Package: bamp (via r-universe)

October 30, 2026

Type Package

Title Bayesian Age-Period-Cohort Modeling and Prediction

Version 3.0.0

Date 2026-08-31

Author Volker Schmid [aut, cre] (ORCID: <https://orcid.org/0000-0003-2195-8130>), Florian Geressen [ctb], Leonhard Held [ctb], Evi Rainer [ctb], Chris Kypridemos [aut] (ORCID: <https://orcid.org/0000-0002-0746-9229>)

Maintainer Volker Schmid <volker.schmid@lmu.de>

Description Bayesian Age-Period-Cohort Modeling and Prediction using efficient Markov Chain Monte Carlo Methods. This is the R version of the previous BAMP software as described in Volker Schmid and Leonhard Held (2007) <DOI:10.18637/jss.v021.i08> Bayesian Age-Period-Cohort Modeling and Prediction - BAMP, Journal of Statistical Software 21:8. This package includes checks of convergence using Gelman's R.

License GPL-3

Encoding UTF-8

Depends R (>= 3.5.0)

Imports coda, graphics, parallel, stats, abind

LazyData true

VignetteBuilder knitr, R.rsp

Suggests knitr, rmarkdown, R.rsp, testthat (>= 3.0.0)

URL https://volkerschmid.github.io/bamp/, https://github.com/volkerschmid/bamp

BugReports https://github.com/volkerschmid/bamp/issues

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

NeedsCompilation yes

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

Date/Publication 2026-08-31 20:13:15 UTC

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

RemoteRef HEAD

RemoteSha 8102223ede92a22de49163cd721e5459596fb2cd

Contents

apc . . . 2
apc.data . . . 3
apcSimulate . . . 3
bamp . . . 4
checkConvergence . . . 8
coh . . . 9
effects.apc . . . 10
plot.apc . . . 12
predict_apc . . . 13
print.apc . . . 14
selectModel . . . 15

Index **18**

---

apc *apc S3 class*

Description

Class for (Bayesian) age-period-cohort objects

Usage

```
apc()
```

Details

`bamp` will return an object of class apc. Available functions are

- `plot.apc` plots main effects
- `print.apc` print summary of model and effects
- `effects.apc` extract effects (mean, median and quantiles)

Value

apc class

apc.data *Example dataset for APC model*

Description

A dataset containing case counts and population numbers in eight age groups for ten years. Each age group consists of five years.

Usage

```
data(apc)
```

Format

- `population`: matrix of population data
- `cases`: matrix of case counts
- `cov_p`: covariate for period
- `cov_c`: covariate for cohort

apcSimulate *Simulate from age-period-cohort model*

Description

This functions simulates a data set of cases on the Lexis diagram from given age, period and cohort effects. Population numbers have to be given; can be one number for all age group/period combinations.

Usage

```
apcSimulate(intercept, age, period, cohort, periods_per_agegroup, population)
```

Arguments

`intercept`	Intercept
`age`	Vector of effect for age groups
`period`	Vector of effects for periods
`cohort`	Vector of effect for cohorts
`periods_per_agegroup`	Periods per age group
`population`	Population number. Either a matrix or a scalar.

Value

List with number of cases (matrix) and population numbers (matrix).

See Also

```
vignette("simulation", package = "bamp")
```

Examples

```
age=sqrt(seq(5,0,length=10)); age<-1-age-mean(age)
period=15:1; period[8:15]<-8:15; period<-period/6; period<-period-mean(period)
periods_per_agegroup=5; number_of_cohorts <- periods_per_agegroup*(10-1)+15
cohort<-rep(0,60); cohort[1:10]<-10:1; cohort[41:60]<- -(1:20)/2; cohort<-cohort/10;
cohort<-cohort-mean(cohort)
simdata<-apcSimulate(-5, age, period, cohort, periods_per_agegroup, 1e6)
par(mfrow=c(3,1))
plot(age, type="l")
plot(period, type="l")
plot(cohort, type="l")
## Not run:
simmod <- bamp(cases = simdata$cases, population = simdata$population, age = "rw1",
period = "rw1", cohort = "rw1", periods_per_agegroup =periods_per_agegroup)
plot(simmod)

## End(Not run)
```

---

bamp *Bayesian Age-Period-Cohort Modeling and Prediction (bamp)*

Description

Bayesian Age-Period-Cohort Modeling for the analyze of incidence or mortality data on the Lexis diagram. For each pixel in the Lexis diagram (that is for a specific age group and specific period) data must be available on the number of persons under risk (population number) and the number of disease cases (typically cancer incidence or mortality). A hierarchical model is assumed with a binomial model in the first-stage. As smoothing priors for the age, period and cohort parameters random walks of first and second order (RW1 or RW2) available. Deviance information criterion and effective number of parameters is computed for model comparison. Note that there is a non-identifiability in the likelihood of the APC-model, see e.g. Clayton and Schifflers (1987, DOI:10.1002/sim.4780060406), which indices some problems in interpreting the latent effects. Only for RW1 model, the parameters are (weakly) identifiable. Period and age groups do not need to be on the same grid, for example periods can be in one year intervals and age groups in five year intervals.
Additionally to the model described in Knorr-Held and Rainer (2001, DOI:10.1093/biostatistics/2.1.109), bamp can handle

- AP and AC models,
- models with and without global heterogeneity parameter (overdispersion),

- models with additional age, period and/or cohort heterogeneity,
- additional covariates.

Usage

```
bamp(
  cases,
  population,
  age,
  period,
  cohort,
  overdisp = FALSE,
  period_covariate = NULL,
  cohort_covariate = NULL,
  periods_per_agegroup,
 mcmc.options = list(number_of_iterations = "auto", burn_in = "auto", step = "auto",
    tuning = 500),
 hyperpar = list(age = c(1, 0.5), period = c(1, 5e-04), cohort = c(1, 5e-04), overdisp =
    c(1, 0.05)),
  dic = TRUE,
  parallel = TRUE,
  verbose = FALSE,
  method = c("pg", "taylor"),
  prior_scale = FALSE,
  pg_engine = c("C", "R")
)
```

Arguments

`cases`	number of cases
`population`	population number
`age`	prior for age groups ("rw1", "rw2", "rw1+het", "rw2+het", " ")
`period`	prior for periods ("rw1", "rw2", "rw1+het", "rw2+het", " ")
`cohort`	prior for cohorts ("rw1", "rw2", "rw1+het", "rw2+het", " ")
`overdisp`	logical, add overdispersion to model
`period_covariate`	covariate for period
`cohort_covariate`	covariate for cohort
`periods_per_agegroup`	periods per age group
`mcmc.options`	list of options for MCMC. • number_of_iterations: number of iterations per chain. • burn_in: number of iterations used as burnin at the beginning of the algorithm, these iterations will be removed. • step: Step size, so only every step-th iteration is stored.

- tuning: number of iterations for automatic tuning (used by `method="taylor"`). Depending on the model, the MCMC algorithm will tune certain parameters for more efficient MCMC chains. After tuning, the algorithm is restarted.

Each of `number_of_iterations`, `burn_in` and `step` may be a number or the string `"auto"` (the default). `"auto"` chooses the value from the data: rare or zero-heavy counts (whose rare-event cells mix more slowly) get more iterations (from 40000 for well-populated data up to 120000 when almost every cell is empty or has very few events), `burn_in` defaults to half the iterations, and `step` is set to keep about 1000 stored samples per chain. Any value given as a number is used exactly as supplied, so explicit settings reproduce the previous behaviour.

`hyperpar` list of hyper parameters. The hyper prior for the precision (inverse variance) in the random walk priors is a Gamma distribution with parameters a and b; expected value is a/b, variance is a/b^2. Weak hyper parameters are suggested, defaults are $a = 1, b = 0.5$ for age, $a = 1, b = 0.0005$ for period and cohort effects and $a = 1, b = 0.05$ for overdispersion (if added). It is recommended to choose the hyper priors depending on the model, in particular on the order of the random walk.

`dic` logical. If true. DIC will be computed

`parallel` should the chains be run in parallel. `TRUE`/`FALSE`, or a number giving the requested number of cores (capped at the number of chains). Uses the `parallel` package: forked workers (`mclapply`) on Unix and macOS, and – for `method = "pg"` – a PSOCK cluster on Windows (where forking is unavailable), so the default engine now runs in parallel on all platforms. (The legacy `method = "taylor"` engine still runs serially on Windows.) Parallel runs are reproducible: the per-chain seeds are drawn in the main process, so a given `set.seed()` yields the same result serially or in parallel.

`verbose` verbose mode

`method` MCMC engine. `"pg"` (default) is a joint sampler that combines Polya-Gamma data augmentation (Polson, Scott & Windle 2013) with a Laplace (Newton) Metropolis-Hastings refinement: each sweep draws the intercept and the age, period and cohort effects jointly in one exact Gibbs step and then refines them with a joint Newton proposal against the true binomial likelihood. It has no Metropolis tuning, never restarts on low acceptance and does not prune chains; it is markedly more robust for RW2 priors and converges the high-population, rare-event cells of incidence/mortality data that the Gibbs step alone mixes only slowly. It natively supports all of the package's models – RW1/RW2 priors, heterogeneity (`"rw1+het"`/`"rw2+het"`), overdispersion and period/cohort covariates. The Polya-Gamma weights use a normal approximation that is essentially exact for the large population counts of incidence/mortality data, so it typically needs far fewer iterations than the legacy sampler. `"taylor"` is the original block Metropolis-Hastings sampler with taylor expansion proposals (the default in versions 2.x); it remains available and can be faster on well-behaved (non rare-event) data, but it can fail to converge or prune all chains on sparse/zero-cell data.

`prior_scale` logical; only used by `method="pg"`. If `TRUE`, the intrinsic random-walk structure matrices are scaled to unit generalised variance (Sorbye & Rue 2014) so that a single hyper-prior is comparable across random-walk orders, grid sizes and data

sets. The default is FALSE, which keeps the same prior parameterisation (and the same default hyper-parameters) as method="taylor"; if you set it to TRUE you should choose hyper-parameters appropriate for the scaled prior. See 'Scaling the random-walk priors' below for the rationale and benefits, and the examples for a short demonstration.

pg_engine — implementation of the method="pg" sampler, one of "C" (default) or "R". Both run the identical algorithm and, for a given seed, produce the same draws to floating-point tolerance; the "C" engine is a compiled port of the inner loop (no extra package dependency) and is roughly twice as fast. "R" is the readable reference implementation, kept for verification. Ignored for method="taylor".

Details

This functions returns an apc object. Only samples from the posterior are computed, point estimates and credible intervals will be computed in effects.apc, print.apc and plot.apc. predict_apc can be used for for prediction of the future rates and number of cases and for a retrospective prediction for model checking.

Scaling the random-walk priors (prior_scale)

Each age, period and cohort effect has an intrinsic Gaussian (random-walk) prior with precision (smoothing) parameter κ: the effect vector x has density proportional to $\exp(-\frac{1}{2}\kappa\, x'Kx)$, where $K = D'D$ is built from the first- or second-order difference operator D. A $\mathrm{Gamma}(a, b)$ hyper-prior is placed on κ. The difficulty is that the smoothness implied by a given κ is governed not by κ alone but by the marginal variance of the effect, the generalised inverse of κK; and the eigenvalues of K grow with the number of time points and with the random-walk order. The *same* hyper-prior on κ therefore implies very different prior smoothness for, say, an RW1 over 10 periods and an RW2 over 50 cohorts. A hyper-prior tuned on one model silently means something different on another, which is one reason a fixed default can behave inconsistently across data sets.

With prior_scale = TRUE the structure matrix K is rescaled so that the geometric mean of the (generalised) marginal variances equals one (Sorbye and Rue, 2014, DOI:10.1080/01621459.2013.866549). After scaling, $1/\sqrt{\kappa}$ is, to a good approximation, the marginal standard deviation of a typical effect element *on the log-odds (logit) scale*, independently of the random-walk order, the number of age/period/cohort points and the grid spacing.

Benefits: (i) **portable hyper-priors** – one $\mathrm{Gamma}(a, b)$ encodes the same smoothness belief across RW1/RW2 and across data sets of different size; (ii) an **interpretable prior** – you can set (a, b) to express a belief about $1/\sqrt{\kappa}$ as a prior effect standard deviation on the logit scale; (iii) **fairer model comparison** (e.g. RW1 vs RW2 by DIC), because the prior is not implicitly penalising one model far more than another. Scaling affects only the smooth random-walk blocks; the i.i.d. heterogeneity components and overdispersion already have an interpretable scale and are unchanged.

The default is prior_scale = FALSE so that method = "pg" reproduces the prior parameterisation (and default hyperpar) of the legacy method = "taylor" engine. If you turn scaling on you should set hyperpar for the scaled prior, where $\kappa \approx 1/\mathrm{variance}$; using the unscaled defaults with prior_scale = TRUE would impose a different (and probably unintended) amount of smoothing. Scaling is most worthwhile when fitting many models or data sets and you want one coherent, interpretable prior across all of them. The example below shows the effect concretely.

See Also

```
vignette("modeling", package = "bamp")
```

Examples

```
## Not run:
data(apc)
model <- bamp(cases, population, age="rw1", period="rw1", cohort="rw1", periods_per_agegroup = 5)

## End(Not run)

## Demonstration of prior_scale (no MCMC, runs instantly): for a fixed
## precision kappa, report the geometric-mean prior marginal standard
## deviation of a random-walk effect on the logit scale, with and without
## Sorbye-Rue scaling, across random-walk orders and grid sizes.
prior_sd <- function(L, order, kappa = 1, scale = FALSE) {
  K <- crossprod(diff(diag(L), differences = order))   # structure matrix D'D
  if (scale) {                                        # Sorbye-Rue unit-variance scaling
    e <- eigen(K, symmetric = TRUE); keep <- e$values > max(e$values) * 1e-9
    V <- e$vectors[, keep, drop = FALSE]
    Sigma <- V %*% diag(1 / e$values[keep], sum(keep)) %*% t(V)
    K <- K * exp(mean(log(diag(Sigma))))
  }
  e <- eigen(K, symmetric = TRUE); keep <- e$values > max(e$values) * 1e-9
  V <- e$vectors[, keep, drop = FALSE]
  Sig <- V %*% diag(1 / e$values[keep], sum(keep)) %*% t(V) / kappa
  sqrt(exp(mean(log(diag(Sig)))))                      # geometric-mean marginal SD
}
grid <- expand.grid(order = 1:2, L = c(10, 25, 50))
data.frame(grid,
           unscaled = round(mapply(prior_sd, grid$L, grid$order, scale = FALSE), 3),
           scaled   = round(mapply(prior_sd, grid$L, grid$order, scale = TRUE), 3))
## With prior_scale = FALSE the same kappa = 1 implies an effect SD ranging
## from ~1.2 to ~14.6 across these models; with prior_scale = TRUE it is 1.0
## throughout, so a single hyper-prior on kappa means the same smoothness for
## every random-walk order and grid size.
```

---

checkConvergence *Check apc object, whether MCMC has converged*

Description

This function uses Gelman and Rubin's R (potential scale reduction factor) to check convergence. All checked quantities should have R<1.1. bamp runs at least four MCMC chains by default (more if `parallel` is more than four).

Usage

```
checkConvergence(x, info = FALSE, level = 2, auto = FALSE)
```

Arguments

x	An apc object
info	logical; print more information (including the raw per-effect diagnostic, which is affected by the age-period-cohort identifiability and should not be used on its own, see Details)
level	level of check; 1 uses point estimate, 2 uses upper C.I.
auto	logical; should be TRUE if called automatically from bamp

Details

In an age-period-cohort model the age, period and cohort effects are linearly dependent (Clayton and Schifflers, 1987): a linear trend can be moved between the three effects without changing the likelihood. The individual effect chains can therefore drift along this non-identified direction even when the model has fully converged, which makes a naive Gelman-R on the raw effects report spurious non-convergence.

checkConvergence therefore assesses the quantities that are actually identified: the smoothing precisions and the fitted linear predictor (log-odds) in every cell of the Lexis diagram, which is invariant to the trend re-allocation. With info=TRUE the raw per-effect diagnostic is also printed for reference.

Value

logical; TRUE if check is fine.

Examples

```
## Not run:
data(apc)
model <- bamp(cases, population, age="rw1", period="rw1", cohort="rw1", periods_per_agegroup = 5)
checkConvergence(model)

## End(Not run)
```

---

coh *Compute cohort index from age and period index*

Description

Compute cohort index from age and period index

Usage

```
coh(agegroup, period, noa, periods_per_agegroup)
```

Arguments

agegroup	age group index
period	period index
noa	number of age groups in total
periods_per_agegroup	periods per age group

Value

cohort index

Examples

```
# last agegroup in first period equals first cohort
coh(10, 1, 10, 5)

# first agegroup in last period equals last cohort
coh(1, 8, 10, 5)
```

---

effects.apc *Effects from Fitted APC Model*

Description

Effects from Fitted APC Model

Usage

```
## S3 method for class 'apc'
effects(
  object,
  mean = FALSE,
  quantiles = 0.5,
  update = FALSE,
  convention = c("age", "period", "cohort", "none"),
  combined = FALSE,
  ...
)
```

Arguments

object	an apc object
mean	logical. If TRUE, mean effects are computed
quantiles	Scalar or vector of quantiles to compute (only if mean=FALSE)
update	logical. If TRUE, the apc object including the effects is returned

convention	display-layer gauge convention for the linear trend (drift) in a full age-period-cohort model; one of "age" (default), "period", "cohort" or "none". See Details.
combined	logical. For heterogeneity models ("rw1+het" / "rw2+het"), if TRUE the returned effect is the full effect (smooth + iid heterogeneity component); if FALSE (default) only the smooth component is returned. Ignored for models without heterogeneity.
...	Additional arguments will be ignored

Details

In a full age-period-cohort model the age, period and cohort effects are identifiable only up to one shared linear trend (drift), because a linear trend can be moved between the three effects without changing the fitted rates (Clayton and Schifflers, 1987). When the raw posterior samples are summarised directly, this non-identified trend makes the effect curves drift between MCMC iterations and between runs, so they are not reproducible.

convention fixes that single degree of freedom for display: "age" removes the linear slope of the age effect (age is shown as curvature about a zero trend, the drift is shown in the period and cohort effects); "period" and "cohort" pin the corresponding effect's slope to zero instead; "none" returns the raw, un-gauged effects (previous behaviour, curves may differ between runs). The gauge is applied per MCMC draw before the quantiles are computed, and only for full APC models – for models without all three effects there is no trend aliasing and the argument is ignored.

Fixing the gauge removes the run-to-run linear-trend (drift) component, which is typically the dominant source of non-reproducibility in the effect curves; the residual curvature and Monte-Carlo sampling noise are unaffected, so two independent runs are made much closer but need not agree exactly. The zero-slope property holds for every individual MCMC draw; because quantiles are non-linear, the summarised median/quantile curve has only approximately (not exactly) zero slope.

The gauge is display-only: it never modifies the stored samples (object$samples), and the fitted rates, the predictions from predict_apc and the DIC are invariant to it – only the way the common linear trend is split among the three curves changes. The convention actually used is recorded in attr(result, "gauge_convention").

Value

List of age, period, cohort effects or apc object including effects (if update=TRUE)

Examples

```
## Not run:
data(apc)
model <- bamp(cases, population, age="rw1", period="rw1", cohort="rw1", periods_per_agegroup = 5)
effects(model)

## End(Not run)
```

`plot.apc` *Plot apc object*

Description

Plot apc object

Usage

```
## S3 method for class 'apc'
plot(
  x,
  quantiles = c(0.05, 0.5, 0.95),
  convention = c("age", "period", "cohort", "none"),
  combined = FALSE,
  ...
)
```

Arguments

`x`	apc object
`quantiles`	quantiles to plot. Default: `c(0.05,0.5,0.95)` is median and 90% credible interval.
`convention`	display-layer gauge convention for the linear trend (drift) in a full age-period-cohort model; one of `"age"` (default), `"period"`, `"cohort"` or `"none"`. In a full APC model the three effects are identifiable only up to a shared linear trend; this fixes that single degree of freedom, removing the run-to-run drift that is the dominant source of non-reproducibility in the plotted curves (residual curvature and Monte-Carlo noise remain). `"age"` shows the age effect as curvature about a zero trend and puts the drift in the period and cohort effects; `"none"` plots the raw, un-gauged effects. It is display-only (the fitted rates and predictions are unchanged) and is ignored for models that are not full APC. See `effects.apc`.
`combined`	logical. For heterogeneity models, if `TRUE` plot the full effect (smooth + iid heterogeneity); default `FALSE` plots the smooth component only. Ignored for models without heterogeneity.
`...`	Additional arguments will be ignored

Details

Plot of age, period and cohort effects from apc objects. If covariates have been used for period/cohort, a second plot with covariate, absolute effect and relative effect is created. Absolute effect is relative effect times covariate.

Value

plot

Examples

```
## Not run:
data(apc)
model <- bamp(cases, population, age="rw1", period="rw1", cohort="rw1", periods_per_agegroup = 5)
plot(model)

## End(Not run)
```

`predict_apc` *Prediction for age-period-cohort models*

Description

Prediction of rates and, if possible, cases from the Bayesian age-period-cohort model using the prior assumptions (random walks) of the model and the estimated variance of the random walk. For example, random walk of first order (rw1) for period effect predicts constant effects for future periods plus noise.

Usage

```
predict_apc(
  object,
  periods = 0,
  population = NULL,
  quantiles = c(0.05, 0.5, 0.95),
  update = FALSE
)
```

Arguments

`object`	apc object
`periods`	number of periods to predict
`population`	matrix of (predicted) population, if NULL, population data from original bamp call will be used
`quantiles`	vector of quantiles to compute
`update`	boolean. If TRUE, object will be returned with results added to the object

Details

This function will return predicted rates for future periods. For this, future period and cohort effects will be predicted. Further age group effects will not be predicted. The rates are random samples from the predictive distribution; number of samples is equal to number of MCMC iterations. Quantiles will be provided for convenience, but all samples are available. If population numbers are given, number of cases will also be predicted. Number of cases will not only be predicted for future periods, but also for the time periods where data are available; this can be used for model assessment.

Value

list with quantiles of predicted probabilities (pr), predicted cases (cases) and predicted cases per period (cases_period) and a list samples with MCMC samples of pr, cases and cases_period. If update=TRUE, the apc object will be returned with this list (predicted) added.

See Also

vignette("prediction", package = "bamp")

Examples

```
## Not run:
data(apc)
model <- bamp(cases, population, age="rw1", period="rw1", cohort="rw1", periods_per_agegroup = 5)
pred <- predict_apc(model, periods=1)
plot(pred$pr[2,11,], main="Predicted rate per agegroup", ylab="p")

## End(Not run)
```

---

print.apc *Print apc objects*

Description

Print apc objects

Usage

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

Arguments

x	apc object
...	additional arguments will be ignored

Value

print

Examples

```
## Not run:
data(apc)
model <- bamp(cases, population, age="rw1", period="rw1", cohort="rw1", periods_per_agegroup = 5)
print(model)

## End(Not run)
```

`selectModel` *Automatic model selection for age-period-cohort models*

Description

Searches over age-period-cohort model specifications and returns the one best supported by the data, by Deviance Information Criterion (DIC). It answers the practical questions "is a first- or second-order random walk more appropriate for each effect?", "are the data overdispersed?" and (optionally) "is extra heterogeneity warranted?" without the user fitting every model by hand.

Usage

```
selectModel(
  cases,
  population,
  periods_per_agegroup,
  age = NULL,
  period = NULL,
  cohort = NULL,
  overdispersion = NULL,
  try_heterogeneity = FALSE,
  dic_margin = 4,
  psrf_tol = 1.1,
 screen = list(number_of_iterations = 10000, burn_in = 5000, step = 5, tuning = 200),
  final = "auto",
  refit = TRUE,
 hyperpar = list(age = c(1, 0.5), period = c(1, 5e-04), cohort = c(1, 5e-04), overdisp =
   c(1, 0.05), age_het = c(1, 0.05), period_het = c(1, 0.05), cohort_het = c(1, 0.05)),
  parallel = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`cases`	number of cases (matrix, periods x age groups), as in `bamp`.
`population`	population number, as in `bamp`.
`periods_per_agegroup`	periods per age group.
`age, period, cohort`	optional fixed value for an effect (`"rw1"`, `"rw2"`, `"rw1+het"`, `"rw2+het"` or `" "` for absent). If `NULL` (default) the effect is present and its random-walk order is searched over `"rw1"/"rw2"`.
`overdispersion`	optional fixed logical. If `NULL` (default), whether to include overdispersion is part of the search.

try_heterogeneity
: logical; if TRUE the search may also add heterogeneity ("+het") to an effect. Default FALSE.

dic_margin
: minimum DIC improvement required to adopt a more complex model (parsimony threshold). Default 4 (a conventional "clearly better" DIC difference).

psrf_tol
: convergence tolerance: a fit counts as converged if its maximum fitted-value Gelman-Rubin statistic is at or below this. Default 1.1.

screen
: list of MCMC settings used for the comparison fits, kept moderate for speed (default 10000 iterations, 5000 burn-in, step 5). Passed as mcmc.options. If no candidate converges at this length the search warns and selects by DIC only; increase these settings and re-run.

final
: list of MCMC settings used to refit the selected model, or "auto" to use the data-adaptive default of bamp.

refit
: logical; if TRUE (default) refit the selected model with the final settings and return it; if FALSE return the screening fit of the selected model.

hyperpar
: hyper-parameter list passed to bamp; defaults include heterogeneity hyper-parameters so "+het" models can be fitted.

parallel
: passed to bamp (chains run in parallel).

verbose
: logical; if TRUE (default) report progress and the running best model.

...
: further arguments passed to bamp (e.g. prior_scale, pg_engine).

Details

The search is a **greedy forward selection by complexity**. It starts from the simplest model – a first-order random walk ("rw1") for every effect that is present, no overdispersion – and at each round considers every candidate that is exactly one step more complex than the current best: an effect upgraded from "rw1" to "rw2", overdispersion switched on, or (if try_heterogeneity = TRUE) heterogeneity added to an effect. All candidates in a round are fitted and the one with the lowest DIC is adopted, but **only if** it improves DIC by at least dic_margin *and* its chains converged; otherwise the search stops. This costs a handful of fits rather than the full grid, follows an interpretable path, and – through the margin – prefers the simpler model unless the data clearly favour the more complex one. Each distinct specification is fitted at most once (results are cached).

Model comparison uses DIC (lower is better), which rewards fit and penalises effective complexity (pD); see bamp. A specification that did not converge is never selected, because a low DIC from a chain that has not mixed is not trustworthy – convergence is judged via the same criterion as checkConvergence: the maximum Gelman-Rubin statistic over the smoothing precisions of the effects present and the fitted log-odds across Lexis cells must be at or below psrf_tol.

For speed and fairness all candidates are fitted with the same short screen MCMC settings; the selected model is then optionally refitted (refit = TRUE) with the longer final settings before being returned. Fitting uses method = "pg", which is robust on the sparse, rare-event data where the legacy Taylor sampler can fail to converge.

Pin an axis to exclude it from the search by passing a fixed value: e.g. age = "rw2" fixes the age effect (it is not searched), age = " " removes the age effect entirely, and overdispersion = FALSE forbids overdispersion. Any axis left NULL is searched.

Value

A list (class "apcselect") with elements

- table: a data frame of every specification fitted, with its DIC, effective number of parameters pD, mean deviance, convergence flag, maximum fitted-value PSRF and fit time, ordered by DIC.
- best: the selected specification (a named list).
- model: the fitted apc object for the selected specification (refitted with final settings if refit = TRUE).
- path: the sequence of specifications adopted by the greedy search.

See Also

bamp, checkConvergence

Examples

```
## Not run:
data(apc)
sel <- selectModel(cases, population, periods_per_agegroup = 5)
sel$table          # ranked comparison of the models tried
sel$best           # the chosen specification
plot(sel$model)    # the refitted best model

## End(Not run)
```

Index

∗ **datasets**
apc.data, 3

apc, 2, *7*, *17*
apc.data, 3
apcSimulate, 3

bamp, *2*, 4, *8*, *9*, *15–17*

cases *(*apc.data*)*, 3
checkConvergence, 8, *16*, *17*
coh, 9
cov_c *(*apc.data*)*, 3
cov_p *(*apc.data*)*, 3

effects.apc, *2*, *7*, 10, *12*

mclapply, *6*

plot.apc, *2*, *7*, 12
population *(*apc.data*)*, 3
predict_apc, *7*, *11*, 13
print *(*print.apc*)*, 14
print.apc, *2*, *7*, 14

selectModel, 15