Package: picreg (via r-universe)

October 20, 2026

Type Package

Title Variable Selection using the Pivotal Information Criterion

Version 0.2.0

Date 2026-10-20

Description Sparse regression and classification via the Pivotal Information Criterion (PIC), an alternative to the Bayesian Information Criterion (BIC), cross-validation, and Lasso-based tuning. The regularization parameter is selected from a pivotal null-distribution statistic, eliminating the need for cross-validation and yielding sharper support recovery. Provides Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) optimization for the L1, Smoothly Clipped Absolute Deviation (SCAD), and Minimax Concave Penalty (MCP) penalties across six response distributions: Gaussian, binomial, Poisson, exponential, Gumbel, and Cox. Under standard sparsity assumptions, the selector achieves a phase transition for exact support recovery, analogous to results in compressed sensing. See Sardy, van Cutsem and van de Geer (2026) <doi:10.48550/arXiv.2603.04172>.

License GPL-2

URL https://github.com/VcMaxouuu/picreg, https://vcmaxouuu.github.io/picreg/

BugReports https://github.com/VcMaxouuu/picreg/issues

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

Imports stats, graphics, grDevices, Matrix, Rcpp (>= 1.0.10)

LinkingTo Rcpp, RcppArmadillo

Suggests testthat (>= 3.0.0), knitr, rmarkdown, xfun, glmnet, future, future.apply

VignetteBuilder knitr

SystemRequirements C++17

RoxygenNote 8.0.0

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

NeedsCompilation yes

Author Maxime van Cutsem [aut, cre], Sylvain Sardy [aut]

Maintainer Maxime van Cutsem <maxime.vancutsem@unige.ch>

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

Date/Publication 2026-10-20 12:20:02 UTC

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

RemoteRef HEAD

RemoteSha 105e8cfa7b774e9e4d1788db9982e16eb9f9bf08

Contents

assess . . . 3
baseline_functions . . . 4
BinomialExample . . . 4
coef.pic . . . 5
concordance_index . . . 6
cox_partial_log_likelihood . . . 7
CoxExample . . . 7
feature_effects_on_survival . . . 8
lambda_pdb . . . 9
pdb_asymptotic . . . 12
pdb_summary . . . 13
phase_transition . . . 14
pic . . . 16
pic_families . . . 20
pic_methods . . . 20
pic_penalties . . . 21
pic_plots . . . 22
pic_survival . . . 22
plot.pic . . . 22
plot.pic.lambda_pdb . . . 23
plot.pic.pdb_asymptotic . . . 23
plot.pic.phase_transition . . . 24
plot_baseline . . . 25
plot_survival_curves . . . 25
predict.pic . . . 26
print.pic.family . . . 27
print.pic.pdb_asymptotic . . . 27
print.pic.phase_transition . . . 28
print.summary.pic . . . 28

QuickStartExample 29
summary.pic 30

Index **31**

`assess` *Assess performance of a* `pic` *fit.*

Description

Given a test set, reports a small set of family-appropriate predictive metrics. Optionally appends support-recovery diagnostics when the true active set is known.

Usage

```
assess(object, ...)

## S3 method for class 'pic'
assess(object, newx, newy, true_features = NULL, ...)
```

Arguments

`object`	A fitted `pic` object.
`...`	Unused; present for S3 method consistency.
`newx`	Numeric design matrix at which predictions are evaluated, with the same columns as the training data.
`newy`	Response on the new observations. Numeric vector for all families except Cox (logical or factor values are accepted for binomial fits and mapped to `0`/`1` as at training); for Cox, a two-column matrix `(time, event)` matching the training response format.
`true_features`	Optional integer or character vector listing the indices (or names) of the true active variables. When supplied, support-recovery metrics are appended.

Details

The metrics depend on the family:

`"gaussian"` MSE, MAE, R-squared.

`"binomial"` accuracy, AUC, binomial deviance.

`"poisson"` MSE, MAE, Poisson deviance.

`"exponential"` MSE, MAE, Exponential deviance.

`"gumbel"` MSE, MAE, deviance computed from the per-sample negative log-likelihood with the scale parameter profiled at its maximum-likelihood estimate.

`"cox"` Harrell's C-index and the Breslow partial log-likelihood (negative, normalized by `n`).

When `true_features` is non-`NULL`, four support-recovery metrics are appended (independent of `newx` / `newy`): `exact_recovery`, `tpr` (true-positive rate / sensitivity), `fdr` (false-discovery rate) and `f1` (harmonic mean of precision and recall). Names are accepted in addition to integer positions when the fit carries column names.

Value

A two-column `data.frame` with columns `metric` (character) and `value` (numeric).

Examples

```
data(QuickStartExample)
X <- QuickStartExample$X
y <- QuickStartExample$y
fit <- pic(X, y, family = "gaussian", penalty = "scad")
assess(fit, X, y, true_features = paste0("gene_", 1:5))
```

---

baseline_functions *Breslow baseline cumulative hazard and survival.*

Description

Breslow baseline cumulative hazard and survival.

Usage

```
baseline_functions(times, events, predictions)
```

Arguments

`times`	Survival times (length `n`).
`events`	Event indicators (0 or 1; length `n`).
`predictions`	Risk scores; higher = higher risk = shorter survival.

Value

A data.frame with columns `time`, `cumulative_hazard`, `survival`.

BinomialExample *Small binary-classification dataset for the Binomial section of the vignette.*

Description

A synthetic logistic-regression dataset used to illustrate `pic()` with the Binomial family. It contains $n = 300$ observations of $p = 50$ predictors, of which $s = 5$ carry signal; the remaining 45 are noise.

Usage

```
BinomialExample
```

Format

A list with two components:

`X` Numeric matrix of dimension 300×50 with column names `gene_*` and `noise_*`.

`y` Integer vector of length 300 containing $0/1$ class labels.

Details

Column names follow the same convention as `QuickStartExample`: active variables are labeled `gene_1, ..., gene_5` and inactive ones `noise_1, ...,noise_45`, interleaved in random order.

The binary response is generated as

$$Y_i \sim \text{Bernoulli}\left(\frac{1}{1+e^{-X_i\beta}}\right),$$

with non-zero coefficients drawn uniformly in $[1.5, 3]$ with random sign. The class balance is roughly 45% of positives.

Examples

```
data(BinomialExample)
fit <- pic(BinomialExample$X, BinomialExample$y, family = "binomial")
fit$selected
```

`coef.pic` *Coefficients of a fitted pic model.*

Description

Returns the fitted coefficients as a one-column **sparse matrix** of class `"dgCMatrix"` (from the **Matrix** package), mirroring the output of `coef()` for **glmnet**. The first row is the intercept, labeled `"(Intercept)"`; the remaining rows are the predictors. Cox fits have no intercept row, the Cox model being invariant to a shift of the linear predictor. Row names are taken from the column names of `X` when available (matrix `colnames(X)` or data-frame column names) and otherwise default to `V1, ..., Vp`. Zero coefficients are stored implicitly and printed as `"."`, which keeps the display compact in high dimensions.

Usage

```
## S3 method for class 'pic'
coef(object, standardized = FALSE, ...)
```

Arguments

`object`	A fitted `pic` object.
`standardized`	Logical; if `TRUE`, return the coefficients on the standardized scale used internally during fitting. Default `FALSE` (return coefficients on the original scale of `X`).
`...`	Unused; present for S3 method consistency.

Details

Internally the model is fitted on a standardized design matrix, so the raw coefficients live on the standardized scale. By default this method rescales them back to the **original scale** of `X` — the values to plug into the un-standardized design for prediction — via

$$beta_orig = beta/s \quad intercept_orig = intercept - sum(m * beta_orig)$$

where `m` and `s` are the column mean and standard deviation. The intercept absorbs the centering even for fits with `intercept = FALSE`, so that the coefficients reproduce `predict.pic()`. Pass `standardized = TRUE` to skip the rescaling and return the raw fit values (identical to `fit$beta`, with intercept `0` when none was fitted).

Value

A sparse `(p + 1)` by `1` matrix of class `"dgCMatrix"`, with row names `c("(Intercept)", <variables>)` and column name `"coefficient"` (`p` by `1`, without the intercept row, for Cox fits). Use `as.numeric()` to obtain a plain numeric vector, or `which(coef(fit) != 0)` to list the selected entries.

`concordance_index` *Harrell's concordance index.*

Description

A pair (i, j) is comparable when i has an event and either $t_i < t_j$, or $t_i = t_j$ and j is censored. It is concordant when i has the higher risk score; tied scores count $1/2$. Computed in $O(n \log n)$.

Usage

```
concordance_index(times, events, predictions)
```

Arguments

`times`	Survival times (length `n`).
`events`	Event indicators (0 or 1; length `n`).
`predictions`	Risk scores; higher = higher risk = shorter survival.

Value

Numeric scalar in `[0, 1]` (`0` when no pair is comparable).

cox_partial_log_likelihood *Breslow partial log-likelihood (negative, normalized by* n*).*

Description

Breslow partial log-likelihood (negative, normalized by n).

Usage

```
cox_partial_log_likelihood(times, events, predictions)
```

Arguments

times	Survival times (length n).
events	Event indicators (0 or 1; length n).
predictions	Risk scores; higher = higher risk = shorter survival.

Value

A numeric scalar: the negative Breslow partial log-likelihood for the Cox proportional-hazards model, normalized by the sample size n. Lower values indicate a better fit.

CoxExample *Small survival dataset for the Cox section of the vignette.*

Description

A synthetic survival dataset used to illustrate pic() with the Cox family. It contains $n = 250$ subjects observed on $p = 50$ covariates, of which $s = 5$ carry signal; the remaining 45 are noise.

Usage

```
CoxExample
```

Format

A list with two components:

X Numeric matrix of dimension 250×50 with column names gene_* and noise_*.

y Numeric matrix of dimension 250×2 with columns time and event.

Details

Column names follow the same convention as QuickStartExample: active variables are labeled gene_1, ..., gene_5 and inactive ones noise_1, ...,noise_45, interleaved in random order.

Event times are drawn from an exponential proportional-hazards model

$$T_i \sim \mathrm{Exp}\big(e^{X_i\beta}\big),$$

and independent censoring times from $C_i \sim \mathrm{Exp}(0.5)$. The observed response is the standard two-column $(\min(T_i, C_i),\ \mathbf{1}\{T_i \le C_i\})$. The censoring rate is roughly 40%.

Examples

```
data(CoxExample)
fit <- pic(CoxExample$X, CoxExample$y, family = "cox")
fit$selected
```

---

feature_effects_on_survival

Effect of one feature on the Cox survival curve.

Description

Builds the family of survival curves obtained by varying a single covariate while holding the others at their column means. The returned object has the same shape as predict.pic() with type = "survival", so it composes directly with plot_survival_curves().

Usage

```
feature_effects_on_survival(object, idx, values = NULL)
```

Arguments

object	A fitted pic.cox object.
idx	Index of the feature to vary. Either an integer column position in the training X or, if X carried column names, the variable name. The feature must lie in the model's selected support; otherwise the curve would be flat in v and the call is rejected.
values	Optional numeric vector of values to evaluate. When NULL (default), the cached grid described in *Details* is used.

Details

Both the per-column mean row and a default grid of representative values (used when values = NULL) are cached on the fit by pic() under attr(fit, "preproc"), so the training design matrix does not need to be passed back in. The cached default grid uses the unique values of the column when there are at most five of them (handy for ordinal / categorical covariates), and the four equispaced empirical quantiles (0, 1/3, 2/3, 1) otherwise.

Value

A list with components `time` (length `K`) and `survival` (matrix `K x length(values)`, one column per evaluated value). Column names of `survival` are formatted as `"<feature_name> = <value>"` and are picked up automatically by `plot_survival_curves()` for the legend.

See Also

`predict.pic()`, `plot_survival_curves()`.

---

lambda_pdb *Pivotal Detection Boundary regularization selector*

Description

Computes the data-driven regularization parameter $\hat{\lambda}_{\alpha}^{\mathrm{PDB}}$ using the Pivotal Detection Boundary (PDB) principle. The selected value is defined as the empirical $(1-\alpha)$ quantile of a null-distribution gradient statistic

$$\hat{\lambda}_{\alpha}^{\mathrm{PDB}} = q_{1-\alpha}\left(\|\nabla \ell_0\|_{\infty}\right),$$

where ℓ_0 denotes the loss evaluated under the null model.

Usage

```
lambda_pdb(
  X,
  family,
  n_simu = 2000L,
  alpha = 0.05,
  method = c("mc_exact", "mc_gaussian", "analytical"),
  seed = NULL
)
```

Arguments

`X`	Numeric design matrix. Columns should typically be standardized to zero mean and unit variance.
`family`	A PIC family specification. Can be either a family object or a character string accepted by `get_family()`.
`n_simu`	Number of Monte Carlo simulations used by the stochastic methods. Ignored when `method = "analytical"`. Default 2000.
`alpha`	Nominal tail probability used to define the quantile level.
`method`	One of `"mc_exact"`, `"mc_gaussian"`, or `"analytical"`.
`seed`	Optional seed for the Monte Carlo draws, for a reproducible $\hat{\lambda}$. The caller's random stream is restored afterwards. Default `NULL` (no seeding).

Details

Under the null $\beta = 0$, the gradient of the loss carries only sampling noise. The smallest λ large enough to dominate this noise is the natural threshold separating signal from noise, i.e. the value above which a coefficient should be kept rather than shrunk to zero. Calibrating $\hat{\lambda}$ this way has two consequences. First, the quantile depends only on the design matrix X, the family, and the level α (not on the response y), so cross-validation is no longer needed. Second, and more importantly, it leads to sharper support recovery than prediction-error-based selectors such as cross-validated lasso: by targeting the noise level of the gradient directly, PDB controls the inclusion of noise variables and more reliably identifies the true non-zero coefficients. Computing the quantile requires only the distribution of $\|\nabla \ell_0\|_\infty$, which `lambda_pdb()` estimates through one of the three methods below.

Details on `method` option:

The empirical quantile is obtained using one of the three following methods:

`"mc_exact"` Family-aware Monte Carlo. For each of the `n_simu` draws, a response vector is sampled under the null model ($\beta = 0$) of the chosen family; the family-specific gradient of the loss at $\beta = 0$ is evaluated on the fixed design X, and its supremum norm is recorded. For Gumbel, both null location and scale are estimated jointly by maximum likelihood for every draw. The empirical $(1 - \alpha)$ quantile of the `n_simu` recorded norms is returned. Most accurate but slowest of the three.

`"mc_gaussian"` Monte Carlo under the Gaussian approximation of the null gradient. A central-limit argument gives $\nabla \ell_0 \approx \mathcal{N}(0,\, c(n)\, \Sigma_X / n)$ with $\Sigma_X = X^\top X / n$. Each of the `n_simu` draws samples directly from this Gaussian and records its supremum norm — no family-specific evaluation needed. Family-agnostic and noticeably faster than `"mc_exact"`; valid in the regime where the CLT kicks in (moderate to large n).

`"analytical"` Closed-form Bonferroni bound on the Gaussian tail. Combining a union bound over the p coordinates of the gradient with the standard Gaussian tail bound gives

$$\hat{\lambda}_\alpha^{\text{analytical}} = \Phi^{-1}(1 - \alpha/(2p))\ \sqrt{c(n)/n}.$$

Deterministic and $O(1)$ — no simulation. Conservative (over-estimates the true quantile): nearly exact when the columns of X are weakly correlated, increasingly conservative under strong correlation. Useful when speed matters or when Monte Carlo is overkill.

The `"mc_gaussian"` and `"analytical"` methods use a variance scaling factor $c(n)$ depending on the family:

- Gaussian / Binomial / Poisson / Exponential: $c(n) = 1$
- Gumbel: $c(n) = \exp(2(\gamma + 1))$, with γ the Euler-Mascheroni constant.
- Cox: $c(n) = 1/(4 \log n)$

Computational cost:

Both Monte Carlo methods are dominated by the $p \times n_{\text{simu}}$ matrix product $X^\top R$, where R stacks the simulated residuals. This product is dispatched to BLAS as `gemm` calls over blocks of draws, which bounds the memory footprint without changing the result, and is essentially as fast as a Monte Carlo selector can be on dense designs. For very large n or p, however, the constant becomes large and `"mc_exact"` may be unnecessarily expensive:

- As n grows, the central-limit approximation tightens and `"mc_gaussian"` gives essentially the same $\hat{\lambda}$ as `"mc_exact"` at a fraction of the cost (no family-specific residual generation, fewer dependencies on y-draws).

- `"analytical"` is $O(1)$ and useful as a quick upper bound — conservative, but accurate enough for triage and for weakly correlated designs, where the union bound is nearly tight.

Practical rule of thumb: prefer `"mc_exact"` for small to moderate problems (default), `"mc_gaussian"` once n grows and the design is dense, and `"analytical"` when even the Monte Carlo cost is a concern.

Details on `alpha` option:

The level α is the nominal Type-I error of the test of H_0: $\beta = 0$. By construction of the quantile,

$$\Pr\Big(\|\nabla \ell_0\|_\infty > \hat{\lambda}_\alpha^{\mathrm{PDB}} \,\Big|\, H_0\Big) = \alpha,$$

so under the null model at least one variable enters the active set with probability at most α. With the default $\alpha = 0.05$, this caps the false-discovery rate at 5% under the null: when the data carry no signal, picreg returns the empty support 95% of the time.

Value

An object of class `"pic.lambda_pdb"`.

`value`	Selected regularization parameter $\hat{\lambda}_\alpha^{\mathrm{PDB}}$.
`statistics`	Simulated null statistics used to estimate the quantile. `NULL` for the analytical method.
`method`	Estimation method used.
`alpha`	Quantile level parameter.
`n_simu`	Number of Monte Carlo simulations (`NA` for the analytical method).
`seed`	The seed used, or `NULL`.
`family`	Name of the family.
`n`, `p`	Dimensions of `X`.

Examples

```
X <- scale(matrix(rnorm(100 * 20), 100, 20))
lam <- lambda_pdb(
  X,
  family = "gaussian",
  method = "mc_exact"
)
print(lam)
```

pdb_asymptotic *Asymptotic behavior of the PDB null distribution.*

Description

For each n in n_grid, draws a standardized Gaussian design matrix of shape (n, p) and computes the null gradient-norm statistic via the three available selectors: "mc_exact", "mc_gaussian", and "analytical". Stores the simulated Monte Carlo statistics and the three resulting $\hat{\lambda}$ values per n.

Usage

```
pdb_asymptotic(
  n_grid,
  p,
  type = c("gaussian", "binomial", "poisson", "exponential", "gumbel", "cox"),
  alpha = 0.05,
  n_simu = 2000L,
  verbose = FALSE
)
```

Arguments

n_grid	Integer vector of sample sizes to evaluate.
p	Number of features (scalar integer).
type	Family name: "gaussian", "binomial", "poisson", "exponential", "gumbel", or "cox".
alpha	Nominal level used for the (1 - alpha) quantile.
n_simu	Monte Carlo size for each selector. Default 2000.
verbose	Logical; if TRUE, prints a one-line progress message per n.

Details

The intended use is to **visualize the convergence** of the exact family-specific null distribution to the Gaussian approximation as n grows — i.e., to check empirically that mc_gaussian is a valid substitute for mc_exact in the asymptotic regime.

Value

An object of class c("pic.pdb_asymptotic", "pic.diagnostic").

n_grid, p, type, alpha, n_simu
: Configuration.

stats_exact, stats_gaussian
: Lists of length length(n_grid) where each element is a numeric vector of length n_simu containing the simulated null statistics from the corresponding selector.

`lambda_exact`, `lambda_gaussian`, `lambda_analytical`
: Numeric vectors of length `length(n_grid)` - the (1 - alpha) quantile under each selector at each `n`.

`call`
: The call.

See Also

`plot.pic.pdb_asymptotic()` for visualization.

Examples

```
as_ <- pdb_asymptotic(n_grid = c(50, 200, 1000),
                      p = 200, type = "poisson")
plot(as_)
```

---

pdb_summary *Summary of the PDB lambda selector.*

Description

Prints a formatted summary of the Pivotal Detection Boundary selector used by `pic()` to choose $\hat{\lambda}$: method, nominal level, Monte Carlo size, selected $\hat{\lambda}$, and a compact view of the null distribution when Monte Carlo was run. For models fitted with `method = "analytical"` or with a user-supplied `lambda`, only the selector metadata is shown.

Usage

```
pdb_summary(model, digits = 4L)
```

Arguments

`model`
: A fitted `pic` object.

`digits`
: Number of significant digits used in the distribution table.

Value

Invisibly returns `NULL`. Called for its side effect (printing).

Examples

```
data(QuickStartExample)
fit <- pic(QuickStartExample$X, QuickStartExample$y,
           family = "gaussian", penalty = "lasso")
pdb_summary(fit)
```

phase_transition *Phase-transition analysis of support recovery.*

Description

For a fixed `(n, p)` configuration, varies the sparsity level `s` from `0` to `s_max` and, for each `s`, estimates by Monte Carlo (`m` replications) the probability that `pic()` recovers exactly the support of the true coefficient vector. If several penalties are supplied, one result is produced for each `(n, p, penalty)` combination.

Usage

```
phase_transition(
  n,
  p,
  type = c("gaussian", "binomial", "poisson", "exponential", "gumbel", "cox"),
  s_max,
  m = 50,
  penalty = "lasso",
  beta_value = 3,
  lambda_method = "mc_exact",
  lambda_alpha = 0.05,
  lambda_n_simu = 2000L,
  verbose = FALSE,
  parallel = FALSE,
  workers = NULL
)
```

Arguments

`n`	Integer or integer vector — number of observations per configuration.
`p`	Integer or integer vector of the same length as `n` — number of features.
`type`	Family name: `"gaussian"`, `"binomial"`, `"poisson"`, `"exponential"`, `"gumbel"`, or `"cox"`.
`s_max`	Largest sparsity level evaluated. Must satisfy `s_max < min(p)`.
`m`	Number of Monte Carlo replications per `s`.
`penalty`	One or more penalties for `pic()`: `"lasso"`, `"scad"`, or `"mcp"`.
`beta_value`	Magnitude of the non-zero coefficients used to generate the response. The sign is fixed to `+`.
`lambda_method`	Passed to `pic()`. Default `"mc_exact"`.
`lambda_alpha`	Nominal level for the PDB selector.
`lambda_n_simu`	Monte Carlo size for the PDB selector. Default 2000.
`verbose`	Logical; if `TRUE`, prints a one-line progress message per `(n, p, penalty, s)`.

`parallel` Logical; if `TRUE`, distribute the `m` Monte Carlo replications of each `(n, p, penalty, s)` cell across multiple R processes via the `future` framework (a `future::multisession` plan, the caller's plan being restored on exit). Requires the **future** and **future.apply** packages.

`workers` Integer; number of background R processes to use when `parallel = TRUE`. Default `NULL` uses all available cores but one (`future::availableCores(omit = 1)`, which honors the limits set by CRAN or a scheduler). Ignored otherwise.

Details

At each Monte Carlo replicate a design matrix is drawn from a standard Gaussian, `s` features are sampled uniformly at random, the response is generated under the chosen `type`, and one `pic` fit is run for each requested penalty. The selected support is compared to the truth and three metrics are stored:

`exact_recovery` 1 if the selected set equals the true set, 0 otherwise.

`tpr` $\left|\hat{S} \cap S\right| / |S|$ - true positive rate. For `s = 0`, this is set to 1 when no true feature exists.

`fdr` $\left|\hat{S} \setminus S\right| / \max\left(|\hat{S}|, 1\right)$ - false discovery rate.

Fits that fail are not counted as failed recoveries: they are excluded from the averages, counted in the `n_failed` column of `results`, and reported in a single warning.

Details on data sampling:

At each replicate, the design X is drawn iid $\mathcal{N}(0,1)$, the true support S is sampled uniformly at random, and $\beta_j =$ `beta_value` for $j \in S$, $\beta_j = 0$ otherwise. The linear predictor is $\eta = X\beta$. The response y is then drawn conditionally on η according to the requested family:

`"gaussian"` $y_i = \eta_i + \varepsilon_i$ with $\varepsilon_i \sim \mathcal{N}(0,1)$.

`"binomial"` $y_i \sim \text{Bernoulli}(\sigma(\eta_i))$, where $\sigma(z) = 1/(1+e^{-z})$ is the logistic function.

`"poisson"` $y_i \sim \text{Poisson}(e^{\eta_i})$.

`"exponential"` $y_i \sim \text{Exp}(\text{rate} = e^{\eta_i})$.

`"gumbel"` $y_i = \eta_i + \varepsilon_i$ with $\varepsilon_i \sim \text{Gumbel}(0,1)$, drawn as $-\log(-\log U_i)$ for $U_i \sim \mathcal{U}(0,1)$.

`"cox"` Event times $T_i \sim \text{Exp}(e^{\eta_i})$ and independent censoring times $C_i \sim \text{Exp}(1)$. The response is the 2-column matrix $\big(\min(T_i, C_i),\ \mathbf{1}\{T_i \le C_i\}\big)$.

Value

An object of class `c("pic.phase_transition", "pic.diagnostic")`.

`results` Long-format `data.frame` with one row per `(n, p, penalty, s)` and columns `n`, `p`, `penalty`, `s`, `exact_recovery`, `tpr`, `fdr`, `n_failed`.

`exact_recovery`, `tpr`, `fdr` Wide-format `data.frames` with one row per `(n, p, penalty)` curve (columns `n`, `p`, `penalty`) and one column `s_<k>` per sparsity level.

`s_grid` `0:s_max`.

`config` A list of all configuration arguments for downstream plotting / reporting.

`call` The call.

See Also

plot.pic.phase_transition() for visualization.

Examples

```
pt <- phase_transition(n = 50, p = 100, type = "gaussian",
                       s_max = 8, m = 20,
                       penalty = c("lasso", "scad"))
plot(pt)
```

---

pic *Sparse linear regression using the Pivotal Information Criterion.*

Description

Fits sparse regression, classification, and survival models built on a linear predictor, using a family-specific loss whose gradient at the null is (asymptotically) pivotal, combined with a sparsity-inducing penalty. Supported families are Gaussian, binomial, Poisson, exponential, Gumbel, and Cox proportional hazards.

Usage

```
pic(
  X,
  y,
  family = c("gaussian", "binomial", "poisson", "exponential", "gumbel", "cox"),
  penalty = "lasso",
  standardize = TRUE,
  intercept = TRUE,
  lambda_method = c("mc_exact", "mc_gaussian", "analytical"),
  mcp_gamma = 3,
  scad_a = 3.7,
  lambda = NULL,
  lambda_alpha = 0.05,
  lambda_n_simu = 2000,
  lambda_seed = NULL,
  tol = 1e-05,
  path_length = 10L,
  maxit = 1000
)
```

Arguments

X	Numeric design matrix of shape (n, p), where rows are observations and columns are candidate predictors. X may be a matrix or a data frame coercible to a numeric matrix. It must contain at least two columns and no NA, NaN, or infinite values.

`y` Response variable with length `n`. For Gaussian and Gumbel models, `y` must be numeric. For Binomial models, `y` must contain only `0` and `1`, or be logical (`TRUE` = 1) or a two-level factor (second level = 1). For Poisson and Exponential models, `y` must be non-negative. For `family = "cox"`, `y` must be a two-column numeric matrix `(time, event)`, where `time` is non-negative and `event` is coded as `0` or `1`.

`family` A character name determining the response distribution and the associated loss function used during optimization. See **Details on family-specific losses** below for the exact objective functions associated with each family. Default `"gaussian"`.

`penalty` Penalty name: one of `"lasso"`, `"scad"`, or `"mcp"` (lowercase). Default `"lasso"`. See pic_penalties for the precise form of each penalty and the role of `scad_a` / `mcp_gamma`.

`standardize` Whether to standardize columns of `X` to zero mean and unit variance prior to computing the PDB regularization parameter and fitting the model. Strongly recommended for proper PDB calibration. When `TRUE`, the optimization is performed on the standardized design matrix and the stored fitted coefficients therefore correspond to the standardized scale. Use `coef.pic()` to recover coefficients on the original scale of `X`. Default is `TRUE`. Standardization centers `X`; for non-Cox families the resulting column-mean shift is absorbed by the intercept, so combining `standardize = TRUE` with `intercept = FALSE` assumes the data are already centered (see `intercept`).

`intercept` Whether to fit an unpenalized intercept. Forced to `FALSE` for the Cox family. When `intercept = FALSE`, the data are assumed to be centered.

`lambda_method` One of `"mc_exact"`, `"mc_gaussian"`, `"analytical"`. See `lambda_pdb()` for details. The default `"mc_exact"` is a safe choice for small to moderate designs; for very large n or p, `"mc_gaussian"` matches `"mc_exact"` closely at a fraction of the cost (it skips the family-specific null draws and residual evaluations), and `"analytical"` is closed-form.

`mcp_gamma` MCP concavity parameter when `penalty = "mcp"`. Default 3.0.

`scad_a` SCAD concavity parameter when `penalty = "scad"`. Default 3.7.

`lambda` Optional regularization parameter. When `NULL`, `lambda_pdb()` is used to select it automatically. Either a single non-negative number, or a `"pic.lambda_pdb"` object (typically `fit$lambda_pdb` from a previous fit on the same `X` and family), whose calibration is then reused and kept on the new fit. Providing `lambda` avoids recomputing the PDB calibration, which is useful for example when fitting several penalties (`"lasso"`, `"scad"`, `"mcp"`) on the same dataset, since the PDB choice of λ does not depend on the penalty itself.

`lambda_alpha` Nominal level for PDB. See `lambda_pdb()`.

`lambda_n_simu` Number of Monte Carlo draws for PDB. See `lambda_pdb()`.

`lambda_seed` Optional seed for the Monte Carlo PDB calibration, for a reproducible λ. Only the calibration is seeded; the global random stream is left unchanged. Default `NULL` (no seeding).

`tol` FISTA convergence tolerance at the final path step.

`path_length` Number of points in the warm-start regularization path running from $\lambda_{\max}$ down to λ^{PDB}. Default 10.

`maxit` Maximum number of FISTA iterations per path step. A warning is issued if the final step does not converge within `maxit`.

Details

Minimizes the objective function

$$\hat{\beta} = \arg\min_{\beta} \; L(\beta) \; + \; \lambda_{\alpha}^{\mathrm{PDB}} \, \mathrm{pen}(\beta) \quad \text{with} \quad L(\beta) = \phi(\ell_n(g(\beta))) \, ,$$

where the regularization parameter is selected automatically using the Pivotal Detection Boundary (PDB) principle implemented in `lambda_pdb()`. The PDB choice is not just a substitute for cross-validation: by calibrating λ against a pivotal null-distribution statistic rather than out-of-sample prediction error, it yields sharper support recovery - fewer false positives on noise variables and more accurate identification of the true non-zero coefficients. Here, ϕ and g are family-specific transformations chosen so that the gradient at the null has a distribution free of the nuisance parameter, which is precisely what makes the use of $\lambda_{\alpha}^{\mathrm{PDB}}$ valid. Optimization follows Section 3.3 of the paper: a warm-started FISTA path with the ℓ_1 penalty runs from $\lambda_{\max}$ (the smallest λ for which $\beta = 0$ is a solution) down to λ; for SCAD and MCP, a final stage then minimizes the criterion with the target penalty, initialized at the ℓ_1 solution.

Details on family-specific losses:

`"gaussian"` Uses the mean squared error for ℓ_n, with $\phi(\cdot) = \sqrt{\cdot}$ and $g(\cdot) = \mathrm{Id}$, giving the square-root lasso loss

$$L(\beta) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \mathbf{x}_i^{\top} \beta) \right)^2} .$$

This makes the gradient at $\beta = 0$ scale-free, so the PDB threshold needs no estimate of the noise standard deviation.

`"binomial"` A variance-stabilized transformation of the Bernoulli likelihood. With $\phi(\cdot) = \mathrm{Id}$ and the logistic link $\theta_i = g(\eta_i) = (1 + e^{-\eta_i})^{-1}$, the loss is

$$L(\beta) = \frac{1}{n} \sum_{i=1}^{n} \left(2 y_i \sqrt{\frac{1 - \theta_i}{\theta_i}} + 2(1 - y_i) \sqrt{\frac{\theta_i}{1 - \theta_i}} \right) .$$

The classical logistic link is preserved; only the loss itself is modified to obtain a pivotal gradient at the null.

`"poisson"` The same pivotalization applied to the Poisson likelihood. With $\phi(\cdot) = \mathrm{Id}$ and the canonical log link $\theta_i = g(\eta_i) = e^{\eta_i}$, the loss is

$$L(\beta) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{2 y_i}{\sqrt{\theta_i}} + 2 \sqrt{\theta_i} \right) .$$

The canonical log link is kept unchanged; pivotality is obtained through the loss rather than through the link.

`"exponential"` Uses the standard Exponential negative log-likelihood directly (no transformation). With $\phi(\cdot) = \mathrm{Id}$ and $\theta_i = g(\eta_i) = e^{\eta_i}$, the loss is

$$L(\beta) = \frac{1}{n} \sum_{i=1}^{n} \left(\log \theta_i + \frac{y_i}{\theta_i} \right) .$$

`"gumbel"` A location-scale model with extreme-value noise. The base negative log-likelihood is

$$\ell_n(\theta, \sigma) = \log(\sigma) + \frac{1}{n}\sum_{i=1}^{n}\left(z_i + e^{-z_i}\right), \qquad z_i = \frac{y_i - \theta_i}{\sigma}.$$

With $\phi(\cdot) = \exp(\cdot)$ and the identity link $\theta_i = g(\eta_i) = \eta_i$, the objective is $L(\beta, \sigma) = \exp(\ell_n(\theta, \sigma))$. The scale σ is re-estimated internally by maximum likelihood at every iteration; the user does not supply it.

`"cox"` A square-root-transformed Cox partial log-likelihood. With $\phi(\cdot) = \sqrt{\cdot}$ and the identity link $\theta_i = g(\eta_i) = \eta_i$, the objective is

$$L(\beta) = \sqrt{-\frac{1}{n}\left(\sum_{i=1}^{n}\delta_i\eta_i - \sum_{i=1}^{n}\delta_i \log\left(\sum_{j\in R_i} e^{\eta_j}\right)\right)},$$

where δ_i is the event indicator and R_i the risk set at event time t_i. The Breslow approximation is used for tied event times, and the model is always fitted without an intercept.

Value

An object of class `c("pic.<family>", "pic")`.

`call`	The matched call.
`beta`	Fitted coefficients (length `p`), on the standardized scale when `standardize = TRUE`. Use `coef.pic()` for the original scale of `X`.
`intercept`	Fitted intercept or `NULL`.
`df`	The number of nonzero coefficients.
`selected`	Names of the selected variables when `X` has column names, otherwise their integer column indices.
`lambda`	λ used during training (PDB or user-supplied).
`family`	The family object used.
`penalty`	The penalty object used.
`lambda_pdb`	The `lambda_pdb()` calibration, or `NULL` when a numeric λ is supplied.
`n_samples`	The number of observations in the training set.
`n_features`	The number of variables in the training set.
`n_iter`	Total number of FISTA iterations over the path.
`converged`	Whether the final FISTA step reached `tol` within `maxit` iterations.

For `family = "cox"`, the fit additionally carries:

`baseline_cumulative_hazard`	Estimated Breslow baseline cumulative hazard function.
`baseline_survival`	Estimated baseline survival function.
`unique_times`	Sorted unique event times used in the baseline estimation.
`censoring_rate`	Percentage of censored observations in the training set.

Examples

```
data(QuickStartExample)
X <- QuickStartExample$X
y <- QuickStartExample$y
fit <- pic(X, y, family = "gaussian", penalty = "scad")
fit$beta
fit$selected
```

`pic_families` *GLM families for pic — descriptor layer.*

Description

Each family is a thin descriptor carrying:

- `name` - used by the dispatchers to route to the C++ implementation.
- `g` - mean-function link (object with `name` and callable `fn`). `g$fn(eta)` is applied at predict time to obtain the mean response.
- `phi` - output transform applied to the base loss (object with `name`). Purely informational on the R side; the actual transform is applied inside the C++ math.

Details

Typing `fit$family` gives a one-glance summary of the model's family (see `print.pic.family()`). The actual loss / gradient math lives in C++:

- `src/family_*.cpp` — Gaussian, Binomial, Poisson, Exponential, Gumbel.
- `src/cox.cpp` — Cox.

`pic_methods` *S3 methods for fitted pic objects.*

Description

The fitted object has class `c("pic.<family>", "pic")` — methods dispatch on the second class for shared behavior and on the first class for family-specific overrides.

`pic_penalties` *Sparsity-inducing penalties for pic.*

Description

Three penalties are supported, identified by lowercase name to match the C++ registry. Each penalty enters the `pic()` objective as

$$\mathrm{pen}(\beta) = \sum_{j=1}^{p} p_\lambda(|\beta_j|),$$

where $p_\lambda(\cdot)$ depends on the penalty.

Details

`"lasso"` L1 (soft-thresholding) penalty:

$$p_\lambda(|t|) = \lambda|t|.$$

Convex. Of the three penalties it shrinks large coefficients the most: its bias does not vanish as $|t| \to \infty$.

`"scad"` **(Smoothly Clipped Absolute Deviation, Fan & Li 2001)** Non-convex penalty with concavity parameter `scad_a` > 2 (default 3.7):

$$p'_\lambda(|t|) = \lambda\left\{\mathbf{1}\{|t| \le \lambda\} + \frac{(a\lambda - |t|)_+}{(a-1)\lambda}\mathbf{1}\{|t| > \lambda\}\right\}.$$

Behaves like the lasso for small $|t|$, then tapers off so large coefficients are barely penalized - yields nearly unbiased estimates on strong signals.

`"mcp"` **(Minimax Concave Penalty, Zhang 2010)** Non-convex penalty with concavity parameter `mcp_gamma` > 1 (default 3.0):

$$p'_\lambda(|t|) = \left(\lambda - \frac{|t|}{\gamma}\right)_+.$$

Similar motivation as SCAD but a smoother transition: starts at the lasso derivative for small $|t|$ and tapers linearly to zero at $|t| = \gamma\lambda$.

The actual evaluation and proximal operators live in C++ (`src/penalty_*.cpp`). Larger `scad_a` / `mcp_gamma` make the penalty closer to the lasso; smaller values amplify the non-convexity (and the bias reduction on strong signals).

pic_plots *Plot methods for pic fits and diagnostics.*

Description

Entry points:

Details

- `plot(fit)` - lollipop plot of the non-zero coefficients.
- `plot(fit$lambda_pdb)` - histogram of the PDB null distribution with a vertical line at the selected $\hat{\lambda}$.
- `plot_baseline()` - Cox-only: baseline cumulative hazard and baseline survival, two panels.
- `plot(phase_transition(...))` - recovery curves.
- `plot(pdb_asymptotic(...))` - null-distribution convergence panels.

pic_survival *Survival utilities for the Cox family.*

Description

Provides Harrell's C-index, the Breslow partial log-likelihood, baseline cumulative hazard / survival, and feature-effect curves.

plot.pic *Horizontal lollipop plot of the non-zero coefficients of a pic fit.*

Description

One row per selected variable, sorted by descending absolute coefficient value (largest at the top). Each variable is drawn as a horizontal segment from zero to its fitted value.

Usage

```
## S3 method for class 'pic'
plot(x, standardized = TRUE, max_features = NULL, ...)
```

Arguments

`x`	A fitted `pic` object.
`standardized`	Logical; if `TRUE`, plot the standardized coefficients used internally during fitting. If `FALSE`, use the coefficients on the original scale of `X`. Default `TRUE`.
`max_features`	Optional cap on the number of features displayed (the strongest are kept).
`...`	Additional graphical parameters forwarded to `graphics::plot()` for the empty frame.

Value

Invisibly returns the plotted (named) coefficient vector.

`plot.pic.lambda_pdb` *Plot the PDB null distribution.*

Description

Histogram of the simulated null gradient-norm statistics, with a vertical line at the selected $\hat{\lambda}$.

Usage

```
## S3 method for class 'pic.lambda_pdb'
plot(x, breaks = 40L, ...)
```

Arguments

`x`	A `pic.lambda_pdb` object (typically `fit$lambda_pdb`).
`breaks`	Number of histogram bins (default 40).
`...`	Unused; present for S3 method consistency.

Value

Invisibly returns `x`.

`plot.pic.pdb_asymptotic`
Plot of the PDB asymptotic behavior.

Description

Multi-panel histogram comparison of the simulated null gradient-norm statistic under the `"mc_exact"` (light grey fill) and `"mc_gaussian"` (dashed outline) selectors, one panel per `n` in `n_grid`. Two vertical lines are added per panel:

- $\hat{\lambda}_{\alpha}^{\mathrm{PDB}}$ - solid navy, the empirical (1 - alpha) quantile of `"mc_exact"` (what pic would actually use).
- $\hat{\lambda}_{analytical}$ - dotted black, the Bonferroni closed-form bound.

Usage

```
## S3 method for class 'pic.pdb_asymptotic'
plot(x, breaks = 40L, ...)
```

Arguments

x	An object returned by pdb_asymptotic().
breaks	Number of histogram bins (default 40).
...	Unused; present for S3 method consistency.

Value

Invisibly returns x.

---

plot.pic.phase_transition

Phase-transition plot for a pic.phase_transition *object.*

Description

Plots the chosen recovery metric as a function of the sparsity level s. Curves are distinguished by line type (grayscale ramp beyond five curves). When multiple (n, p) configurations are compared across several penalties, panels are laid out in a grid with at most three columns per row.

Usage

```
## S3 method for class 'pic.phase_transition'
plot(x, metric = c("exact_recovery", "tpr", "fdr"), ...)
```

Arguments

x	An object returned by phase_transition().
metric	One of "exact_recovery" (default), "tpr", "fdr".
...	Additional graphical parameters forwarded to graphics::plot().

Value

Invisibly returns x.

plot_baseline *Plot Cox baseline cumulative hazard and baseline survival.*

Description

Two-panel step plot for a fitted `pic.cox` model: the Breslow baseline cumulative hazard $H_0(t)$ on top and the baseline survival $S_0(t) = \exp(-H_0(t))$ below.

Usage

```
plot_baseline(model)
```

Arguments

`model`	A fitted `pic.cox` object.

Value

Invisibly returns `NULL`.

plot_survival_curves *Plot subject-specific Cox survival curves.*

Description

Step-line visualization of the output of `predict.pic()` with `type = "survival"` or `feature_effects_on_survival()`: one survival curve per subject (or per feature value) on a common time grid. To keep curves distinguishable, each curve is drawn with its own line type and a sparse set of marker glyphs is overlaid at regular time points.

Usage

```
plot_survival_curves(
  sf,
  subjects = NULL,
  max_subjects = 10L,
  labels = NULL,
  n_marks = 8L,
  main = "Individual survival curves",
  ...
)
```

Arguments

sf	A list as returned by predict.pic() with type = "survival" or feature_effects_on_survival(), with components time (length K) and survival (matrix K x m, one column per curve).
subjects	Optional integer vector selecting which columns of sf$survival to plot. Defaults to all curves, capped to the first max_subjects for legibility.
max_subjects	Maximum number of curves drawn when subjects is NULL. Default 10.
labels	Optional character vector of labels used in the legend, one per plotted curve. Defaults to the column names of sf$survival when set, otherwise "subject 1", "subject 2", etc.
n_marks	Number of marker glyphs overlaid on each curve to help distinguish them. Default 8. Set to 0 to disable.
main	Plot title.
...	Additional graphical parameters forwarded to graphics::matplot().

Value

Invisibly returns NULL.

---

predict.pic *Linear predictor / response prediction for a pic fit.*

Description

Linear predictor / response prediction for a pic fit.

Usage

```
## S3 method for class 'pic'
predict(object, newx, type = c("response", "link", "class", "survival"), ...)
```

Arguments

object	A fitted pic object.
newx	Matrix of new values at which predictions are to be made.
type	"link" (linear predictor), "response" (default; family g-link applied, and for Cox the relative risk $\exp(\eta)$, as in **glmnet**), "class" (binomial only; 0/1 labels, where 1 stands for TRUE or the second factor level of the training response), or "survival" (Cox only; returns a list with components time and survival).
...	Unused; present for S3 method consistency.

Value

A numeric vector for types "link", "response", and "class". For type = "survival", a list with time (length K) and survival (matrix K x m, one column per subject).

print.pic.family *Pretty-print a pic family descriptor.*

Description

One-line summary showing the family name and its (g, phi) pair.

Usage

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

Arguments

x	A pic.family object.
...	Ignored.

Value

Invisibly returns x.

print.pic.pdb_asymptotic *Print PDB asymptotic diagnostic.*

Description

Print PDB asymptotic diagnostic.

Usage

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

Arguments

x	A pic.pdb_asymptotic object.
...	Unused; present for S3 method consistency.

Value

Invisibly returns x.

print.pic.phase_transition *Print phase-transition analysis.*

Description

Print phase-transition analysis.

Usage

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

Arguments

x	A pic.phase_transition object.
...	Unused; present for S3 method consistency.

Value

Invisibly returns x.

print.summary.pic *Print a* summary.pic *object.*

Description

Print a summary.pic object.

Usage

```
## S3 method for class 'summary.pic'
print(x, digits = 4L, ...)
```

Arguments

x	A summary.pic object.
digits	Number of significant digits for the printed values.
...	Unused; present for S3 method consistency.

Value

Invisibly returns x.

`QuickStartExample` *Small Gaussian dataset for the introductory vignette.*

Description

A synthetic Gaussian regression dataset used to illustrate `pic()` throughout the introductory vignette. It contains $n = 100$ observations of $p = 30$ predictors, of which only $s = 5$ carry signal; the remaining 25 are pure noise.

Usage

```
QuickStartExample
```

Format

A list with two components:

`X` Numeric matrix of dimension 100×30 with column names `gene_*` and `noise_*`.

`y` Numeric vector of length 100.

Details

Column names are chosen to make the underlying support obvious at a glance:

- `gene_1, ..., gene_5`: the five active variables, whose non-zero coefficients are drawn uniformly in $[0.5,\ 1.5]$ with random sign.
- `noise_1, ..., noise_25`: the remaining inactive variables, with true coefficient 0.

The columns are interleaved in random order; column names are the only indicator of which features are part of the true support.

The response is generated as $y = X\beta + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 1)$.

Examples

```
data(QuickStartExample)
fit <- pic(QuickStartExample$X, QuickStartExample$y)
fit$selected
```

summary.pic *Summarize a fitted pic model.*

Description

Returns a structured summary of the fit: the family, penalty, selected $\hat{\lambda}$, the problem dimensions, the number of selected variables, the intercept, and a table of the **non-zero** coefficients ordered by decreasing absolute value. Coefficients are returned on the original scale of `X` by default (pass `standardized = TRUE` for the internal standardized scale, matching `fit$beta`).

Usage

```
## S3 method for class 'pic'
summary(object, standardized = FALSE, ...)
```

Arguments

`object`	A fitted `pic` object.
`standardized`	Logical; if `TRUE`, summarize the standardized coefficients used internally. Default `FALSE` (original scale of `X`).
`...`	Unused; present for S3 method consistency.

Value

An object of class `"summary.pic"`: a list with elements `family`, `penalty`, `lambda`, `n_samples`, `n_features`, `df`, `intercept`, `standardized`, and `coefficients` (a two-column `data.frame` of the non-zero coefficients).

Examples

```
data(QuickStartExample)
fit <- pic(QuickStartExample$X, QuickStartExample$y)
summary(fit)
```

Index

∗ **datasets**
BinomialExample, 4
CoxExample, 7
QuickStartExample, 29

assess, 3

baseline_functions, 4
BinomialExample, 4

coef.pic, 5
coef.pic(), *17*, *19*
concordance_index, 6
cox_partial_log_likelihood, 7
CoxExample, 7

feature_effects_on_survival, 8
feature_effects_on_survival(), *25, 26*

get_family(), *9*
graphics::matplot(), *26*
graphics::plot(), *22*, *24*

lambda_pdb, 9
lambda_pdb(), *17–19*

pdb_asymptotic, 12
pdb_asymptotic(), *24*
pdb_summary, 13
phase_transition, 14
phase_transition(), *24*
pic, 16
pic(), *13*, *21*
pic_families, 20
pic_methods, 20
pic_penalties, *17*, 21
pic_plots, 22
pic_survival, 22
plot.pic, 22
plot.pic.lambda_pdb, 23
plot.pic.pdb_asymptotic, 23
plot.pic.pdb_asymptotic(), *13*
plot.pic.phase_transition, 24
plot.pic.phase_transition(), *16*
plot_baseline, 25
plot_baseline(), *22*
plot_survival_curves, 25
plot_survival_curves(), *8, 9*
predict.pic, 26
predict.pic(), *6*, *8, 9*, *25, 26*
print.pic.family, 27
print.pic.family(), *20*
print.pic.pdb_asymptotic, 27
print.pic.phase_transition, 28
print.summary.pic, 28

QuickStartExample, *5*, *8*, 29

summary.pic, 30