  # correlated person effects
dat$observed = as.integer(0.3 + 0.5 * dat$z + 0.7 * u[dat$person] + rnorm(240) > 0)
dat$y = ifelse(dat$observed == 1, 1 + 0.5 * dat$x + v[dat$person] + rnorm(240), NA)
fit = panel_select(y ~ x, observed ~ x + z, dat, "person", "linear")
head(random_effects(fit))
```

---

# `residuals.panelselect` *Calculate response residuals for observed outcomes*

---

## Description

Subtract a predicted mean from each observed outcome on the fitting sample. The default target is the selected mean in a selection model and the response mean in a standalone model. Unselected outcomes have no residual.

## Usage

```
## S3 method for class 'panelselect'
residuals(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | A fitted `panelselect` model. Residuals use its retained fitting rows; this method is not a new-data prediction interface. |
| `...` | Options forwarded to `predict.panelselect()`. `type` selects the mean to subtract, not a residual standardization: default `"selected_mean"` for selection models and `"response"` otherwise; the other supported prediction targets retain their usual meanings. `conditioning = "history"` requires explicit `history` and supported GH/AGH rules. Requests for new data, aggregated output, SE output or interval matrices are not supported. There are no Pearson or deviance residuals. |

## Value

A numeric vector in fitting-row order, with `NA` for unselected rows. Under `na_action = "exclude"`, excluded fitting rows are also restored as `NA`; `"omit"` does not restore dropped rows.

## Common choices

- `residuals(fit)` uses the population mean appropriate to the observed outcomes: selected mean for selection models, response mean otherwise.
- `residuals(fit, type = "response")` uses the potential-outcome mean; selection can make these residuals systematically nonzero among participants.
- `residuals(fit, conditioning = "history", history = dat)` uses person effects informed by `dat`. If it includes each row's own outcome, these are descriptive residuals and should not be interpreted as forecast errors.

`fitted()` has a different default target in selection models. To reconstruct selected outcomes, use `predict(fit, type = "selected_mean") + residuals(fit)`.

## See Also

`predict.panelselect()`, `random_effects()`

Other prediction functions: `predict.panelselect()`, `random_effects()`

## Examples

```
set.seed(1)
dat = data.frame(person = rep(1:60, each = 4), x = rnorm(240), z = rnorm(240))
u = rnorm(60); v = 0.5 * u + rnorm(60, sd = 0.8)  # correlated person effects
dat$observed = as.integer(0.3 + 0.5 * dat$z + 0.7 * u[dat$person] + rnorm(240) > 0)
dat$y = ifelse(dat$observed == 1, 1 + 0.5 * dat$x + v[dat$person] + rnorm(240), NA)
fit = panel_select(y ~ x, observed ~ x + z, dat, "person", "linear")
summary(residuals(fit))
# Residuals from each person's posterior mean given their own rows.
head(residuals(fit, conditioning = "history", history = dat))
```

summary.panelselect *Summarize and print a fitted model*

## Description

Display the model specification, retained sample, estimates, uncertainty and numerical diagnostics. `print(fit)` gives a compact coefficient view; `summary(fit)` constructs the full coefficient table and its printed summary.

## Usage

```
## S3 method for class 'panelselect'
summary(object, vcov = object$control$vcov, level = 0.95, ...)

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

## S3 method for class 'summary.panelselect'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

## Arguments

| | |
|---|---|
| `object` | A fitted `panelselect` model to summarize. |
| `vcov` | Covariance used by the summary: `"opg"`, `"observed"`, `"sandwich"`, or a named natural-scale covariance matrix. Defaults to the choice stored in the fit. Observed/sandwich requests calculate information if needed and reuse that calculation for `Q_Hessian`; the input fit is not modified. Assign `add_information()` first to retain it for later requests. |
| `level` | Confidence level between zero and one for returned coefficient intervals; default `0.95`. Printed coefficient tables show estimates, SEs, z statistics and p-values; interval columns remain in the summary object. |
| `...` | `summary()` rejects unused arguments. Print methods ignore them; select covariance in `summary()`, not in `print()`. |
| `x` | A fitted model for `print.panelselect()`, or the result of `summary()` for `print.summary.panelselect(` |
| `digits` | Significant digits for printing; default is at least three, otherwise `getOption("digits") - 3`. |
| `signif.stars` | Whether to print significance stars; defaults to `getOption("show.signif.stars")`. |

## Value

summary() returns a summary.panelselect list. coefficients contains estimate, se, z, p, lci, and uci; stationarity contains the OPG status/threshold/flag and the two named Q values. Other components describe the model, likelihood criteria, covariance and numerical settings. exclusion_restriction is a character vector (character(0) when none, NULL for standalone models). Print methods return their input invisibly.

## What to inspect

| Component | Interpretation |
|---|---|
| Specification and sample | Outcome/selection formulas, panel versus cross-sectional structure, retained rows/people and or |
| Exclusion restriction | Raw selection variables absent from the active outcome equation; not a statistical test of validity |
| Coefficient table | Natural-scale estimates and SEs; correlations use Fisher-scale tests, SDs have no ordinary zero |
| Structural-parameter legend | Meanings of the SD and correlation parameters for this model. |
| Covariance | Requested estimator and its availability; it is never silently replaced. See panelselect-inference. |
| Integration | The fitted rules/orders and available integration diagnostics. See check_quadrature(). |
| Optimizer and Q | Backend stopping status, Q_OPG, and Q_Hessian when valid information exists. These assess di |

Q_OPG > stationarity_warning is flagged as substantial (0.1 by default). There is no automatic warning cutoff for Q_Hessian. Neither a small Q nor optimizer convergence proves integration accuracy, identification or a global maximum. Saved native fits are displayed under the current Q convention using stored scores/covariances; inspecting them does not refit or overwrite them.

## See Also

panelselect-methods, check_stationarity(), panelselect-inference

## Examples

```
dat = data.frame(person = rep(1:20, each = 3), y = rep(c(0, 1, 3, 1, 2, 4), 10))
fit = panel_select(y ~ 1, NULL, dat, "person", "poisson",
                   fixed = c(sd_re_outcome = 0))
print(fit)
answer = summary(fit, vcov = "observed", level = 0.9)
print(answer, digits = 4, signif.stars = FALSE)
answer$stationarity
```

---

# update.panelselect *Refit a model with changed formulas, data or settings*

---

## Description

Refit a panel_select() model using retained data and fitted coefficients as warm starts. Settings and restrictions persist unless explicitly overridden. Use start = NULL to run an initializer afresh instead of warm-starting.

## Usage

```
## S3 method for class 'panelselect'
update(object, formula., ..., evaluate = TRUE)
```

## Arguments

| | |
|---|---|
| `object` | A fitted `panelselect` model. Its retained data and settings supply defaults, so the original data-frame variable need not still exist. |
| `formula.` | Optional updated outcome formula following `stats::update.formula()`, for example `. ~ . + z` or `. ~ . - x`. Omit to keep the fitted outcome formula. |
| `...` | Named overrides of `panel_select()` arguments: `outcome`, `selection`, `data`, `id`, `family`, `start`, `fixed`, `control`, `subset`, `na_action`, `init`, or `vcov`. Values are evaluated in the calling environment; `subset` is evaluated in the data. See the options below. |
| `evaluate` | Logical; default `TRUE` fits the updated model. `FALSE` returns a self-contained fitting call after argument checks and design preparation; rank, identification and initialization checks run when the returned call is evaluated. This constructs the call without running initialization or optimization. The returned call can embed data. |

## Value

A newly fitted `panelselect` model, or a fitting call when `evaluate = FALSE`. The supplied fit is unchanged.

## Common updates

| Change | How to request it |
|---|---|
| Add/remove outcome terms | `update(fit, . ~ . + z)`; a `.outcome` selection shorthand follows the updated outcome. |
| Change selection terms | `update(fit, selection = . ~ . + z)` or supply a complete selection formula. |
| Override some warm starts | `start = c(cor_re = 0.2)`; other surviving coefficients remain warm starts. |
| Try a fresh initializer | `start = NULL, init = "pooled"` (or another supported `init`); changing `init` alone norm |
| Change restrictions | Supply a complete replacement `fixed` vector; `fixed = numeric()` removes explicit restrictio |
| Change one numerical control | `control = modifyList(fit$control, list(maxit = 1000))` preserves the remaining cont |
| Replace all numerical controls | `control = panel_control(...)` uses current defaults for omitted controls, including GH fo |
| Change default inference | `vcov = "observed"`, `"opg"`, or `"sandwich"`; this is a fitting argument, not a control-list optio |
| Use new data | Supply `data`; if the fit used `subset`, also supply its new value or `subset = NULL`. |

## Starting values and changed models

Explicit starts must be finite named numeric vectors with unique, valid parameter names; matrices, unnamed vectors and duplicate names are rejected before they can overwrite warm starts. Named partial starts remain supported. Parameters absent from a changed design/family leave inherited starts and restrictions. Newly freed SDs previously fixed at zero get automatic starts. An empty start

vector adds no overrides; `start = NULL` requests automatic starts for all free parameters. Explicit start/fixed conflicts and nonpositive free SD starts are rejected when the fitting call is evaluated.

The stored `na_action`, `init`, `vcov`, integration rule and node orders persist unless overridden. Staged meshes are rebuilt for each update. No method is silently changed to improve convergence or integration accuracy.

## Formula shorthand and saved fits

In a dotted selection update, `.outcome` is one parenthesized term, so `. ~ . - x` cannot remove just one regressor inside that shorthand; write the selection formula in full for that change. Earlier native 2.0 fits are read using current names, but removed integration/optimizer settings stop with guidance for an explicit replacement. See the migration vignette. The former initializer string `"ProbitRE"` can remain in a displayed saved fit, but it is not a supported update/refit setting. Supply `init = "default"` (or another current initializer) explicitly, even with `evaluate = FALSE`. `update(fit, init = "default")` keeps fitted warm starts; `update(fit, init = "default", start = NULL)` reruns the automatic initialization described in `panel_select()`. Earlier person-effect parameter names in a saved fit are read as `sd_re_selection`, `sd_re_outcome` and `cor_re`; explicit new `start` and `fixed` inputs must use these current names. The `id` data-column argument is unchanged. Previously stored Gaussian information is not repaired merely by printing or calling `add_information()` again; update/refit explicitly for the corrected outcome-scale differentiation steps.

## See Also

`panel_select()`, `panel_control()`, panelselect-methods

## Examples

```
dat = data.frame(person = rep(1:20, each = 3), y = rep(c(0, 1, 3, 1, 2, 4), 10))
fit = panel_select(y ~ 1, NULL, dat, "person", "poisson",
                   fixed = c(sd_re_outcome = 0))
observed = update(fit, vcov = "observed")
fresh = update(fit, start = NULL, init = "pooled")
call = update(fit, control = modifyList(fit$control, list(maxit = 1000)),
              evaluate = FALSE)
coef(fresh)
```

# Index


∗ **diagnostic functions**
  add_information, 3
  check_quadrature, 5
  check_stationarity, 7
∗ **fitting functions**
  panel_control, 14
  panel_select, 19
∗ **prediction functions**
  predict.panelselect, 26
  random_effects, 29
  residuals.panelselect, 30

add_information, 3
add_information(), *3*, *7*, *9*, *11*, *23*, *25*, *32*, *35*
AIC(), *25*
AIC.panelselect *(*logLik.panelselect*)*, 13

BIC(), *25*
BIC.panelselect *(*logLik.panelselect*)*, 13

check_quadrature, 5
check_quadrature(), *2*, *4*, *9*, *14*, *17*, *19*, *22*, *23*, *25*, *33*
check_stationarity, 7
check_stationarity(), *2*, *4*, *7*, *11*, *16*, *21*, *23*, *25*, *33*
coef(), *21*, *25*
coef.panelselect, 9
confint(), *4*, *23*, *25*
confint.panelselect *(*coef.panelselect*)*, 9

fitted(), *25*
fitted.panelselect *(*predict.panelselect*)*, 26
formula(), *25*
formula.panelselect, 11

logLik(), *21*, *25*
logLik.panelselect, 13

model.frame(), *25*
model.frame.panelselect *(*formula.panelselect*)*, 11
model.matrix(), *25*
model.matrix.panelselect *(*formula.panelselect*)*, 11

na.action(), *25*
na.action.panelselect *(*formula.panelselect*)*, 11
nobs(), *25*
nobs.panelselect *(*logLik.panelselect*)*, 13

panel_control, 14
panel_control(), *2*, *6*, *7*, *9*, *21*, *23–25*, *35*
panel_select, 19
panel_select(), *2*, *3*, *9*, *12*, *14*, *19*, *25*, *26*, *30*, *33–35*
PanelSelect *(*PanelSelect-package*)*, 2
panelselect-comparison *(*logLik.panelselect*)*, 13
panelselect-design, *22*
panelselect-design *(*formula.panelselect*)*, 11
panelselect-inference, *23*, *29*, *33*
panelselect-inference *(*coef.panelselect*)*, 9
panelselect-methods, *2*, *4*, *22*, *24*, 25, *33*, *35*
PanelSelect-package, 2
predict(), *10*, *25*, *29*, *30*
predict.panelselect, 26
predict.panelselect(), *2*, *10*, *12*, *23*, *24*, *30*, *31*
print(), *25*
print.panelselect *(*summary.panelselect*)*, 32
print.panelselect_control *(*panel_control*)*, 14

print.panelselect_quadrature_check
(check_quadrature), 5
print.summary.panelselect
(summary.panelselect), 32

random_effects, 29
random_effects(), 2, 3, 17, 23, 25, 29, 31
residuals(), 25, 28
residuals.panelselect, 30
residuals.panelselect(), 2, 29, 30

stats::update.formula(), 34
summary(), 4, 10, 21, 23, 25
summary.panelselect, 32
summary.panelselect(), 11

terms(), 25
terms.panelselect
(formula.panelselect), 11

update(), 6, 22, 25
update.panelselect, 33
update.panelselect(), 7, 12, 14, 23

vcov(), 4, 21, 23, 25
vcov.panelselect (coef.panelselect), 9