

KernelMix reference manual

Nilson Chapagain

Contents

1	KernelMix	2
1.1	Install	2
1.2	Example	3
1.3	Documentation	4
1.4	References and license	4
2	Model and interpretation	4
2.1	Continuous outcomes	4
2.2	Binary outcomes	5
2.3	Count outcomes	5
2.4	Survival outcomes	6
2.5	Random intercepts	6
2.6	Priors and exposure selection	7
2.7	Inputs and computation	7
2.8	Prediction and contrasts	8
2.9	Diagnostics and scope	8
3	An exposure-mixture analysis	9
3.1	Define the data	9
3.2	Fit and inspect a continuous outcome	9
3.3	Predict at exposure profiles	10
3.4	Summarize exposure associations	10
3.5	Fit a binary outcome	10
3.6	Fit counts and include observation time	11
3.7	Fit right-censored survival times	11
3.8	Account for clustered observations	13
3.9	Plot the results	13
3.10	Save and load a fit	13
4	Ozone and weather in New York	14
4.1	Data and question	14
4.2	Model and run	14
4.3	Results and checks	16
4.4	Reuse and limits	19
5	Structured models and Gaussian process approximations	19
5.1	Choose an exposure kernel	19
5.2	Reduce GP matrix size	20
5.3	Model random intercepts and slopes	21
5.4	Separate Gaussian variance and correlation	22
5.5	Encode delayed entry and competing events	23
5.6	Record left or interval censoring	23
5.7	Supply known covariate paths	24
5.8	Set starting values	24

6	API reference	25
6.1	Fitting	25
6.2	Structured model results	28
6.3	Fit results and diagnostics	28
6.4	Prediction	29
6.5	Cluster and survival results	29
6.6	Exposure contrasts and curves	30
6.7	Plotting	32
6.8	Kernel helper	32
6.9	Method sources	32
7	Numerical evidence and limits	32
7.1	Fitted R comparisons	32
7.2	Independent extension checks	33
7.3	Original studies	33
7.4	Platform evidence	34

1 KernelMix

KernelMix fits Bayesian kernel machine regression (BKMR) for exposure mixtures in Python. It models how an outcome varies with several exposures together, allowing curved associations and interactions between exposures. Other measured variables enter as linear adjustment terms.

To handle different outcome types, the package combines this exposure function with continuous, binary, count or survival models. NumPy and SciPy handle fitting and prediction; R is not required.

Task	Start here
Choose an outcome model and priors	Models and assumptions
Fit, inspect, predict and compare exposures	Analysis tutorial
Model clusters, kernels or detailed survival records	Structured models
Find arguments and result fields	API reference
Check numerical evidence and limitations	Evidence

1.1 Install

Install directly from GitHub:

```
python -m pip install "kernelmix[plots] @ git+https://github.com/nilson01/kernelmix.git@main"
```

Or, from a downloaded source folder:

```
python -m pip install ".[plots]"
```

Python 3.10 or later is required. Omit `[plots]` if plots are not needed.

1.2 Example

```
import numpy as np
from kernelmix import fit, univariate_response, plot_response

rng = np.random.default_rng(17)
Z = rng.normal(size=(24, 2))
y = np.sin(Z[:, 0]) + 0.3 * Z[:, 1] + rng.normal(0, 0.5, 24)
result = fit(y, Z, draws=200, warmup=200, chains=2, seed=17)
print(result.summary())
curve = univariate_response(result, variable=0, points=25, seed=17)
ax = plot_response(curve)
```

Example output (first rows):

```
BKMRFit(family='gaussian', n=24, exposures=2, chains=2, draws=200)
95% pointwise credible intervals
Parameter      Mean      SD      Lower      Upper
lambda         16.93    9.047    4.145    38.61
sigma2          0.2159   0.08346  0.1142   0.4188
```

These short chains illustrate the output and trigger mixing warnings. For an analysis, use longer runs and inspect agreement between chains and effective sample sizes, which account for dependence between draws.

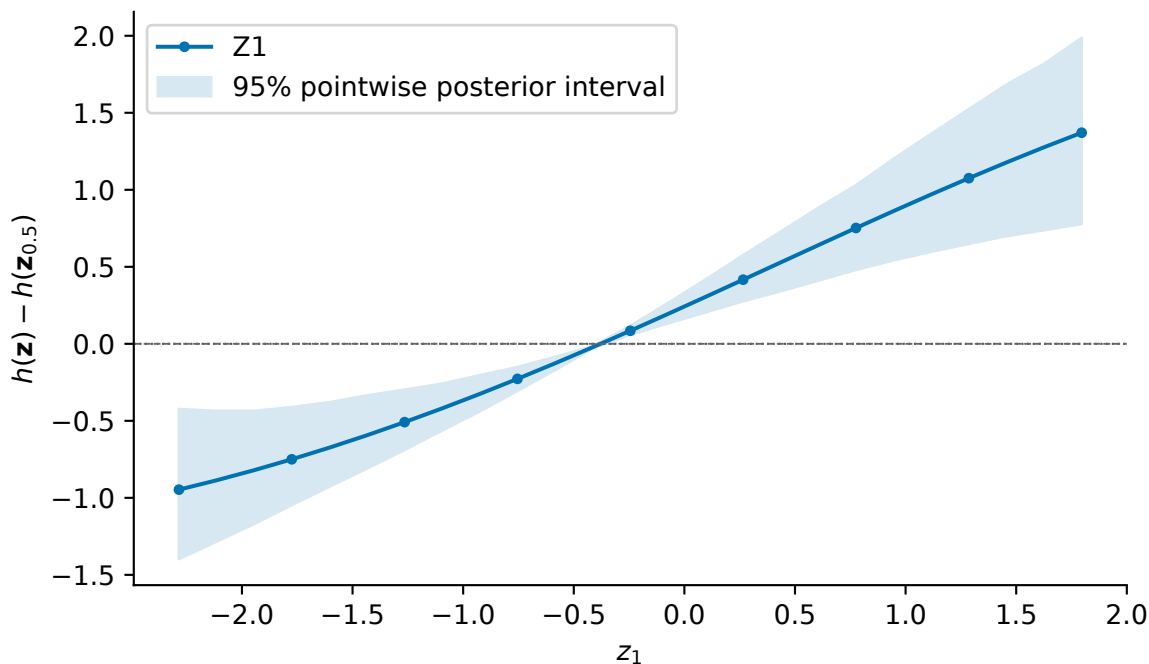


Figure 1: Exposure response curve with pointwise 95% credible intervals

The curve varies the first exposure with the second at its median, relative to the all-median exposure profile. It describes a fitted association; a causal interpretation requires additional assumptions.

1.3 Documentation

Browse the searchable documentation.

The common questions explain prediction scales, missing values, selection and diagnostics. The observed ozone example includes source data and saved results. Download the reference manual for offline reading. Contributors can find development checks separately.

1.4 References and license

Bobb et al. (2015) introduced BKMR for exposure-mixture analysis. Bobb et al. (2018) described the R software and its binary-outcome extension. KernelMix implements these methods separately in Python and adds count and survival likelihoods, defined in the model guide.

We compare selected Gaussian and probit posterior summaries and conditional calculations, including random intercepts, with installed R `bkmr` 0.2.2. The evidence summary records results and limits. See NOTICE for attribution and LICENSE for GPL-2.0-only terms.

2 Model and interpretation

2.1 Continuous outcomes

For a continuous outcome, we separate the joint exposure contribution from linear covariate adjustment. Let Y_i be the outcome for observation i , $\mathbf{z}_i$ its M exposures, and $\mathbf{x}_i$ its adjustment covariates. Following Bobb et al. (2015), the function $h(\mathbf{z}_i)$ describes the joint exposure contribution, while $\mathbf{x}_i^\top \boldsymbol{\beta}$ provides linear adjustment:

$$Y_i = h(\mathbf{z}_i) + \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \quad \epsilon_i \stackrel{\text{iid}}{\sim} N(0, \sigma^2).$$

The error ϵ_i is independent normal noise with variance σ^2 . The function h allows curved associations and interactions, where the association with one exposure depends on another.

To relate values of h at different exposure profiles (rows of exposure values), we use a Gaussian process (GP) prior. A prior specifies their distribution before using outcome data. At the observed profiles,

$$\mathbf{h} = (h_1, \dots, h_n)^\top \sim N(\mathbf{0}, \tau \mathbf{K}_{Z, \mathbf{r}}), \quad K(\mathbf{z}, \mathbf{z}'; \mathbf{r}) = \exp \left\{ - \sum_{m=1}^M r_m (z_m - z'_m)^2 \right\}.$$

Here $h_i = h(\mathbf{z}_i)$, n is the observation count, τ controls prior variation in h , and $\mathbf{K}_{Z, \mathbf{r}}$ contains similarities between observed profiles. Each relevance parameter $r_m \geq 0$ controls how exposure m changes similarity; $r_m = 0$ removes it from h .

The default Gaussian sampler integrates out $\mathbf{h}$: it accounts for its distribution without sampling each value during parameter fitting. It stores $\lambda = \tau/\sigma^2$ as `lambda`, giving covariance $\sigma^2(\mathbf{I} + \lambda \mathbf{K}_{Z, \mathbf{r}})$, where $\mathbf{I}$ is the identity matrix.

The table maps the model notation to API names. Check the meaning of `lambda` when changing fitting routes: some routes store a variance ratio, while others store the GP amplitude directly.

Paper quantity	Meaning	API name
$\mathbf{z}_i, \mathbf{x}_i$	Exposure and adjustment rows	Rows of Z and X
h, β	Exposure function and adjustment coefficients	predict_h, beta
σ^2, r_m	Residual variance and exposure relevance	sigma2, r
τ	GP covariance amplitude	Default Gaussian/probit: lambda stores τ/σ^2 ; structured and default count/survival: it stores τ directly

The structured guide defines its separate priors. Equal numerical values of **lambda** need not describe the same covariance across these routes.

2.2 Binary outcomes

For a binary outcome, the exposure function and linear adjustment instead determine a probability. With `family="binomial"`, the probit link gives

$$\Pr(Y_i = 1 \mid h, \beta) = \Phi\{h(\mathbf{z}_i) + \mathbf{x}_i^\top \beta\}.$$

Here Φ is the standard normal cumulative distribution function. The binary outcome is determined by whether an underlying normal variable exceeds zero; its residual variance is fixed at one. This extension follows Bobb et al. (2018). Probability contrasts depend on exposures and adjustment covariates and are not odds ratios.

Default probit fits use a flat coefficient prior. They reject complete or quasi-complete separation: adjustment covariates separate the outcome classes, possibly with observations on the separating boundary. Under this prior, separation can give an infinite integral for the unnormalized posterior density, so no posterior probability distribution can be obtained. Structured fits use a proper normal coefficient prior, which integrates to one.

2.3 Count outcomes

For counts, the same exposure and adjustment contributions enter the log mean. A random intercept $u_{c(i)}$ can also account for observation i 's cluster $c(i)$; it is zero without clustering. For Poisson and negative-binomial (NB2) models, define

$$\eta_i = h(\mathbf{z}_i) + \mathbf{x}_i^\top \beta + u_{c(i)}, \quad \mu_i = \exp(\eta_i + o_i).$$

The known log offset o_i accounts for observation time or exposure amount. Conditional on the predictors and cluster effect, the Poisson mean and variance are both μ_i . NB2 allows extra variation:

$$E(Y_i \mid \mu_i, \phi) = \mu_i, \quad \text{Var}(Y_i \mid \mu_i, \phi) = \mu_i + \mu_i^2/\phi, \quad \phi > 0.$$

dispersion stores ϕ . Larger values approach Poisson variation. These likelihoods follow the standard Poisson and NB2 parameterizations. R **bkmr** 0.2.2 does not implement these count likelihoods.

2.4 Survival outcomes

For event times, the predictor η_i determines a relative hazard rather than a count mean. Using η_i above without a count offset, the model is

$$\lambda_i(t) = \lambda_0(t) \exp(\eta_i), \quad S_i(t) = \exp\{-\Lambda_0(t) \exp(\eta_i)\}.$$

A hazard is the instantaneous event rate among subjects still event-free. Here $\lambda_0(t)$ is the baseline hazard, $\Lambda_0(t)$ its integral, and $S_i(t)$ the probability of remaining event-free through t .

Each record supplies a positive observed time t_i and an indicator d_i : one for an observed event, zero when follow-up ends first (right censoring). Under independent right censoring, its likelihood contribution is $\{\lambda_i(t_i)\}^{d_i} S_i(t_i)$. We use constant baseline rates λ_{0j} within right-closed intervals. Let D_{ij} be subject i 's time in interval j , $j(i)$ its event interval, and d_j the event count in interval j . Then

$$\ell = \sum_i d_i \{\log \lambda_{0,j(i)} + \eta_i\} - \sum_i e^{\eta_i} \sum_j D_{ij} \lambda_{0j}.$$

An event at a cut belongs to the interval on its left. Every event then has positive time at risk in its event interval. With independent $\lambda_{0j} \sim \text{Gamma}(a_0, b_0)$ priors, the conditional update is

$$\lambda_{0j} \mid \dots \sim \text{Gamma}\left(a_0 + d_j, b_0 + \sum_i D_{ij} e^{\eta_i}\right).$$

This is a full piecewise-exponential event-time likelihood, not Cox partial likelihood. See the INLA manual, section 3.15 for the model form and the Stan survival guide for censoring contributions. KernelMix uses its own independent gamma baseline prior. Once selected, the time cutpoints are held fixed. The model does not include uncertainty from choosing them. Defaults use event-time deciles, so assess sensitivity to their number and placement. Baseline priors depend on the chosen time unit.

The likelihood above describes the default right-censored model. The structured survival interface also supports delayed entry, competing causes, predictor histories and left/interval censoring. All-censored data are accepted with a warning: the posterior then depends strongly on the priors. An intercept and the baseline hazard can compensate for each other without changing the fitted hazard. Their separate estimates therefore depend on the priors; `intercept=False` avoids adding the redundant intercept.

2.5 Random intercepts

Observations in the same cluster may share variation beyond the exposure function and adjustment covariates. A random intercept represents this shared contribution. In the default fitting route, $c(i)$ identifies observation i 's cluster and $u_g \sim N(0, \sigma_u^2)$ independently across clusters. Conditional on h , covariates and these intercepts, observations are independent under the outcome likelihood. The `cluster` labels identify observations; `groups` instead groups exposures for selection.

For default Gaussian/probit fits, `lambda_cluster` is σ_u^2/σ^2 , with probit $\sigma^2 = 1$. The marginalized covariance is

$$\sigma^2 \{\mathbf{I} + \lambda \mathbf{K}_{Z,\mathbf{r}} + \lambda_u \mathbf{J}\}, \quad J_{ij} = 1\{c(i) = c(j)\}.$$

For count/survival fits there is no residual Gaussian variance parameter: `lambda` and `lambda_cluster` are the GP and random-intercept variances directly. The random-intercept model follows the Gaussian/probit extension described by Bobb et al. (2018). KernelMix also combines these intercepts with count and survival likelihoods.

For prediction, known labels use fitted intercepts, while equal unseen labels share one new Gaussian draw. Omitting labels fixes $u = 0$, rather than averaging over cluster effects. Random intercepts do not provide arbitrary serial correlation or cluster-robust inference. Constant kernels or clusters with one observation may leave little information to distinguish GP variation from cluster variation.

2.6 Priors and exposure selection

To fit the model, we specify prior distributions for its unknown parameters. Default Gaussian and probit fits use the priors in R `bkmr` 0.2.2; structured fits use different priors. Gamma distributions below use shape and **rate**. The inverse-gamma density is proportional to $(\sigma^2)^{-a-1} \exp(-b/\sigma^2)$ for shape a and parameter b .

Quantity	Default
λ and λ_u , when clustered	Gamma(1, 0.1)
σ^2 , continuous outcome	Inverse-gamma(0.001, 0.001)
Active r_m	$1/r_m \sim \text{Uniform}(0, 100)$
Inclusion probability π	Beta(1,1)
β	Flat prior

Count/survival `fit(..., prior=None)` instead uses $\lambda, \lambda_u \sim \text{Gamma}(4, 4)$, independent $\beta \sim N(\mathbf{0}, 25\mathbf{I})$, $\phi \sim \text{Gamma}(2, 0.1)$ for NB2, and $\lambda_{0j} \sim \text{Gamma}(0.1, 0.1)$ for survival. At fixed finite predictor values, the resulting prior distributions of the count mean and relative hazard have finite means and second moments. A `Prior` object supplies all prior settings. A dictionary changes only the supplied fields and keeps the selected family's other defaults, for example `prior={"r_distribution": "gamma"}`. Warnings identify supplied amplitude or dispersion priors for which these moment guarantees do not hold. Assess prior sensitivity; large finite moments can still be numerically unstable. `sigma2=1` is a storage convention for count/survival fits and is omitted from their summaries.

Under the inverse-uniform relevance prior, r_m has no finite upper bound. A finite average of sampled values does not establish that its posterior mean exists. Inspect posterior quantiles and chain mixing, and compare results under alternative priors. `Prior(r_distribution="gamma")` and `"unif"` provide alternatives. Gamma parameters use mean and standard deviation; uniform bounds apply to r_m for `"unif"` and $1/r_m$ for `"invunif"`.

With component selection, $\delta_m \sim \text{Bernoulli}(\pi)$ indicates whether $r_m > 0$. Excluded exposures have $r_m = 0$. Group selection uses the groups S_g and indicator ω_g from Bobb et al. (2015): each active group contains **one** selected exposure, chosen uniformly within that group. KernelMix does not offer unequal within-group prior probabilities. A group PIP is its posterior inclusion probability; a conditional PIP is an exposure's inclusion probability given that its group is active. A never-selected group has an undefined conditional PIP. PIPs depend on the model and priors; they do not measure causal importance.

2.7 Inputs and computation

`fit` adds an intercept and standardizes exposures by default. Standardization uses the retained training mean and sample standard deviation. Prediction reuses these values. Pass `intercept=False` or `standardize=False` to disable either. Do not include a constant adjustment column when adding an intercept. R `bkmr` does not automatically apply these Python preprocessing defaults.

`missing="raise"` rejects missing data. `missing="drop"` removes rows jointly across required inputs, including applicable cluster labels, offsets and survival inputs. For recorded subject histories, it removes the whole affected history; see structured survival. Retained original row indices are stored in `result.data["rows"]`. Complete-case fitting does not correct bias caused by missing data. Covariates must be numeric; categorical encoding is the caller's responsibility.

After preprocessing, each chain samples the posterior distribution, which combines data and priors. Chains run sequentially with separate random streams. `draws` counts saved draws after `warmup`, and `thin` sets their spacing. Proposal scales stay fixed during sampling. The result's `sampling_info` records acceptance counts and runtime.

The sampler depends on the outcome. Default Gaussian fits integrate out h during parameter sampling. Binary fits also sample underlying normal outcomes constrained by the observed response signs; prediction uses these saved values.

Default count/survival fits sample a noncentered GP with elliptical slice sampling, log-scale Metropolis updates and conjugate baseline updates. Eigenvalues at most $100n\epsilon \max(1, \lambda_{\max}(\mathbf{K}))$, where ϵ is machine precision, are treated as zero in numerical calculations. Eigenvalues measure variation along matrix directions; very small ones cannot be distinguished reliably from zero at machine precision. No small constant is added to the kernel diagonal (diagonal

jitter). Prediction uses the same rank rule and the saved latent function. The sampler also saves standardized auxiliary variables. The eigenvectors used to represent them may have different signs on different platforms while representing the same function. Archive validation therefore checks the represented matrix directions without requiring identical eigenvector signs.

Structured fits instead sample the GP and other normal-prior coordinates jointly by elliptical slice sampling, with log-scale Metropolis updates for amplitudes, relevance, dispersion and survival baseline rates. Gaussian residual variance has a conditional inverse-gamma update. Their priors and covariance options are defined in the structured guide.

Dense matrix factorization requires roughly cubic work in n per parameter update: doubling the observation count can multiply this work by about eight. Working-memory checks reject oversized fitting or prediction requests; they are protective estimates, not a promise about peak process memory.

2.8 Prediction and contrasts

The fitted model can return either the exposure contribution or the full outcome prediction. `predict_h` draws h jointly across profiles. `predict(kind="mean")` returns Gaussian/count means, binary probabilities or survival relative hazards. `predict_survival` returns survival probabilities or hazards on a time grid. `predict(kind="observation")` adds new outcome variation for Gaussian, binary and response-scale count predictions; survival observation sampling is unavailable. The binary latent scale is the underlying normal scale.

To compare exposure profiles, contrasts subtract **joint** posterior draws, retaining covariance between predictions. **overall_effect** compares all-exposure quantile profiles with a reference; **single_exposure_effects** changes one exposure with others fixed; **interaction_effects** compares two single-exposure contrasts. Curves and surfaces vary one or two exposures with others fixed.

Equal-tailed intervals leave the same posterior probability below and above their limits. They apply to individual points, not the whole curve simultaneously. Response-scale summaries fix adjustment covariates at retained training means unless **X** supplies another profile.

A quantile profile combines a percentile chosen separately for each exposure. That combination may be uncommon when exposures are correlated. Check whether observed exposure rows are close to the requested combinations before interpreting contrasts. Causal interpretation requires additional identification assumptions; fitting a model does not supply them.

2.9 Diagnostics and scope

`diagnostics()` reports rank-normalized/folded split $\hat{R}$, bulk effective sample size and tail effective sample size. $\hat{R}$ compares sampled distributions across parts of the chains; values near one are desirable. Effective sample size accounts for dependence between draws, in the main part (bulk) or extremes (tails) of the distribution. Large $\hat{R}$, small effective sample sizes, or constant sampled traces need investigation. Fixed parameters are excluded from these diagnostics. Trace plots and sensitivity checks remain necessary; favorable diagnostics do not prove convergence.

KernelMix supports Gaussian, probit, Poisson, NB2 and right-censored piecewise-exponential survival models with independent Gaussian random intercepts. The structured interface adds alternative kernels, finite GP bases, random slopes, Gaussian residual correlation and variance regression. It also supports delayed entry, competing causes, known covariate paths and left/interval censoring, with separate proper priors and a direct GP amplitude. See verified results and open gaps for the limits of the numerical checks and original-study reproduction.

3 An exposure-mixture analysis

This tutorial starts with a continuous outcome and then changes the outcome model for binary, count and survival data. The models share an exposure-response function h , which describes joint exposure associations, including curves and interactions. Other measured variables enter as linear adjustment terms.

Run the Python blocks in order after installation. The data are constructed and the chains are deliberately short, so the examples demonstrate the calls and outputs. For a research analysis, use longer chains, inspect sampling diagnostics and assess how results change under reasonable prior distributions.

3.1 Define the data

Each row is an observation. Columns of **Z** are exposures; columns of **X** are adjustment covariates. Keep **Z** in its original units. The fit stores the training means and sample standard deviations and reuses them for prediction.

```
import numpy as np
import kernelmix as km

t = np.linspace(-1.0, 1.0, 12)
Z = np.column_stack((1.5 + t, 2.0 + np.sin(3 * t), 10.0 + np.cos(5 * t)))
X = t[:, None]
y = 0.3 + np.sin(2 * t) + 0.2 * t**2
names = ["Exposure A", "Exposure B", "Exposure C"]
```

The default adds an intercept, so **X** should not include a constant column. Missing values raise an error. Use `missing="drop"` only when analysis of complete rows is appropriate. It drops a row if its outcome, exposure or adjustment value is missing. Dropping rows does not correct bias caused by missing data.

3.2 Fit and inspect a continuous outcome

We first fit the continuous outcome. With component selection, an exposure can enter or leave h . Its posterior inclusion probability (PIP) is the fraction of saved draws in which it is included.

```
gaussian = km.fit(
    y, Z, X,
    family="gaussian", selection="component",
    exposure_names=names, covariate_names=["Adjustment"],
    draws=12, warmup=12, chains=2, seed=19,
)
print(gaussian.summary())
parameter_rows = gaussian.summary_table()
pip_rows = gaussian.inclusion_probabilities()
checks = gaussian.diagnostics()
```

The summary reports means, standard deviations and credible intervals from the posterior, which combines data and priors. A 95% credible interval contains 95% of the posterior probability for the reported quantity.

Before interpreting the summary, inspect split R-hat and effective sample size. R-hat compares chain segments, with values near one desirable; effective sample size accounts for dependence between draws. These diagnostics assess sampling, not model validity. PIPs also depend on the selection priors and do not measure causal importance.

3.3 Predict at exposure profiles

To predict, supply new exposure profiles (rows of exposure values) in the training column order and original units. When adjustments were fitted, also supply raw **X** columns without an intercept.

```
Znew = Z[:3].copy()
Xnew = X[:3].copy()
mean_prediction = gaussian.predict(Znew, Xnew, seed=20)
observation_prediction = gaussian.predict(
    Znew, Xnew, kind="observation", seed=20,
)
h_prediction = gaussian.predict_h(Znew, seed=20)
prediction_rows = mean_prediction.to_records()
```

Mean predictions exclude new outcome noise; observation predictions include it. `predict_h` isolates h , excluding the intercept and linear adjustment. Each saved draw contains predictions for all rows together, preserving their covariance. Results store these joint draws and row-specific intervals. Omitting new data uses retained training rows and adjustments.

3.4 Summarize exposure associations

To summarize an exposure association, compare predictions at specified profiles. An overall contrast changes every exposure to a chosen percentile of its own distribution; a single-exposure contrast changes one with others fixed. These conditional associations need additional identification assumptions for a causal interpretation.

```
overall = km.overall_effect(gaussian, reference=0.5, seed=21)
single = km.single_exposure_effects(gaussian, seed=21)
interaction = km.interaction_effects(
    gaussian, variables=["Exposure A", "Exposure B"], seed=21,
)
curve = km.univariate_response(
    gaussian, variable="Exposure A", points=7, seed=21,
)
surface = km.bivariate_response(
    gaussian, variables=("Exposure A", "Exposure B"), points=4, seed=21,
)
overall_rows = overall.to_records()
```

By default, these summaries compare h . Curves and surfaces vary one or two exposures, hold others at their medians, and subtract the all-median profile. Zero means no change from that reference; `center=False` disables subtraction. Intervals are pointwise, not simultaneous bands for a whole curve.

Joint draws retain covariance between compared profiles, so identical profiles have exactly zero contrast. A combination of marginal percentiles may be rare in correlated data; check its distance from observed profiles before interpreting it.

3.5 Fit a binary outcome

For binary data, retain the exposure function and linear adjustment but use `family="binomial"`. The probit link converts the underlying normal-scale predictor to a probability. Outcomes must be zero or one.

```

y_binary = np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1])
binary = km.fit(
    y_binary, Z, X,
    family="binomial", exposure_names=names,
    covariate_names=["Adjustment"],
    draws=12, warmup=12, chains=2, seed=22,
)
probability_prediction = binary.predict(Znew, Xnew, seed=23)
probability_contrast = km.overall_effect(
    binary, scale="response", X=[0.0], seed=23,
)

```

Mean predictions are probabilities; observation predictions are zero or one. A response-scale contrast subtracts joint predicted probabilities and is not an odds ratio. In these summaries, **X** is one raw adjustment vector fixed across profiles; omitting it uses retained training means. Latent-scale summaries compare h alone. Because they exclude covariate adjustment, supplying **X** raises an error.

3.6 Fit counts and include observation time

For counts, the predictor instead determines a log mean. Poisson assumes conditional variance equals the mean; negative binomial allows extra variation. A known log observation time or exposure amount enters as **offset**, with coefficient one.

```

counts = np.array([0, 1, 0, 2, 1, 3, 1, 2, 4, 3, 5, 2])
observation_time = np.array([1, 1, 2, 1, 2, 1, 1, 3, 1, 2, 1, 1])
count_fit = km.fit(
    counts, Z, family="negative_binomial", offset=np.log(observation_time),
    draws=12, warmup=12, chains=2, seed=24,
)
count_mean = count_fit.predict(Z[:3], offset=np.log(observation_time[:3]), seed=25)
count_observation = count_fit.predict(
    Z[:3], offset=np.log(observation_time[:3]), kind="observation", seed=25,
)

```

count_mean gives expected counts at the supplied observation times; observation predictions give integer counts. Use **family="poisson"** for Poisson. For NB2, **dispersion** controls the extra conditional variation beyond the mean; exposure and cluster effects enter that mean.

For response-scale exposure summaries, the log offset is zero, corresponding to one unit of observation time or exposure amount. Latent summaries instead compare h on the log-mean scale. The model guide gives the proper coefficient priors used for count models.

3.7 Fit right-censored survival times

For event-time data, the predictor determines a relative hazard. Supply a positive observed time and **event**: 1 for an event, 0 when follow-up ends first. The proportional-hazards model multiplies a piecewise-constant baseline hazard by a predictor-dependent factor. A hazard is an event rate among subjects still at risk.

```

observed_time = np.array([1.0, 1.4, 2.0, 2.6, 3.0, 3.8, 4.0, 4.7, 5.2, 6.0, 6.4, 7.0])
event = np.array([1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0])
survival_fit = km.fit(
    observed_time, Z, family="survival", event=event,
    baseline_breaks=[2.0, 4.0], intercept=False,
    draws=12, warmup=12, chains=2, seed=26,
)
survival = survival_fit.predict_survival(
    Z[:2], times=np.linspace(0, 7, 30), seed=27,
)
survival_ax = km.plot_survival(survival)
log_hr = km.overall_effect(survival_fit, quantiles=[0.75], reference=0.25, seed=27)
hr_draws = np.exp(log_hr.draws)
hr_interval = np.quantile(hr_draws, [0.025, 0.975], axis=(0, 1))

```

The baseline intervals are $(0, 2]$, $(2, 4]$ and $(4, \infty)$ in this example. An event exactly at a cut belongs to the interval on its left. The model assumes independent right censoring and proportional hazards. For delayed entry, competing causes or known covariate paths, use the structured interface. Beyond observed follow-up, predictions extend the final constant baseline hazard. Such extrapolation needs scientific justification.

`predict_survival` returns survival probabilities, or hazards with `kind="hazard"`. Ordinary `predict` returns relative hazards for this family. Response-scale exposure contrasts therefore subtract relative hazards; they do not subtract survival probabilities. For hazard ratios, exponentiate each latent `h` contrast before summarizing it. Because the intercept and baseline hazard can compensate for each other, this example omits the intercept.

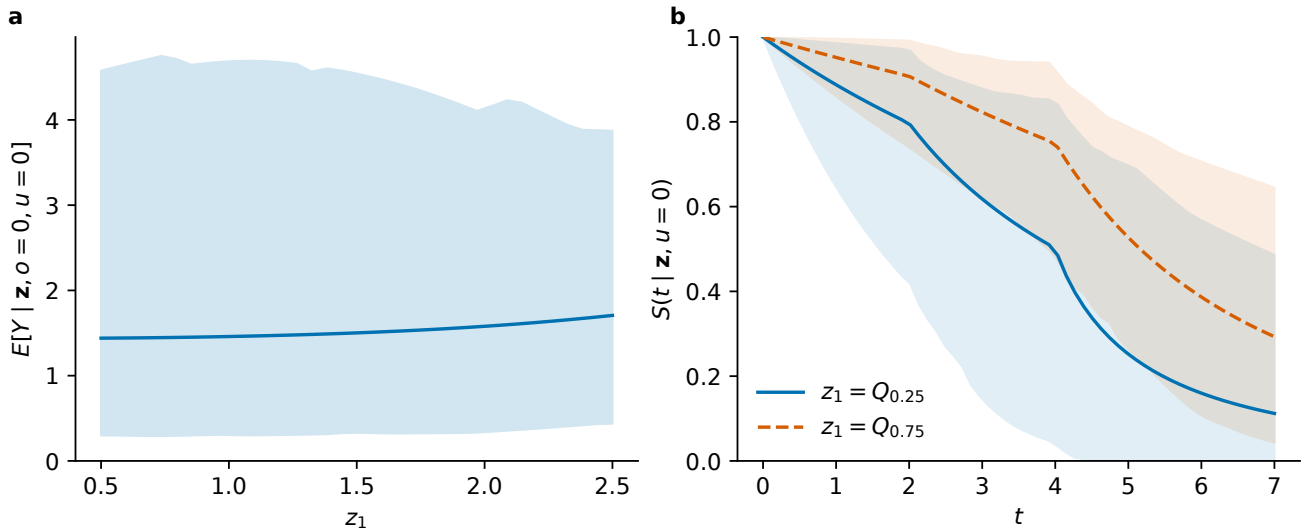


Figure 2: Count response and survival probabilities. (a) Negative-binomial mean counts as z_1 varies, with offset zero. (b) Survival probabilities at the first exposure's 25th and 75th percentiles. Other exposures are fixed at their medians and $u = 0$. Shading shows pointwise 95% credible intervals.

The figure uses the data above, 200 warmup and 200 saved draws per chain, two chains, and the same seeds. Predictions use 100 draws per chain. These short fits have mixing warnings and illustrate the outputs; they do not support inference or show whether the intervals achieve their stated coverage across repeated datasets. Figure values.

3.8 Account for clustered observations

Clustering can be modeled with any of the five outcome families. Supply `cluster` labels to identify observations that share a Gaussian random intercept. These labels group observations; exposure-selection `groups` instead groups exposure columns.

```
cluster = ["A"] * 3 + ["B"] * 3 + ["C"] * 3 + ["D"] * 3
clustered_fit = km.fit(
    counts, Z, family="poisson", cluster=cluster,
    draws=12, warmup=12, chains=2, seed=28,
)
intercepts = clustered_fit.random_intercepts(seed=29)
cluster_prediction = clustered_fit.predict(
    Z[:3], cluster_new=["A", "new", "new"], seed=29,
)
```

For a known cluster, prediction uses its fitted intercept. Rows with the same new label share one newly sampled intercept in each draw. Omitting `cluster_new`, or requesting exposure summaries, sets the intercept to zero; it does not average predictions over clusters. For slopes or Gaussian residual correlation, see structured models.

3.9 Plot the results

Plotting requires Matplotlib ([plots]). Each function returns an axes object; the caller controls display and export.

```
trace_ax = km.plot_trace(gaussian)
pip_ax = km.plot_pips(gaussian)
curve_ax = km.plot_response(curve)
overall_ax = km.plot_overall(overall)
surface_ax = km.plot_surface(surface)

# Optional: display the figures in an interactive session.
# import matplotlib.pyplot as plt
# plt.show()
```

Curve and overall plots show pointwise intervals. Surface plots show posterior means; result objects retain interval limits. Plot exposures separately when units differ. Traces exclude warmup.

3.10 Save and load a fit

A fit includes its retained training data as well as saved draws. Store and share it with the same care as the input data. The archive uses a non-pickle format.

```
from pathlib import Path
from tempfile import TemporaryDirectory

with TemporaryDirectory() as directory:
    path = Path(directory) / "mixture-fit.npz"
    gaussian.save(path)
    restored = km.BKMRFit.load(path)
    restored_prediction = restored.predict(Znew, Xnew, seed=20)
```

The API reference gives defaults, result fields and the definitions of pairwise interactions and median-reference centering. Method sources are Bobb et al. (2015) and the 2018 software and probit extension.

4 Ozone and weather in New York

We examine how daily ozone varies with solar radiation, wind and temperature, adjusting for month. This exploratory analysis uses the public New York air-quality data, separately from the applications in Bobb et al. (2015). Source data, saved results and sampling checks accompany the example.

4.1 Data and question

The R airquality data contain 153 daily records from New York, May–September 1973. Ozone was measured at Roosevelt Island, solar radiation at Central Park, and wind and temperature at LaGuardia Airport. Their measurement times also differ. These are daily environmental measurements, not personal exposures.

Variable	Role	Unit
Ozone	Outcome, transformed with the natural logarithm	Parts per billion
Solar.R	Exposure function	Langleys
Wind	Exposure function	Miles per hour
Temp	Exposure function	Degrees Fahrenheit
Month	Linear adjustment, May as reference	Four indicators

There are 37 missing ozone values and seven missing solar-radiation values. Keeping only rows with all required values retains 111 days and excludes 42. No missing values are filled in. The source CSV preserves all 153 records. Its provenance records the R version, hash, units and excluded row numbers.

Month	Recorded days	Complete days
May	31	24
June	30	9
July	31	26
August	31	23
September	30	29

Ozone medians are 31 ppb in complete rows and 35 ppb in the five rows missing solar radiation. Missingness is uneven across months, and ozone is unknown on 37 days. These observed ozone values do not show whether excluding incomplete days leaves the analysis unbiased.

4.2 Model and run

We use the Gaussian BKMR model with $Y_i = \log(\text{Ozone}_i)$, $\mathbf{z}_i$ containing solar radiation, wind and temperature, and $\mathbf{x}_i$ containing an intercept and four month indicators:

$$Y_i = h(\mathbf{z}_i) + \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \quad \epsilon_i \stackrel{\text{iid}}{\sim} N(0, \sigma^2).$$

We standardize each exposure using the mean and sample standard deviation of the 111 retained days. Month indicators describe broad seasonality; independent errors remain a working assumption for this daily series.

For the Gaussian kernel, independent $r_m \sim \text{Gamma}(1,1)$ priors use shape and rate one. Residual variance has an inverse-gamma(2,1) prior. We retain the default flat prior for adjustment coefficients and $\lambda = \tau/\sigma^2 \sim \text{Gamma}(1,0.1)$. Here τ is the GP covariance amplitude and σ^2 the residual variance. Exposure selection is disabled.

From the source folder, with KernelMix and its plotting extra installed:

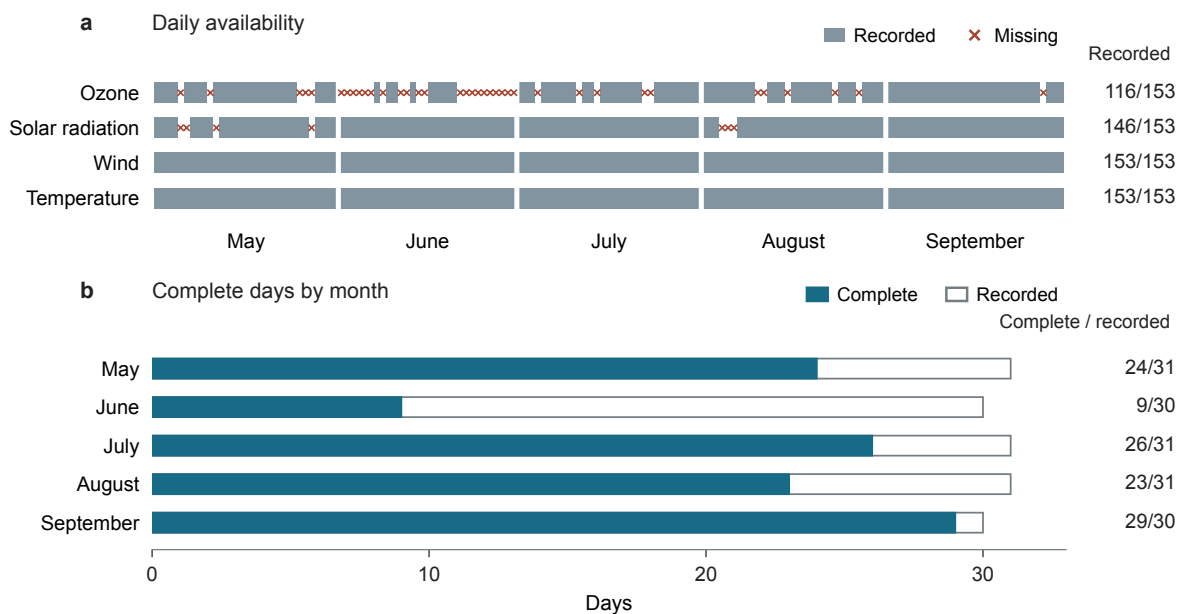


Figure 3: Measurement availability among all 153 daily records. (a) Filled cells mark recorded measurements; crosses mark missing values. Fractions give recorded counts out of 153 days. (b) Filled bars show complete days within outlines for all recorded days, with complete/recorded counts. All four measurements are available on 111 days.

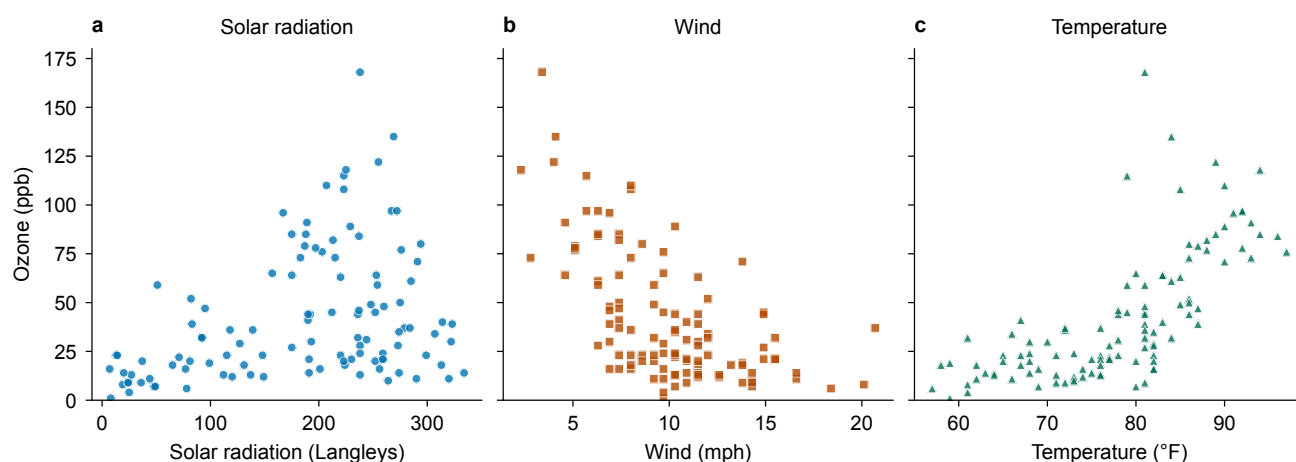


Figure 4: Observed ozone against solar radiation, wind and temperature for the 111 complete days. Each point is one day, without smoothing or adjustment for month.

```
python tools/analyze_airquality.py --draws 20000 --warmup 5000 --output ozone-results
python tools/plot_airquality.py --results ozone-results --output ozone-figure
```

Four chains run sequentially with seed 73. Each has 5,000 warmup iterations and 20,000 saved draws. An initial 4,000-draw run had maximum $\hat{R} = 1.0142$; one longer run was made to improve sampling, with the model, prior and data unchanged. This is an exploratory workflow, not a prospectively fixed study.

4.3 Results and checks

Check	Recorded result
Maximum rank/folded split $\hat{R}$	1.0053
Minimum bulk effective sample size	885
Minimum tail effective sample size	1,193
Consecutive-day residual pairs	92
Correlation of consecutive-day fitted residuals	0.0136
Fitting time on the recorded local environment	72.6 seconds

Sampling diagnostics use all saved draws for ten parameters. R-hat compares chains; effective sample size accounts for dependence between draws. These parameter diagnostics do not establish convergence of every estimated curve. Curve and fitted-residual summaries use 400 evenly spaced draws per chain.

Residual checks use training data, not held-out observations. The consecutive-day residual correlation cannot establish independent errors.

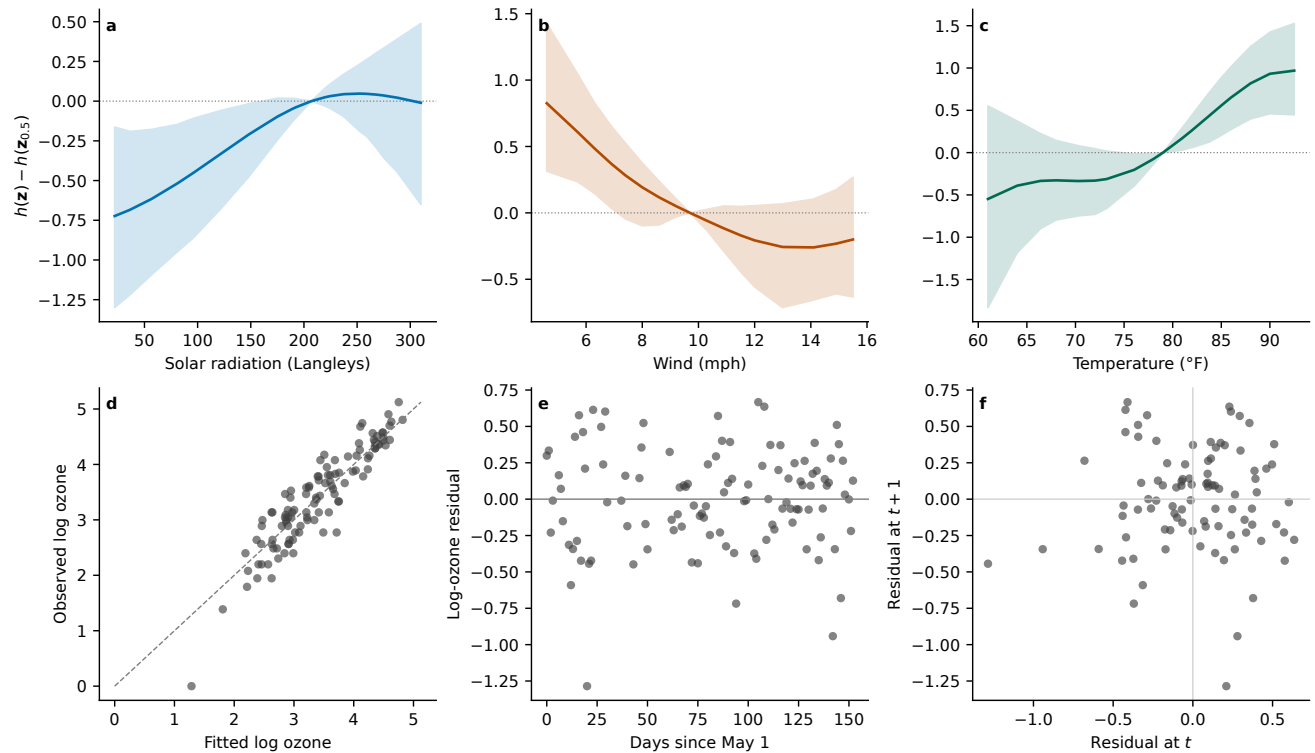


Figure 5: Ozone example: (a–c) posterior means and 95% pointwise credible intervals for $h(\mathbf{z}) - h(\mathbf{z}_{0.5})$, with other exposures fixed at their sample medians and $\mathbf{z}_{0.5}$ the all-median profile; (d) fitted and observed log ozone; (e) residuals by date, with missing dates unconnected; (f) residual pairs one calendar day apart. Intervals assume independent Gaussian working errors.

To isolate a conditional weather association, each curve varies one exposure with the others at their medians. The mean curve generally rises with temperature and falls with wind; the solar-radiation curve is nonlinear. On the

log scale, exponentiating a difference gives a ratio of fitted ozone medians at fixed weather and month values, not an arithmetic mean difference.

Each grid spans the 5th–95th percentiles, with duplicate values removed. Combining a grid value with the other exposures’ medians may produce a profile uncommon in the observed data. The curve table records its distance to the nearest observed profile after standardization; the distance plot below shows how close these combinations are to the data.

4.3.1 Parameter estimates

The kernel relevance parameters r_m describe how exposure differences change the covariance of h . The month coefficients β instead describe differences from May on the log-ozone scale. Points show posterior means; bars show 95% equal-tailed credible intervals.

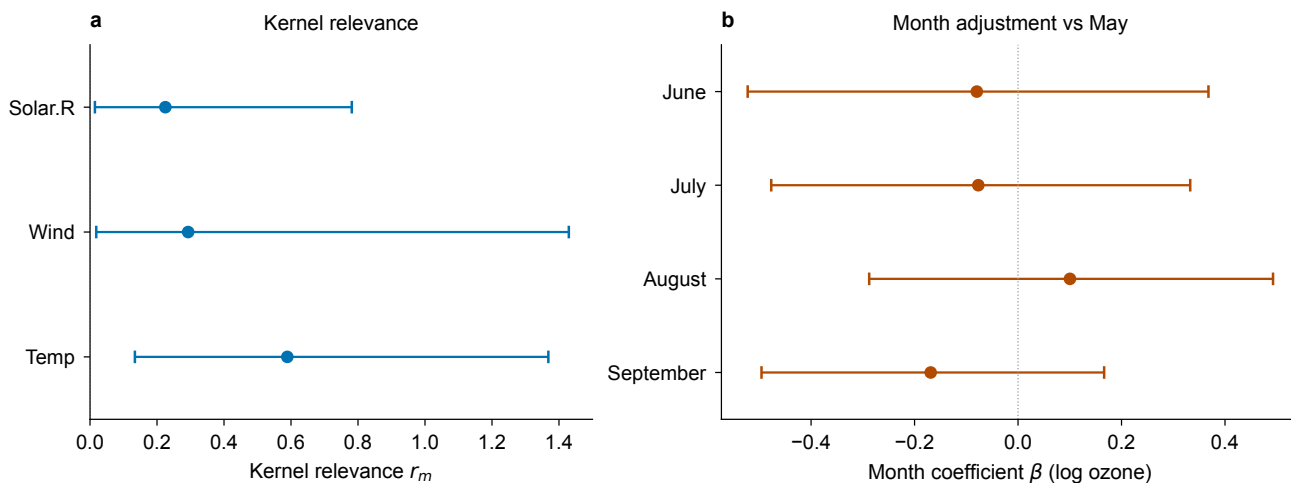


Figure 6: Posterior means and 95% equal-tailed credible intervals. (a) Exposure relevance r_m . (b) Month coefficients in β , relative to May, on the log-ozone scale. The intercept is omitted. Relevance does not measure causal importance.

4.3.2 Sampling checks

To assess the sampling behind these intervals, we show diagnostics for all ten parameters. Bulk effective sample size concerns the center of the posterior; tail effective sample size concerns its tails.

4.3.3 Distance from observed exposure combinations

Sampling checks do not show whether a curve profile occurs in the data. We therefore measure its distance to the nearest observed profile, using training standard deviations to put exposures on a common scale. No distance cutoff establishes prediction reliability.

The plotting script regenerates these five additional figures from saved tables without refitting:

```
python tools/plot_airquality_details.py
```

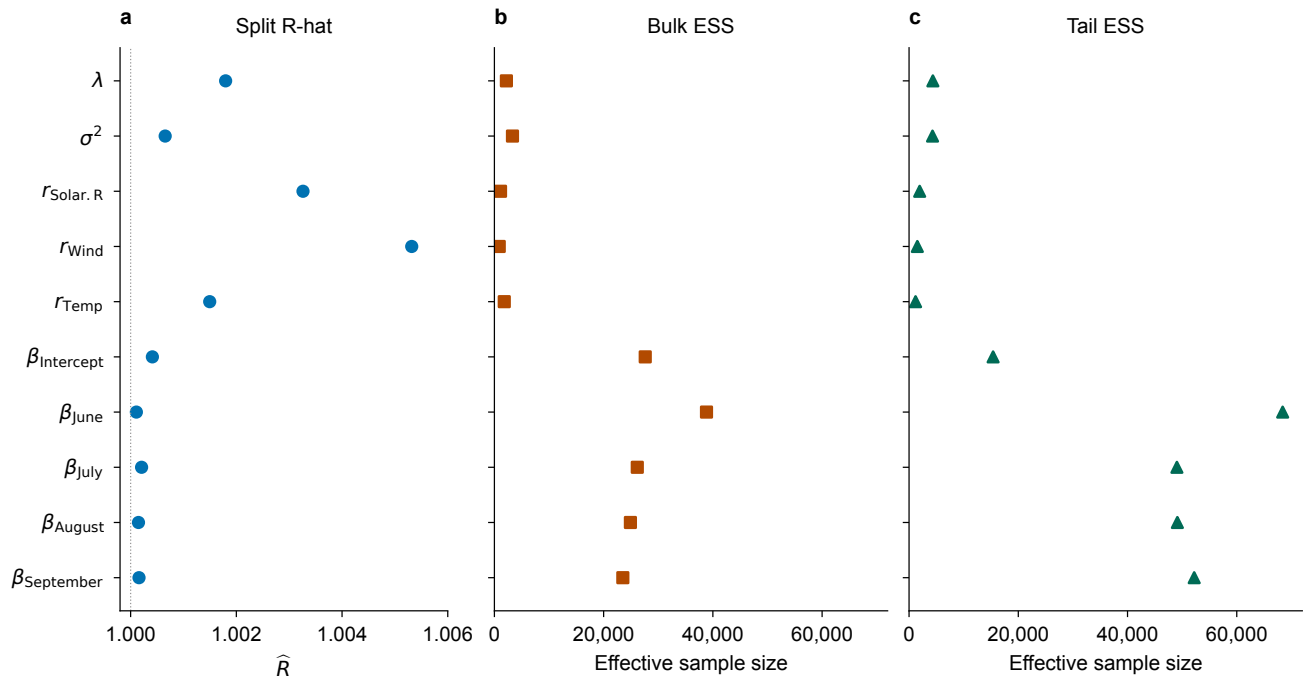


Figure 7: Sampling diagnostics from four chains, each with 20,000 saved draws. (a) Rank-normalized/folded split $\hat{R}$. (b) Bulk effective sample size. (c) Tail effective sample size. These checks do not establish convergence or validate the model.

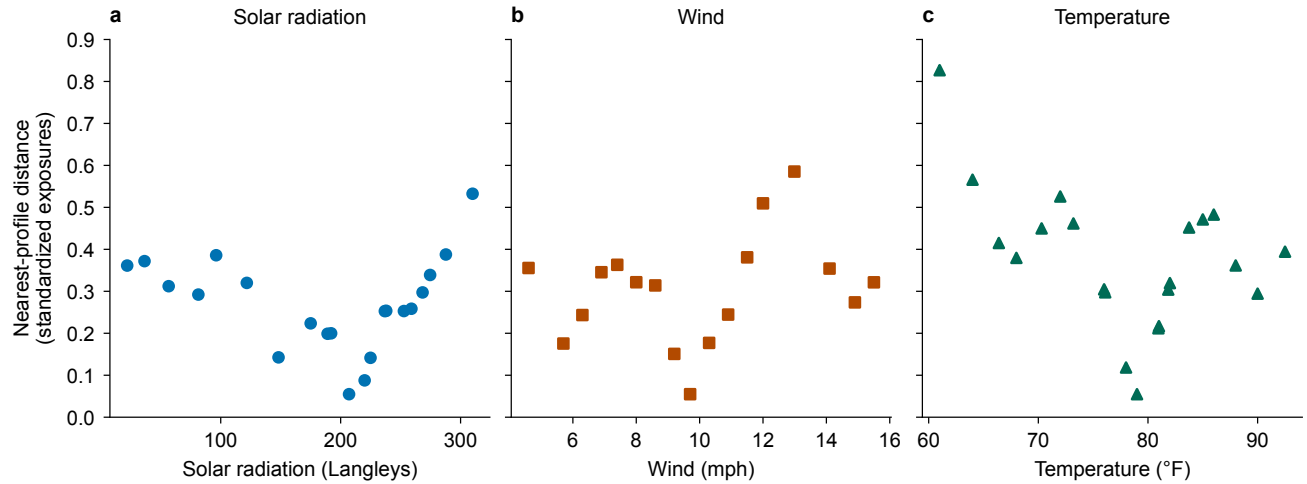


Figure 8: Nearest observed-combination distance along each saved curve grid, after standardizing exposures. Other exposures remain at their medians. There are 21 solar-radiation, 17 wind and 21 temperature locations; tied percentile values were removed. Distance describes data proximity, not model validity.

4.4 Reuse and limits

The results record, parameter summaries, diagnostics, curve values and dated observations/residuals accompany the runnable scripts. R is not required to run the analysis. The source CSV was exported with R 4.5.0.

Missingness, measurement differences, priors and serial dependence limit interpretation. This example establishes neither causal effects nor time-series interval coverage. Original-study reproduction and external research use remain separate evidence gaps; see reference studies.

5 Structured models and Gaussian process approximations

The model guide introduces the exposure function and outcome distribution, following Bobb et al. (2015) and the 2018 extension. A Gaussian process (GP) prior relates exposure-function values at different exposure profiles.

This guide explains how to change that kernel, reduce the cost of its calculations, model dependence between observations, or describe more detailed event-time records. These options use the **StructuredFit** result class and the separate priors defined below.

Run the blocks in order. Eight observations and four warmup/saved draws per chain illustrate the interface. For inference, use longer chains, sampling diagnostics, prior-sensitivity checks and exposure profiles supported by the data.

```
import numpy as np
import kernelmix as km

x = np.linspace(-1.0, 1.0, 8)
visit = np.tile(np.arange(4), 2)
cluster = np.array(["a" * 4 + "b" * 4])
Z = np.column_stack((2 + x, 5 + np.sin(np.arange(8))))
X = np.cos(np.arange(8))[:, None]
y = 0.3 + x + 0.2 * np.sin(2 * np.arange(8))
slope = x[:, None]
V = (x - x.mean())[:, None]
demo = dict(draws=4, warmup=4, chains=2, seed=47)
```

Supply exposures and knots in original units. Prediction reuses training standardization. Adjustment and random-slope inputs exclude the added intercept.

5.1 Choose an exposure kernel

The kernel relates exposure profiles (rows of exposure values). For profiles $\mathbf{z}$ and $\mathbf{z}'$ with M exposures and relevance parameters r_m , define

$$q(\mathbf{z}, \mathbf{z}') = \sum_{m=1}^M r_m (z_m - z'_m)^2, \quad r_m \geq 0.$$

The profiles use the fitted standardization when enabled. Setting $r_m = 0$ removes exposure m from the kernel.

kernel	Kernel at weighted squared distance q
"gaussian"	$\exp(-q)$
"matern32"	$(1 + \sqrt{6q}) \exp(-\sqrt{6q})$
"rational_quadratic"	$(1 + q/\alpha)^{-\alpha}$

The fixed $\alpha > 0$ (`kernel_alpha`) defaults to one; changing it requires `kernel="rational_quadratic"`. As α increases, this kernel approaches the Gaussian kernel. These definitions follow GPML, Chapter 4 with KernelMix relevance parameters.

```
matern = km.fit(y, Z, X, kernel="matern32", **demo)
rq = km.fit(y, Z, X, kernel="rational_quadratic", kernel_alpha=1.5, **demo)
curve = km.univariate_response(matern, variable=0, points=5, seed=48)
```

Kernel choice also affects the fitting route and its priors. For exposure-function values $\mathbf{h}$ at observed profiles, structured fits store `lambda` as the direct GP amplitude τ in $\mathbf{h} \sim N(\mathbf{0}, \tau \mathbf{K})$, including Gaussian fits. Default Gaussian fits store τ/σ^2 . Compare priors on these stated scales. Exposure contrasts remain conditional associations, requiring additional assumptions for causal interpretation.

5.1.1 Structured priors

Quantity	Default
Adjustment coefficients, including the intercept	Independent $N(0, 25)$; SD set by <code>prior.beta_sd</code>
Direct GP amplitude τ	Gamma(shape 4, rate 4)
Gaussian residual variance σ^2	Inverse-gamma(shape 2, rate 1)
Random-coefficient covariance	Defined in the random-effects section

A `prior` dictionary changes only the supplied fields; a `Prior` object supplies every field. Check `fit.settings["prior"]` when comparing models. The `cluster_lambda_mean` and `cluster_lambda_sd` fields apply to default random intercepts, not structured random coefficients.

5.2 Reduce GP matrix size

After choosing how the kernel relates exposure profiles, choose how to represent the GP in computation. `"dense"` uses the full kernel. `"pp"` represents h through values at a smaller set of exposure profiles (knots). `"rff"` uses fixed sine/cosine features to approximate the kernel.

```
pp = km.fit(y, Z, X, kernel="matern32", approximation="pp",
            n_components=4, **demo)
rff = km.fit(y, Z, X, kernel="matern32", approximation="rff",
            n_components=8, **demo)
pp_prediction = pp.predict(Z[:2], X[:2], seed=48)
```

For the predictive process (PP), `n_components` caps the number of automatically chosen knots, which are spread across observed exposure profiles; explicit knots can instead be supplied as `knots=Z[[0, 3, 5, 7]]`. Its covariance is $\tau \mathbf{K}_{ZU} \mathbf{K}_{UU}^+ \mathbf{K}_{UZ}$, where U denotes the knots and $+$ is the numerical pseudoinverse, an inverse restricted to matrix directions distinguishable from zero. This representation omits the full GP's remaining variation after its knot values are known. Uncertainty can therefore be much smaller away from knots; this is the projected model described by Banerjee and colleagues, not an exact dense-GP uncertainty calculation.

For random Fourier features (RFF), `n_components` counts frequency pairs, giving twice as many features. Paired sine/cosine features follow Rahimi and Recht. Frequencies are Gaussian for the Gaussian kernel, multivariate Student-t with three degrees of freedom for Matérn, and a Gaussian/Gamma scale mixture for RQ. Each mixture scale is shared across all coordinates of one frequency vector. Base draws stay fixed throughout sampling and are saved with the fit. Inference treats this saved set of features as fixed. It does not include uncertainty from choosing a different feature set, and a fixed feature count does not establish approximation accuracy.

During fitting, PP and RFF store matrices with one row per observation and one column per feature, together with square matrices whose size is the feature count. Joint predictions and exposure contrasts still return dense matrices over the requested prediction rows; keep these requests small or use suitable batches. Large correlated Gaussian residual blocks can still require dense calculations. Dense/PP eigensystems treat eigenvalues at most $100 * \text{dimension} * \text{machine_epsilon} * \text{largest_eigenvalue}$ as zero for numerical calculations. Eigenvalues measure variation along matrix directions; this threshold removes directions too small to distinguish reliably from zero. Saved dense training values and PP knot values preserve prediction coordinates across reloads.

5.3 Model random intercepts and slopes

For repeated observations, random coefficients describe variation between clusters beyond the shared exposure function. Supply `cluster` labels to identify the observation groups and `random_slopes` to add slopes after the random intercept. All five outcome families support these coefficients. The `groups` argument instead groups exposures for selection.

```
slopes = km.fit(y, Z, X, cluster=cluster, random_slopes=slope, **demo)
coefficients = slopes.random_effects(["a", "new"], seed=49)
intercepts = slopes.random_intercepts(["a", "new"], seed=49)
prediction = slopes.predict(
    Z[:2], X[:2], cluster_new=["a", "new"], random_slopes_new=slope[:2], seed=49,
)
fixed_slopes = km.fit(
    y, Z, X, cluster=cluster, random_slopes=slope,
    random_covariance=[[0.5, 0.1], [0.1, 0.3]], **demo,
)
```

`random_effects()` returns joint coefficient draws; `predict` combines them with the new slope design. A fixed covariance must be positive definite (every nonzero linear combination has positive variance) and match `[intercept, slopes]`. Otherwise, the full covariance is estimated.

The random-coefficient prior separates standard deviations and correlations. Each SD is `random_scale * Phi(raw)`, with standard-normal `raw` and cumulative distribution function `Phi`, giving a uniform SD between zero and `random_scale` (default two). Correlations come from a normalized lower-triangular matrix with normal off-diagonal entries. This proper prior is neither LKJ nor uniform over correlation matrices.

Prediction uses saved coefficients for known clusters. Each distinct new label receives one coefficient vector per draw, shared across its rows. Omitting labels sets random effects to zero. With a nonlinear link, this differs from averaging predictions over new clusters. Supply `random_slopes_new` for new slope rows. Redundant design columns are rejected; few clusters or little slope variation can still make covariance estimates sensitive to priors.

Count/probit inputs retain their usual meanings. For example, an NB2 model can use the same slope workflow:

```
counts = np.array([0, 1, 2, 1, 0, 2, 1, 3])
nb = km.fit(counts, Z, X, family="negative_binomial", cluster=cluster,
    random_slopes=slope, approximation="rff", n_components=8, **demo)
count_means = nb.predict(Z[:2], X[:2], seed=50)
```

For counts, `offset` still supplies the logarithm of observation time or exposure amount. Response predictions are count means or, with `kind="observation"`, sampled counts. Response-scale exposure contrasts use zero offset and zero random effects.

5.4 Separate Gaussian variance and correlation

Random coefficients describe variation shared within a cluster. A Gaussian residual model describes the remaining variation and can allow residual variances to differ between observations or residuals to be correlated within a cluster. With `variance_design=V`, $\text{Var}(\epsilon_i \mid \sigma^2, \gamma) = \sigma^2 \exp(\mathbf{v}_i^T \gamma)$, where $\mathbf{v}_i$ is row i of V and $\gamma \sim N(\mathbf{0}, \mathbf{I})$. No intercept is added. Columns must be linearly independent, and no combination may give a nonzero constant column: `sigma2` already sets the baseline variance. This parameterization is distinct from `nlme`'s variance function.

```
ar = km.fit(
    y, Z, X, cluster=cluster, variance_design=V,
    correlation="ar1", observation_time=visit, correlation_parameter=0.4, **demo,
)
future_rows = np.array([3, 7])
future_mean = ar.predict(
    Z[future_rows], X[future_rows], cluster_new=["a", "b"],
    observation_time_new=[4, 4], variance_design_new=V[future_rows], seed=51,
)
residual_variance = ar.predict_residual_variance(
    Z[future_rows], variance_design_new=V[future_rows],
)
exponential = km.fit(
    y, Z, X, cluster=cluster, correlation="exponential",
    observation_time=visit / 2, **demo,
)
```

Residual correlation requires Gaussian outcomes and clusters. Within each cluster, AR1 times must be distinct integers and exponential times distinct finite values. AR1 uses `rho**gap`, with $-1 < \text{rho} < 1$; exponential uses `exp(-gap/range)`, with positive range. See `nlme`'s AR1 definition and exponential definition. A supplied `correlation_parameter` is fixed. Otherwise, AR1 uses `rho=tanh(raw)` and exponential uses `range=exp(raw)`, each with a standard-normal prior on `raw`.

For known clusters, new residuals may be correlated with training residuals. Mean prediction uses that relationship to predict the new residual component, and observation prediction includes uncertainty in it. Supply original `cluster_new` labels to condition training-row predictions; their times and slope rows are reused. Unseen or omitted labels have no training residual cross-correlation. `predict_h` and exposure contrasts do not use training residuals to predict that component.

A fixed correlation matrix must cover all original rows, have unit diagonal, be positive definite, and be zero between clusters. Supply new-by-training and new-by-new correlations to condition new residuals:

```
R = np.where(cluster[:, None] == cluster[None, :],
             0.3 ** np.abs(visit[:, None] - visit[None, :]), 0.0)
fixed = km.fit(y, Z, X, cluster=cluster, correlation=R, **demo)
cross = np.array([
    np.where(cluster == label, 0.3 ** (4 - visit), 0.0)
    for label in ("a", "b")
])
fixed_prediction = fixed.predict(
    Z[future_rows], X[future_rows], cluster_new=["a", "b"],
    correlation_cross=cross, correlation_new=np.eye(2), seed=52,
)
```

With fixed correlation and no custom cross matrix, new residuals are independent of training residuals. The combined training/new correlation must be positive semidefinite: no linear combination may have negative variance. Residual variance predictions exclude uncertainty in the GP and random coefficients. Random effects and long-range residual correlation can be hard to distinguish with sparse repeated measurements.

5.5 Encode delayed entry and competing events

For survival data, the likelihood must account for when follow-up starts and what is known about the event. Delayed entry starts follow-up after time zero and includes only subjects event-free until entry. Competing causes identify different first-event types. `y` records an exact event time, right-censoring time or interval lower bound; `event=0` means right censoring, and `1..n_causes` identifies a cause. Only the first terminal event is modeled, excluding recurrent events and general state transitions. Baseline intervals are right closed: an event exactly at a cut belongs to the interval on its left.

```
time = np.array([0.8, 1.4, 2.2, 3.2, 1.1, 1.9, 2.8, 3.6])
event = np.array([1, 0, 2, 1, 0, 2, 0, 1])
entry = np.array([0.0, 0.2, 0.5, 0.4, 0.1, 0.3, 0.6, 0.2])
survival = km.fit(
    time, Z, X, family="survival", event=event, entry=entry, n_causes=2,
    baseline_breaks=[1.5, 2.5], intercept=False, **demo,
)
times = np.array([0.0, 1.0, 2.0, 3.0, 4.0])
s = survival.predict_survival(Z[:2], X[:2], times=times, entry=0.2, seed=53)
cif = survival.predict_cumulative_incidence(Z[:2], X[:2], times=times, entry=0.2, seed=53)
cause_two = survival.predict_cumulative_incidence(
    Z[:2], X[:2], times=times, cause=2, entry=0.2, seed=53,
)
hazard_two = survival.for_cause(2).predict_survival(
    Z[:2], X[:2], times=times, kind="hazard", entry=0.2, seed=53,
)
cause_contrast = km.overall_effect(survival.for_cause(2), seed=53)
```

For cause k , the hazard is $\lambda_k(t | \mathbf{z}_i, \mathbf{x}_i) = \lambda_{0k}(t) \exp\{h_k(\mathbf{z}_i) + \mathbf{x}_i^T \boldsymbol{\beta}_k\}$, with random-coefficient terms added when supplied. Cause-specific functions and parameters have independent priors conditional on shared exposure relevance and selection. Posterior dependence can remain through shared parameters and interval-censored likelihoods.

For competing risks, survival means no event of any cause; cumulative incidence (CIF) gives the probability of a specified cause by a time. Joint draws satisfy survival plus the sum of CIFs equals one; see the R survival guide.

Survival response-scale contrasts subtract relative hazards; they are not hazard ratios or survival-probability differences. To select results, `cif.for_cause(2)` extracts one CIF, whereas a fit's `for_cause(2)` selects its exposure function, relative-hazard predictions and contrasts. `km.plot_survival(cause_two)` plots a CIF with pointwise intervals when Matplotlib is installed.

Prediction `entry` describes subjects known to be event-free through that time; before it, survival is one and CIFs are zero. Static-profile prediction extends the final baseline hazard beyond follow-up. Baseline cuts are held fixed; their selection uncertainty is excluded. An intercept and baseline hazard can compensate for each other. Use `intercept=False` unless separate, prior-dependent estimates are intended.

Delayed-entry fits describe subjects included after surviving to their recorded entry times, at their recorded predictor values. They do not recover risk among people who were never observed before entry, or correct for how survival to entry changes who is included. New-cluster predictions use this same model for included subjects.

5.6 Record left or interval censoring

When an event time is uncertain, specify a finite interval with `upper > y`. Left censoring uses lower bound and entry zero. Exact events use `upper == y` or omit `upper`; right censoring uses `event=0`, optionally with infinite upper bound. Unknown cause `event=-1` requires a finite interval; specify `n_causes` if all event causes are unknown. These are KernelMix codes, not R Surv status codes.

```

lower = np.array([0.0, 1.4, 1.8, 3.2, 1.1, 1.5, 2.8, 3.0])
upper = np.array([0.8, np.inf, 2.2, 3.2, np.inf, 1.9, np.inf, 3.6])
interval_event = np.array([1, 0, 2, 1, 0, -1, 0, 1])
interval = km.fit(
    lower, Z, X, family="survival", event=interval_event, upper=upper,
    entry=np.zeros(8), subject=np.arange(8), n_causes=2,
    baseline_breaks=[1.5, 2.5], intercept=False, **demo,
)

```

The likelihood assumes that censoring and inspection times provide no further information about event risk after accounting for modeled information. It also assumes that missing cause labels are ignorable after that information is considered. Summing over unknown causes does not model a recording process that depends on the cause itself. An interval row assumes its supplied predictors apply throughout the uncertain event interval.

5.7 Supply known covariate paths

When predictors change over time, use repeated `subject` labels to identify successive `(entry, y]` intervals for the same subject. Rows must be chronological, contiguous and nonoverlapping; only the last may have an event or finite censoring interval. A later row continues the same subject's history rather than representing entry into the cohort. `missing="drop"` removes a whole subject history if required values are missing, preserving the record structure but not correcting missing-data bias.

```

subject = np.repeat(np.arange(4), 2)
start = np.tile([0.0, 1.0], 4)
stop = np.tile([1.0, 3.0], 4)
path_event = np.array([0, 1, 0, 0, 0, 2, 0, 1])
path_fit = km.fit(
    stop, Z, X, family="survival", event=path_event, entry=start,
    subject=subject, n_causes=2, baseline_breaks=[1.5, 2.5],
    intercept=False, **demo,
)
path_inputs = dict(start=start, stop=stop, subject_new=subject,
    times=[0.0, 1.0, 2.0, 3.0], seed=54)
path_survival = path_fit.predict_survival_path(Z, X, **path_inputs)
path_cif = path_fit.predict_survival_path(
    Z, X, kind="cumulative_incidence", **path_inputs,
)
path_cause_two = path_fit.predict_survival_path(
    Z, X, kind="cumulative_incidence", cause=2, **path_inputs,
)

```

Paths must cover requested times; the final row is never extended. `kind="hazard"` returns cause-specific hazards. Output follows first occurrence of `subject_new`. Repeated cluster labels share a random vector across intervals.

Predictors must be known before their risk interval. Do not copy event-time or later measurements backward. Paths are neither interpolated nor forecast. Draws are joint across times and causes; credible intervals are pointwise.

5.8 Set starting values

Structured `initial` dictionaries accept `beta`, `gp_weight`, `r`, `delta`, `pi`, `lambda` and `sigma2`; NB2 adds `dispersion` and survival adds `baseline_hazard`. Omitted values receive defaults. With K causes (one for nonsurvival models; this dimension label is distinct from the kernel matrix $\mathbf{K}$), p adjustment coefficients including the intercept, and b retained basis coordinates:

Key	Required shape and restriction
<code>beta</code>	(K, p) ; a length- p vector is also accepted when $K = 1$
<code>gp_weight</code>	(K, b) ; a length- b vector is also accepted when $K = 1$
<code>r</code> , <code>delta</code>	One value per exposure; relevance and zero/one inclusion must agree
<code>lambda</code>	Positive scalar or length- K vector of direct amplitudes

Key	Required shape and restriction
<code>sigma2</code>	Positive scalar; fixed at one outside Gaussian models
<code>pi</code>	Inclusion probability; one without selection
<code>dispersion</code>	Positive scalar for NB2
<code>baseline_hazard</code>	Positive (K, J) array for J baseline intervals

Random coefficients, covariance coordinates, variance coefficients and inferred residual-correlation coordinates start at their internal defaults. Those internal sampling variables cannot be set through `initial`. Dense basis dimension is the retained observation count, PP dimension is the knot count, and RFF dimension is twice `n_components`; numerical null columns are retained. Omit `gp_weight` to use zero starting coordinates.

Saved structured arrays have leading `(chain, draw)` axes. They include cause-indexed `beta`, `lambda` and `gp_weight`, shared `r`, `delta` and `pi`, and Gaussian `sigma2`. Dense/PP fits retain coordinate-independent `gp_anchor` values; RFF fits retain their frozen feature map. Applicable models also save `random_coeff`, `random_covariance`, `gamma`, `correlation_parameter`, `dispersion` and `baseline_hazard`. Use result methods for prediction rather than interpreting internal coordinates as exposure effects.

6 API reference

Use `import kernelmix as km` and supply inputs as NumPy arrays; pandas is not required. The tutorial shows how the calls form an analysis. The model guide defines the equations and maps their symbols to API names.

An exposure profile is one row of exposure values. Each posterior draw is a set of parameter values sampled from the distribution given the data and priors. Prediction and contrast objects retain draws jointly across profiles, preserving their dependence.

6.1 Fitting

```
km.fit(
    y, Z, X=None, *, family="gaussian", draws=1000, warmup=1000,
    chains=4, thin=1, seed=None, standardize=True, intercept=True,
    selection="none", groups=None, prior=None, r_fixed=None, missing="raise",
    exposure_names=None, covariate_names=None, initial=None, proposal_scales=None,
    cluster=None, offset=None, event=None, baseline_breaks=None,
    kernel="gaussian", kernel_alpha=1, approximation="dense", n_components=32,
    knots=None, random_slopes=None, random_covariance=None, random_scale=2,
    correlation=None, observation_time=None, correlation_parameter=None,
    variance_design=None, entry=None, upper=None, subject=None, n_causes=None,
)
```

`fit` returns a `BKMRFit` object. Selecting a structured option returns its `StructuredFit` subclass; the structured-model guide defines the additional inputs, priors, array shapes and predictions.

Family	Required outcome inputs	Mean or probability
Gaussian	Numeric <code>y</code>	Continuous outcome mean
Binomial	<code>y</code> equal to 0 or 1	Probit probability
Poisson / negative binomial	Nonnegative integer <code>y</code> ; <code>offset</code> when needed	Count mean, including the log offset
Survival	Default: positive observed times and binary <code>event</code> ; structured: cause codes, entry and censoring bounds	Relative hazard from <code>predict</code> ; survival/hazard from <code>predict_survival</code>

`cluster` supplies one string or integer label per observation. Missing labels participate in `missing="drop"`. Integer 1 and string "1" are distinct. `offset` is a finite vector of log observation exposures for count families. `event` is required only for survival; `baseline_breaks` supplies internal positive, strictly increasing cuts. Omit cuts to use event-time deciles, or pass `[]` for a constant baseline hazard. Family-specific arguments are rejected elsewhere.

Input	Meaning
<code>y, Z, X</code>	Outcome vector, exposure matrix and optional adjustment matrix, with matching rows. Supply raw adjustment columns without an intercept when <code>intercept=True</code> .
<code>family</code>	"gaussian", "binomial" (probit), "poisson", "negative_binomial" (NB2) or "survival" (piecewise-exponential proportional hazards).
<code>draws, warmup, chains, thin</code>	Saved draws per chain, discarded iterations, sequential chains and spacing between saved draws. Total iterations per chain are <code>warmup + draws * thin</code> . At least two saved draws are required.
<code>seed</code>	Nonnegative integer or <code>None</code> ; uses local random generators without changing NumPy's global random state.
<code>standardize</code>	Center exposures and divide by retained-training sample standard deviations. Constant exposures are invalid when this is enabled.
<code>selection</code>	"none", "component" or "group". Group selection requires one <code>groups</code> label per exposure and permits at most one active exposure in each included group.
<code>prior</code>	<code>Prior</code> , a dictionary of its fields, or <code>None</code> for defaults.
<code>r_fixed</code>	Finite nonnegative vector fixing exposure relevance values; disallows variable selection.
<code>missing</code>	"raise" or joint complete-case "drop". Infinity remains invalid except for right-censoring upper bounds.
<code>Names</code>	Optional unique names for exposure and raw adjustment columns. Otherwise use input <code>DataFrame</code> columns, or generated names.
<code>initial</code>	Shared starting values; omitted entries use defaults. See initialization.
<code>proposal_scales</code>	Fixed positive log-scale proposal standard deviations (SDs), default 0.3. All routes accept <code>r</code> and <code>lambda</code> ; default clustered fits add <code>cluster_lambda</code> , NB2 adds <code>dispersion</code> , and structured survival adds <code>baseline_hazard</code> .

Adjustment columns must be linearly independent, with fewer columns than retained rows. Default probit fits

reject complete or quasi-complete separation, where covariates separate outcome classes, possibly with observations on the boundary. With the default flat coefficient prior, the unnormalized posterior density can then have an infinite integral. Structured fits use a proper normal prior.

For computation, dense fitting uses $O(n^2)$ memory and $O(n^3)$ matrix operations, where n is the retained observation count. Finite GP bases reduce this cost, but correlated residual blocks may remain dense.

6.1.1 Initialization

Fitting route	initial keys
Default Gaussian/probit	<code>beta</code> , <code>r</code> , <code>lambda</code> , <code>delta</code> , <code>pi</code> , <code>sigma2</code> ; probit adds <code>ystar</code>
Default count/survival	<code>beta</code> , <code>r</code> , <code>lambda</code> , <code>delta</code> , <code>pi</code> , <code>latent_white</code> ; NB2 adds <code>dispersion</code> , survival adds <code>baseline_hazard</code>
Default clustered fits	Add <code>lambda_cluster</code> ; count/survival also add <code>u_white</code>
Structured	Cause-indexed coefficients and <code>gp_weight</code> ; see shapes and restrictions

Inapplicable keys raise errors.

6.1.2 Prior configuration

```
km.Prior(
    r_distribution="invunif", r_mean=5.0, r_sd=5.0,
    r_lower=0.0, r_upper=100.0, lambda_mean=10.0, lambda_sd=10.0,
    sigma2_shape=0.001, sigma2_rate=0.001,
    inclusion_alpha=1.0, inclusion_beta=1.0,
    cluster_lambda_mean=10.0, cluster_lambda_sd=10.0, beta_sd=5.0,
    dispersion_shape=2.0, dispersion_rate=0.1,
    baseline_shape=0.1, baseline_rate=0.1,
)
```

Check each prior's parameterization when setting its fields. `r_distribution` accepts "gamma", "unif" or "invunif"; inverse-uniform bounds apply to $1/r$ for active exposures, while excluded exposures have $r=0$. Relevance and amplitude gamma priors use mean/SD fields; dispersion and baseline gamma priors use shape/rate. Residual variance uses inverse-gamma shape/rate, and inclusion probabilities use a beta prior. See the prior definitions.

For the default count/survival sampler, `prior=None` uses `lambda_mean=1`, `lambda_sd=.5`, `cluster_lambda_mean=1`, `cluster_lambda_sd=.5`; the other fields retain their listed defaults. A dictionary changes the supplied fields and keeps the family's other defaults. A `Prior` object supplies all its values explicitly, including its listed amplitude defaults. `beta_sd` controls the proper coefficient prior for count/survival. Dispersion and baseline parameters use gamma shape/rate. Structured priors use a direct GP amplitude and separate Gaussian residual-variance defaults.

6.2 Structured model results

Task	Method or helper
Exact alternative kernels	<code>kernel_matrix(Z1, Z2=None, r=None, *, kernel="gaussian", alpha=1)</code>
Intercept/slope coefficient draws	<code>fit.random_effects(cluster_new=None, ...)</code>
Gaussian residual variances	<code>fit.predict_residual_variance(Znew=None, variance_design_new=None, ...)</code>
Cause-specific h and effect summaries	<code>fit.for_cause(cause)</code>
Competing-risk probabilities	<code>fit.predict_cumulative_incidence(..., times=..., cause=None)</code>
Time-varying predictor paths	<code>fit.predict_survival_path(..., start=..., stop=..., subject_new=..., times=...)</code>

`StructuredFit` retains the shared `summary`, `sample`, `diagnostic`, `contrast` and `save/load` methods. `RandomCoefficientPrediction` stores coefficient draws; `CompetingRiskPrediction.for_cause(k)` returns a curve accepted by `plot_survival`. Additional inputs, initialization shapes and runnable examples are in the structured guide.

6.3 Fit results and diagnostics

`BKMRFit` stores `data`, `samples`, `settings` and `sampling_info`, with `family`, `exposure_names` and `covariate_names` properties. Stored arrays are read-only; extracted samples are independent copies.

Method	Return and use
<code>fit.summary(probability=0.95)</code>	Text summary of parameter means, SDs, equal-tailed intervals, available posterior inclusion probabilities (PIPs) and diagnostic messages.
<code>fit.summary_table(probability=0.95)</code>	List of dictionaries with <code>parameter</code> , <code>mean</code> , <code>sd</code> , <code>lower</code> , <code>upper</code> .
<code>fit.posterior_samples(parameter=None, combine_chains=False)</code>	Dictionary of arrays, or one named parameter array. Leading axes are chain and retained draw; combining chains merges those axes.
<code>fit.inclusion_probabilities()</code>	Exposure names and PIPs for selection fits. Group fits also return group PIP and within-group conditional PIP; the conditional value is undefined (<code>NaN</code>) if a group was never selected. Raises without selection.
<code>fit.diagnostics()</code>	Parameter diagnostics and messages. Uses rank-normalized/folded split R-hat and bulk/tail effective sample sizes.
<code>fit.save(path)</code>	Save training data, settings and states in a compressed non-pickle archive. Replaces the destination atomically.
<code>km.BKMRFit.load(path, *, max_bytes=512 * 1024 * 1024)</code>	Load and validate an archive, limiting total uncompressed member size.
<code>km.diagnose(values)</code>	Diagnostics for one finite scalar-parameter array shaped <code>(chains, draws)</code> .

The reported intervals are equal-tailed: the posterior probability below the lower limit equals that above the upper limit. A PIP is the fraction of draws including an exposure or group. R-hat compares chains; bulk/tail effective sample sizes account for draw dependence in the distribution's center/tails.

Default-route saved parameters include `beta`, `r`, `lambda`, `sigma2`, `delta` and `pi`; its binary fits also retain `ystar`. Structured arrays are described in the structured guide. `delta` records exposure inclusion. `beta` includes the

added intercept when fitted. No-selection fits store `pi=1`. Diagnostics cannot establish model validity; R-hat requires multiple chains and short or constant sequences can yield undefined diagnostics.

6.4 Prediction

```
fit.predict_h(Znew=None, *, probability=0.95, max_draws=None, seed=None)
fit.predict(
    Znew=None, Xnew=None, *, kind="mean", scale="response",
    probability=0.95, max_draws=None, seed=None, offset=None, cluster_new=None,
)
fit.posterior_moments_h(Znew=None, max_draws=None)
```

Supply exposures in original units and training column order; named columns must match. Prediction reuses training standardization. New profiles require `Xnew` when adjustments were fitted, without the added intercept. Omitting new data uses retained training rows.

Call	Quantity
<code>predict_h</code>	Joint draws of <code>h</code> , excluding linear adjustment and all random effects.
<code>predict(..., kind="mean", scale="response")</code>	Gaussian/count means, binary probabilities, or survival relative hazards.
<code>predict(..., kind="observation", scale="response")</code>	Gaussian, binary or count future observations; survival is unsupported.
<code>predict(..., kind="mean", scale="latent")</code>	<code>h</code> plus linear adjustment and requested random effects, before link transformation; count offsets are included. Correlated Gaussian means also condition on training residuals.
<code>predict(..., kind="observation", scale="latent")</code>	Gaussian outcomes or binary latent outcomes including residual noise; binary residual variance is one. Count/survival latent observation draws are unsupported.
<code>posterior_moments_h</code>	Dictionary with deterministic integrated <code>mean</code> and <code>covariance</code> for <code>h</code> , averaging conditional moments over saved draws.

`max_draws` selects evenly spaced states **per chain** (at least two), without refitting. `seed` controls prediction draws. A `Prediction` stores `draws` shaped (`chains`, `selected_draws`, `rows`), `mean`, `sd`, `lower`, `upper`, joint `covariance`, `probability`, `scale` and `kind`. `to_records()` gives one summary per row. Intervals are equal-tailed and pointwise: each describes uncertainty for one row, rather than for all rows at once. Observation intervals also include new outcome variation.

6.5 Cluster and survival results

```
fit.random_intercepts(cluster_new=None, *, probability=.95, max_draws=None, seed=None)
fit.predict_survival(
    Znew=None, Xnew=None, *, times, kind="survival", cluster_new=None,
    probability=.95, max_draws=None, seed=None,
)
km.plot_survival(prediction, row=0, ax=None)
```

`random_intercepts` returns a `Prediction` in training-cluster order, or in the supplied label order. The prediction rules for known and new clusters are described below.

Survival result	Meaning
<code>times</code>	Finite, nonnegative, strictly increasing prediction times
<code>draws</code>	Array with axes (<code>chain</code> , <code>draw</code> , <code>row</code> , <code>time</code>)
<code>mean</code> , <code>sd</code> , <code>lower</code> , <code>upper</code>	Arrays shaped (<code>row</code> , <code>time</code>)
<code>probability</code> , <code>kind</code>	Interval probability and survival/hazard scale
<code>to_records()</code>	One summary per profile/time pair

`kind="hazard"` evaluates hazards on the grid. Plots use survival curves or hazard points with intervals; they do not invent hazard steps between times.

`predict` additionally accepts `offset=None` and `cluster_new=None`. For training count prediction, omitted offsets reuse retained offsets. For new rows, offsets are required if they were supplied during fitting. Response-scale count observations are Poisson or NB2 draws. Latent count observations and survival observation draws are unsupported and raise an error.

Omitted `cluster_new` sets random effects to zero, including for training predictions. Known labels use their fitted cluster coefficients; unseen labels receive new Gaussian coefficients. Equal labels share a draw. `predict_h` excludes random effects. For slopes and residual conditioning, see structured prediction. Default Gaussian/probit prediction samples `h + u` jointly when labels are supplied, preserving posterior covariance.

For the default count/survival route, samples include `h` and auxiliary `latent_white`; negative-binomial fits also include `dispersion`; survival fits include `baseline_hazard`. Default-route clustered fits include `lambda_cluster`, and count/survival fits also retain `u`. Default Gaussian/probit fits integrate out random intercepts during parameter sampling; `random_intercepts` draws them conditionally afterward. Count/survival fits store `sigma2=1` for a consistent result format. It is not an estimated residual variance and is omitted from their summaries. Saved white coordinates are internal standardized sampling variables tied to a matrix decomposition. Use saved `h` for prediction that can be transferred across platforms.

6.6 Exposure contrasts and curves

Exposure contrasts compare predictions at specified profiles. A quantile profile sets each exposure to a percentile of its own distribution; the all-median profile uses every exposure's median. Such combinations need not occur in observed rows.

```
km.overall_effect(
    fit, quantiles=(0.25, 0.5, 0.75), reference=0.5, scale="latent",
    X=None, probability=0.95, max_draws=None, seed=None,
)

km.univariate_response(
    fit, variable=None, grid=None, points=50, fixed_quantile=0.5,
    center=True, scale="latent", X=None, probability=0.95,
    max_draws=None, seed=None,
)

km.bivariate_response(
    fit, variables=(0, 1), grids=None, points=20, fixed_quantile=0.5,
    center=True, scale="latent", X=None, probability=0.95,
    max_draws=None, seed=None,
)

km.single_exposure_effects(
    fit, quantiles=(0.25, 0.75), fixed_quantiles=(0.25, 0.5, 0.75),
    scale="latent", X=None, probability=0.95, max_draws=None, seed=None,
)
```

```
km.interaction_effects(
  fit, variables=None, quantiles=(0.25, 0.75), fixed_quantile=0.5,
  scale="latent", X=None, probability=0.95, max_draws=None, seed=None,
)
```

Function	Quantity and result
<code>overall_effect</code>	Each all-exposure quantile profile minus the all-exposure reference profile. Returns EffectResult .
<code>univariate_response</code>	One named or zero-based indexed exposure, or all exposures with <code>variable=None</code> . Other exposures stay at <code>fixed_quantile</code> . Returns one joint CurveResult .
<code>bivariate_response</code>	Every combination of the supplied grid values for two distinct exposures. Other exposures stay at <code>fixed_quantile</code> . Returns SurfaceResult .
<code>single_exposure_effects</code>	High-minus-low quantile contrast for each exposure at every background <code>fixed_quantiles</code> value. Returns EffectResult .
<code>interaction_effects</code>	All unordered pairs from the selected names/indices, or all exposures by default. Returns EffectResult . For each pair the contrast is high/high - low/high - high/low + low/low, with other exposures fixed.

Quantiles use retained original training exposures and lie in `[0, 1]`. Default grids span the observed range; explicit grids must be finite, nonempty and strictly increasing. An all-variable curve shares one supplied `grid`; a surface uses one sequence per axis in `grids`. Dense joint calculations limit calls to 4096 output locations.

With `center=True`, curves and surfaces subtract the **all-median** reference profile, even if `fixed_quantile` is changed. This differs from R `bkmr` curve centering by the average fitted grid value. The pairwise interaction above also differs from the R summary that changes all background exposures together. These definitions should be specified when comparing results between packages.

Summary scale	Target
Latent	Contrasts of $\mathbf{h}$; $\mathbf{X}$ is rejected
Response	Gaussian/count means, binary probabilities or survival relative hazards; random effects are zero
Count response	Zero log offset, representing unit observation exposure
Survival response	Differences in relative hazards, not hazard ratios or survival-probability differences

For response summaries, `X` fixes one raw adjustment vector, defaulting to training covariate means. Each joint draw is transformed to the requested scale before subtraction. This retains covariance and makes identical-profile contrasts exactly zero.

Results store `draws` shaped `(chains, selected_draws, locations)`, `mean`, `sd`, `lower`, `upper`, `scale`, `probability`, `kind` and `metadata`. `locations` labels outputs; `profiles` holds distinct original-unit rows; `coefficients` maps joint profile draws to summaries. `to_records()` combines locations and summaries. **SurfaceResult** adds `variables`, `grids` and `shape`; reshape summaries to `shape` with the second grid changing fastest.

These summaries describe associations at the specified values of the other variables. Check that nearby exposure combinations occur in the data. The intervals do not account for unmeasured confounding or a wrongly specified model.

6.7 Plotting

Matplotlib is an optional dependency and is imported only when a plot is requested.

```
km.plot_trace(fit, parameter="lambda", indices=None, ax=None)
km.plot_pips(fit, ax=None)
km.plot_response(curve, ax=None)
km.plot_overall(effect, ax=None)
km.plot_surface(surface, ax=None)
km.plot_survival(prediction, row=0, ax=None)
```

Each function returns axes, creating them when `ax=None`. The caller controls display, export and styles. Trace `indices` selects a vector component, such as `indices=0` for the first `beta`. Traces exclude warmup. PIP plots require selection.

`plot_response` accepts a univariate curve; `plot_overall` accepts an overall effect. Both show pointwise posterior intervals. `plot_surface` needs at least two values per grid and shows the posterior mean, with zero-centered colors when centered; its result retains interval limits. Use separate curves for different exposure units.

6.8 Kernel helper

```
km.gaussian_kernel(Z1, Z2=None, r=None)
```

Returns the Gaussian kernel between supplied rows. `Z2=None` reuses `Z1` and `r=None` uses unit relevance. Relevance must be nonnegative. Inputs are used directly, without a fitted training transform.

6.9 Method sources

- Bobb et al. (2015), Bayesian kernel machine regression for estimating the health effects of multi-pollutant mixtures.
- Bobb et al. (2018), Statistical software for analyzing the health effects of multiple concurrent exposures via Bayesian kernel machine regression.
- Murray et al. (2010), Elliptical slice sampling.
- Stan count-distribution definitions.
- INLA manual, piecewise-constant proportional hazards (§3.15).

7 Numerical evidence and limits

KernelMix follows the Gaussian BKMR model of Bobb et al. (2015) and the probit extension described by Bobb et al. (2018). Numerical agreement, repeated-dataset performance and original-study reproduction are separate checks.

7.1 Fitted R comparisons

We fit the same 12 observations and two exposures in Python and installed R `bkmr` 0.2.2. Inputs, sample-standardization, intercepts and priors agree. Exposure selection is disabled. Relevance and variance-ratio priors are `Gamma(shape 2, rate 2)`; Gaussian residual variance has an `inverse-gamma(shape 3, scale 1)` prior. Adjustment coefficients have flat priors, and probit residual variance is one. Both implementations use four chains and 3,000 warmup transitions per chain, followed by 120,000 saved draws in R and 60,000 in Python. Different seeds and proposals are allowed because the target is the posterior distribution, not identical chains.

We compare means and the 10th, 50th and 90th percentiles of coefficients, relevance parameters, variance parameters, three values of h and one joint exposure contrast. Monte Carlo standard error (MCSE) measures uncertainty from finite posterior sampling. The pooled posterior standard deviation is the square root of the average of the two sample variances. The combined MCSE is the square root of the sum of the two squared MCSEs.

Model	Mean/percentile comparisons	Result
Gaussian	40	Pass
Gaussian with random intercepts	44	Pass
Probit	36	Pass
Probit with random intercepts	40	Pass

Each quantity must have $\hat{R} \leq 1.01$ and bulk/tail effective sample sizes at least 400. Each summary’s MCSE must be at most 2% of the pooled posterior standard deviation. Differences must be within six combined MCSEs; the absolute difference plus three combined MCSEs must also be within 15% of that standard deviation. These are numerical screening rules, not a calibrated equivalence test or a proof of convergence. The first, shorter attempt was inconclusive because some R summaries lacked precision; its tolerances were not widened.

Constructed-state comparisons separately check conditional means, covariances, inclusion probabilities and prior calculations. Reproduction commands document the required R installation and conventions. R is never required for ordinary fitting or prediction.

7.2 Independent extension checks

R `bkmr` has no matching count or survival models. Here, NB2 denotes the negative-binomial model and GP denotes the Gaussian process. We instead compare three fixed fits with independent numerical integration. Two chains each retain 100,000 draws after 1,000 warmup transitions. All 12 checked posterior moments and four selected covariances pass their precision and error limits.

Target	Unknown quantities sampled together	Fixed structure
NB2 counts	Constant h , GP amplitude and dispersion	Four observations, fixed relevance zero
Right-censored survival	Constant h , GP amplitude and baseline rate	Four subjects, one baseline interval, no delayed entry
Structured Gaussian	Constant h , GP amplitude, random intercept/slope and residual variance	Six observations; fixed random-coefficient covariance and AR1 correlation; one nonzero Fourier feature

The covariance checks detect dependence between selected sampled quantities. A deliberately broken pairing of NB2 draws is rejected. These small constant-function cases do not establish general posterior agreement, calibration of intervals, or validation of every structured option. Separate likelihood and conditional-distribution tests cover additional inputs, censoring terms, kernels and prediction rules. The numerical results record the checked summaries; the developer runner reproduces these targets.

7.3 Original studies

Exact reproduction remains blocked by missing original data and sufficient fitting settings.

Study in the 2015 paper	Design	Missing inputs
Simulations	300 datasets per scenario; $n = 100$; three Bangladesh or 13 Boston exposure components; 10,000 iterations, last 5,000 retained	Empirical exposure rows; complete response/noise, prior, grouping and fitting settings
Bangladesh	375 children; log cord-blood As/Mn/Pb; motor score with covariate adjustment	Analysis rows, preprocessing and full fitting settings
Dog crossover	142 exposure records from 13 dogs; 13 elements; random intercepts	Outcomes, exposures, dog IDs, exclusions, groups and settings

The applications use 25,000 warmup iterations and 25,000 further iterations, retaining every fifth draw. Original seeds remain unknown. The inspected author repository lacks these data; full supplementary settings were not recovered. The simulations resampled observed exposure rows, so generated substitutes would change the target. The observed ozone example demonstrates a separate exploratory analysis. Repeated-dataset bias, interval coverage and selection performance remain open.

7.4 Platform evidence

Environment	Installed-wheel result
macOS, Python 3.14, current dependencies	512 tests passed
macOS, Python 3.10, NumPy 1.24.0 / SciPy 1.10.0 / Matplotlib 3.7.0	512 tests passed
Linux, Python 3.10, minimum dependencies	512 tests passed
Linux, Python 3.14, current dependencies	512 tests passed
Windows, Python 3.13, current dependencies	512 tests passed

The hosted checks test installed wheels outside the source checkout.