mvgam :: CHEATSHEET

The mvgam package provides tools for fitting and interrogating univariate or multivariate time series models that can include nonlinear smooth functions of covariates, dynamic temporal processes and random effects. A wide variety of latent dynamic processes can be specified. The package also provides tools for interpreting effects, computing and scoring forecasts, as well as generating model code and data objects for further customisation. Models are fitted using Stan for full Bayesian inference.

Modelling with mvgam()

Usage: `mvgam(formula, trend_formula, data, trend_model, family, ...)`

`formula`: observation model regression formula, built off the `mgcv` package. See `?mvgam_formulae` for more guidance

`trend_formula`: optional process model formula (see the State-Space model vignette and the shared latent states vignette for guidance on using trend formulae)

`data`: a `data.frame` or `list` containing the response variable(s) and optional predictor variables. See the data formatting vignette for guidance on data preparation

`trend_model`: optional latent dynamic process. Options include:

- `None`: default, no dynamic trend
- `RW(ma = FALSE, cor = FALSE)`: random walk
- `AR(p = 1, ma = FALSE, cor = FALSE)`: autoregressive
- `VAR(ma = FALSE, cor = FALSE)`: vector autoregressive
- `PW(growth = 'linear')`: piecewise linear
- `PW(growth = 'logistic')`: piecewise logistic, with max saturation
- `GP()`: squared exponential Gaussian Process

For autoregressive processes (`RW(), AR() or VAR()`), moving average and correlated process errors can also be specified by changing the `ma` and `cor` arguments

`family`: observation distribution. Options include:

- `gaussian()`: Gaussian with identity link
- `student-t()`: Student's T with identity link
- `lognormal()`: LogNormal with identity link
- `Gamma()`: Gamma with log link
- `betar()`: Beta with logit link
- `poisson()`: Poisson with log link
- `nb()`: Negative Binomial with log link

See the introductory vignette for more guidance on supported families and dynamic processes

`...`: other arguments such as user-specified `priors`, `newdata` for generating probabilistic forecasts and options to control `Stan` MCMC parameters

Prior to modelling, it is useful to:

- Inspect features of the data with `plot_mvgam_series()`
- Ensure there are no `NA`'s in predictors (though `NA`'s are allowed in response variables). See the data formatting vignette for guidance on data preparation
- Inspect default priors with `get_mvgam_priors()`
- Make any necessary changes to default priors with `prior()`

`sim_mvgam()` is useful to generate simple example datasets

```
simdat <- sim_mvgam(n_series = 1)
model <- mvgam(formula = y ~ s(season, bs = 'cc'),
               trend_model = RW(), data = simdat$data_train)
```

Use `code(model)` to see the auto-generated Stan code

Diagnostics and Inference

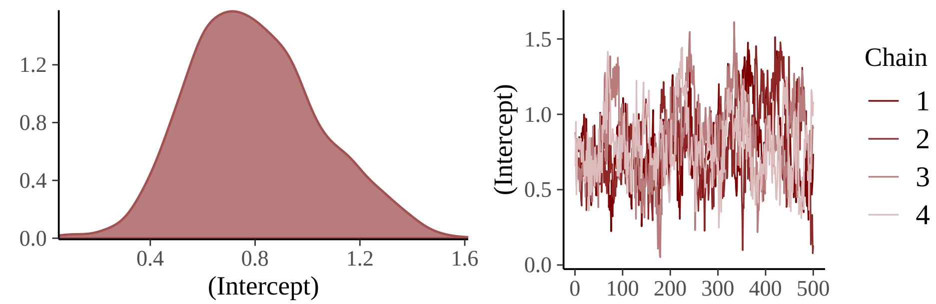
What effects has the model estimated?

`summary(model)` and `coef(model)`: posterior summaries and diagnostics

`fitted(model)`, `logLik(model)` and `residuals(model)`: posterior expectations, pointwise Log-Likelihoods and randomized quantile residuals

`loo(model)` and `loo_compare(model1, model2, ...)`: calculate approximate leave-one-out information criteria

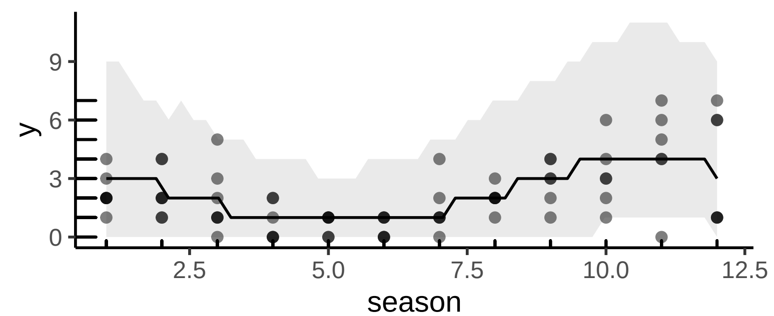
`mcmc_plot(model)`: visualize posterior summaries, pairs plots and a wide range of MCMC diagnostics using functionality from the Bayesplot package



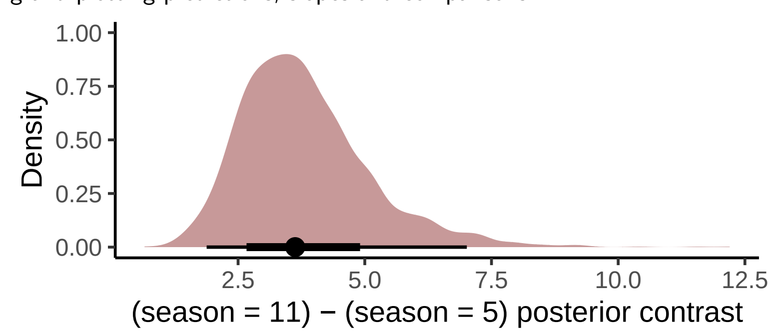
Use `as.data.frame(model)`, `as.matrix(model)`, or `as.array(model)` for posterior extraction. Use `variables(model)` to determine what parameters are available for extraction

The S3 `plot()` function applied to models can visualise smooth functions (`type = 'smooths'`), random effects (`type = 're'`), posterior predictions and trend estimates (`type = 'forecast'` or `type = 'trend'`) uncertainty contributions (`type = 'uncertainty'`) or randomized quantile residual diagnostics (`type = 'residuals'`). Use `trend_effects = TRUE` to visualise effects from any process model formulae

`conditional_effects(model)` gives useful conditional effect plots on either the response or the link scale



For most `mvgam` models, functions from the `marginaleffects` package can be used for more targeted prediction-based inference. See The Marginal Effects Zoo for guidance on computing and plotting predictions, slopes and comparisons



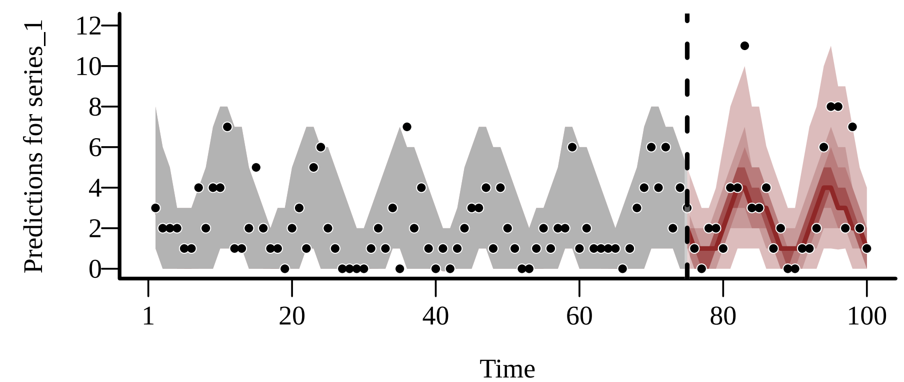
Prediction and forecasting

How good are model predictions?

Use `predict(model)` with `newdata` to make predictions for inference purposes. Change the `type` argument for different types of predictions (link scale, expectation or response scale). Or use the `brms` package equivalents `posterior_predict(model)`, `posterior_linpred(model)` or `posterior_epred(model)`. If generating forecasts for future timepoints, use the `forecast()` function (see below)

Use `ppc(model)` to plot various kinds of posterior predictive checks to compare model predictions against true observations

Extract in-sample posterior predictions with `hindcast(model)`. If validation data exist, generate forecast predictions with `forecast(model, newdata = newdata)`. As above, change the `type` argument for predictions on different scales. Both functions generate an object of class `mvgam_forecast`, that can be plotted with an S3 `plot()` function. See the forecasting vignette for more details about how to produce forecasts.



Compute probabilistic forecast scores using proper scoring rules with the `score()` function:

```
fc <- forecast(model, newdata = simdat$data_test, type = 'response')
crps <- score(fc, score = 'crps')
dplyr::glimpse(crps$series_1)
```

```
## Rows: 25
## Columns: 5
## $ score          <dbl> 0.2435, 0.4058, 0.7062, 0~
## $ in_interval    <dbl> 1, 1, 1, 1, 1, 1, 0, 1~
## $ interval_width <dbl> 0.9, 0.9, 0.9, 0.9, 0.9, ~
## $ eval_horizon   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9~
## $ score_type     <chr> "crps", "crps", "crps", "~
```

Available proper scoring rules in the `score()` function include:

- `type = 'crps'`: Continuous Rank Probability Score (univariate)
- `type = 'drps'`: Discrete Rank Probability Score (univariate)
- `type = 'elpd'`: Expected Log Predictive Density (univariate)
- `type = 'sis'`: Scaled Interval Score (univariate)
- `type = 'energy'`: Energy Score (multivariate)
- `type = 'variogram'`: Variogram Score (multivariate)

Use `lfo_cv(model)` for approximate leave-future-out cross-validation with an expanding window training technique (see Bürkner et al. 2020 for details of the algorithm). This generates expected log predictive density scores at user-specified forecast horizons, which can be used to compare different models