mcp: Regression with multiple change points

Jonas Kristoffer Lindeløv [1*]

[1] Department of Communication and Psychology, Aalborg University, Denmark.
[*] Corresponding author, Email: lindeloev@gmail.com

Abstract

The R package mcp does flexible and informed Bayesian regression with change points. mcp can infer changes in means, variances, autocorrelation structure, and any combination of these. Regression models can be specified on a segment-by-segment basis, including regression on variance and autoregressive parameters. Prior and posterior samples and summaries are returned for all parameters as well as a rich set of plotting options. Bayes Factors can be computed via Savage-Dickey density ratio and posterior contrasts. Cross-validation can be used for a more general model comparison. mcp ships with sensible defaults, including priors, but many options for the user to take control of these settings. The strengths and limitations of mcp are discussed in relation to existing change point packages in R.

Keywords: change point, piecewise linear, break point, regression, R, Bayesian

1 Introduction

A change point is a location on a continuum where a trend changes. Change point analyses are useful for detecting when a change happens in the rate of disasters due to new policies or practices (Raftery and Akman, 1986), changes in stock prices (Chen and Gupta, 1997), discontinuities in human performance (Cowan, 2000), and many other scenarios. The widespread applicability of change point analysis is evident from its many names: switch points, break points, broken line, broken stick, (performance) discontinuity models, bilinear regression, piecewise regression, local linear regression, and segmented regression.

More than 10 R packages are available to analyze change points. Most packages detect change points using a threshold, including `ecp` (James and Matteson, 2015), `bcp` (Erdman and Emerson, 2007), `changepoint` (Killick and Eckley, 2014), `changepoint.np` (Haynes and Killick, 2019), `TSMCP` (Li and Jin, 2018), `cpm` (Ross, 2015), `EnvCpt` (Killick et al., 2018), and `wbsts` (Korkas and Fryzlewicz, 2018). This is useful when the number of change points is unknown a priori. They return point estimates of the change points as their only output. Other packages take a fixed number of change points and a user-specified regression model, with the restriction that the same regression model must apply to all segments, including `segmented` (Muggeo, 2008) and `strucchange::breakpoints()` (Zeileis et al., 2003). A limited number of regression models are supported in the latter packages.

However, there are many cases where it is desirable to specify the regression models on a per-segment basis. More generally, the analyst will often have a priori knowledge about the number of change points and credible parameter values. For example, human short-term memory capacity can be modeled as a

performance discontinuity in accuracy as a function of task difficulty (Cowan, 2000; Camos and Tillmann, 2008; Leibovich-Raveh et al., 2018). Performance is fast and virtually errorless for easy tasks (a high plateau), but there is an abrupt change to deteriorating accuracy (joined negative slope) when task difficulty exceeds the participant's abilities. Identification of the memory capacity has classically been done by eye-balling graphs or coming up with ad-hoc methods to automate this (Leibovich-Raveh et al., 2018). Similarly, the effects of many nutrients follow a positive quadratic trend with an exponent between 0 and 1, followed by a plateau at a saturation point (Pesti et al., 2009).

In these cases, the number of change points is known, there are a priori bounds on the sign of the coefficients, and all parameters of the model are of interest - not just the change point. There may also be classical regression problems where change points are nuisance effects that need to be co-varied out, so detailed information and tests for *all* parameters (again, not just the change points) can also be useful. `mcp` aims to full in the missing piece in the R landscape by being useful for all of these purposes. More generally, `mcp` aims to do GLMM regression with multiple change points that change between user-specified GLMM segments, thus providing maximum modeling flexibility. `mcp` takes a Bayesian computational approach and, as we shall see, this has the added benefit of properly quantifying uncertainty around the change points - a seemingly impossible task for non-computational methods.

Part 1 of this paper introduces the regression model(s) underlying `mcp`. Part 2 discusses priors for change point models. Part 3 demonstrates the usage of `mcp` and it's features. Part 4 discusses the strengths and limitations of `mcp` in general, and in comparison to the other change point packages in R.

2 An indicator model of multiple change points

2.1 From if-else to indicators

Let Δ be a location on the continuous scale $\mathbf{x}$ where the prediction μ_i of y_i changes from following function $f_1(\mathbf{x}|\beta_\mathbf{1})$ to $f_2(\mathbf{x}|\beta_\mathbf{2})$ where β are regression coefficients. An often-used formalization of a single change point (see e.g. (Stephens, 1994; Carlin et al., 1992)) is:

$$\mu_i = \begin{cases} f_1(x_i|\beta_\mathbf{1}) & \text{if } x_i < \Delta \\ f_2(x_i|\beta_\mathbf{2}) & \text{if } x_i > \Delta \end{cases} \tag{1}$$

Each function f takes the coefficients β. For example, a linear segments is $f_k(x_i|\beta_\mathbf{k}) = \beta_{k,0} + \beta_{k,1} \cdot x_i$. However, formulation (1) does not readily generalize to K segments, i.e., $K-1$ change points. In addition, time-series and other change point models are often joined at the change points, so a cumulative formulation would be desirable. The core novelty of `mcp` is to use an indicator formulation to achieve these goals:

$$\mu_i = \sum_{k=1}^{K} [x_i > \Delta_{k-1}] f_k(X_{k,i}|\beta_\mathbf{k}) \tag{2}$$

The indicator function $[x_i > \Delta_{k-1}]$ evaluates to 1 if true and 0 if false (Knuth, 1992). That is, for a given `x_i`, the parameters of the current and previous segments predict μ_i while future segments are multiplied by zero. The indicator can be ignored for the first segment ($k = 1$, corresponding to setting $\Delta_0 = -\infty$) or, as

`mcp` defaults to, the change point can be restricted to the observed range of $\mathbf{x}$ by setting $\Delta_0 = \min(\mathbf{x})$. Also,

$$X_{k,i} = \min\{x_i, \Delta_{k+1}\} - \Delta_k \tag{3}$$

is the "local" x_i to segment k. It is zero at the segment onset (at Δ_k) and reaches a plateau at the segment's offset(at Δ_{k+1}). This allows for the modeling of joined segments, handing over the function value like a baton to the next segment. See appendix 2 for the associated likelihood.

2.2 Relative, absolute, and segment-specific models

Simple manipulations to this basic indicator scheme accommodates more flexible models.

To model *relative* changes in slopes from segment $k-1$ to k, simply replace $\beta_{k,1}$ with $\beta_{k-1,1} + \beta_{k,1}$ so that $\beta_{k,1}$ is the *change* in slope. To model *absolute* intercepts in segment k, simply multiply earlier segments with the additional indicator $[x_i < \Delta_k]$ which will "turn them off" because it evaluates to zero once x_i crosses into segment k.

Various functions can be applied to $\mathbf{x}$. For example, quadratic trends can be added with $\beta_{k,i} \cdot X_{k,i}^2$, logarithms with $\beta_{k,i} \cdot \log(X_{k,i})$, and so forth for exponents, trigonometric functions, etc.

We can arbitrarily add or omit terms in any one segment by writing out (2) and adding/removing individual terms. For example, (4) is a plateau followed by a joined slope followed by a quadratic trend starting at an absolute intercept:

$$\begin{aligned}
X_{2,i} &= \min\{x_i, \Delta_2\} - \Delta_1 \\
X_{3,i} &= \min\{x_i, \Delta_3\} - \Delta_2 \\
\mu_i &= [x_i > \Delta_0][x_i < \Delta_2]\beta_{1,0} + \\
&\quad [x_i > \Delta_1][x_i < \Delta_2]\beta_{2,1}X_{2,i} + \\
&\quad [x_i > \Delta_2](\beta_{3,0} + \beta_{3,1}X_{3,i}^2)
\end{aligned} \tag{4}$$

This model is used as an example throughout this paper. It is visualized in Figure 2).

We can easily share βs between segments. For example, setting $\beta_{3,0} = \beta_{1,0}$ models identical intercepts in segments 1 and 3. This can be used to model, e.g., a change in intercept while keeping a slope identical between segments.

2.3 Generalized linear model

While classic regression takes a Gaussian response family and an identity link function, change points can be modeled for all response families and link functions. For example:

$$\begin{aligned}
y_i &\sim \text{Normal}(\mu_i, \sigma) \\
y_i &\sim \text{Binomial}(n_i, \text{logit}^{-1}(\mu_i)) \\
y_i &\sim \text{Poisson}(\exp(\mu_i))
\end{aligned} \tag{5}$$

where σ is the standard deviation of the residuals and n_i is the number of trials for observation y_i.

2.4 Variance and autocorrelation change points

Hitherto, we have discussed modeling changes in the central tendency, μ. Another frequent application is modeling changes in variance. Existing R packages for variance change points models a change from one variance to another, i.e., an intercept change in variance. However, we can model σ_i as a function of x_i using the same regression model as that for μ_i above. For a change between two segments with different (absolute) means and variances, we have:

$$\begin{aligned}
\mu_i =& [x_i > \Delta_0][x_i < \Delta_1]\beta_{1,0} + \\
& [x_i > \Delta_1]\beta_{2,0} \\
\sigma_i =& [x_i > \Delta_0][x_i < \Delta_1]\theta_{1,0} + \\
& [x_i > \Delta_1]\theta_{2,0} \\
y_i \sim& \text{Normal}(\mu_i, \sigma_i)
\end{aligned} \tag{6}$$

where, again, $\Delta_0 = -\infty$ or $\Delta_0 = \min(\mathbf{x})$. This means that we can easily model slopes on σ_i as well as periodic (sine and cosine), quadratic, and other functions.

Change point analysis is also frequently applied to time-series where the autocorrelation can be accounted for using Nth-order autoregressive parameters. In the context of regression, they apply to the residuals on the link scale ($g(y_i) - \mu_i$) predicting residual i as a slope on residual $i - N$ for each N:

$$\begin{aligned}
\epsilon_i &= \sum_{k=1}^{N} \phi_N (g(y_{i-N}) - \mu_{i-N}) \\
y_i &= \text{Normal}(\mu_i + \epsilon_i, \sigma_i)
\end{aligned} \tag{7}$$

We can subject each ϕ_N to the same regression as we did for μ and σ, making it a target for change points and flexible modeling. For AR(N) models, any variance parameter, σ, now represents the *innovations*, i.e., the variance not accounted for by ϵ_i.

2.5 Varying change points

The change point parameter(s) can be subjected to regression too. However, this is more involved because their property of defining a location on $\mathbf{x}$ raises the requirement that all of $\mathbf{x}$ (and the associated $\mathbf{y}$) must be observed for each level of the regressors. As of v. 0.2.0, `mcp` supports by-group deviances ("random intercepts") from the change points ("fixed effects"). `mcp` implements a hierarchical model where the deviation for group k from the population-level change point, Δ_k, is sampled from a normal distribution with mean, Δ_k, and a dispersion parameter ς_k:

$$\delta_{k,j} \sim \text{Normal}(\Delta_k, \varsigma_k) \tag{8}$$

and the indicator in used to model the varying effect relative to change point, Δ_k, in (2) and (4) becomes

$$[x_i > \delta_{k,j}] \tag{9}$$

While varying effects are often used to "covary out" nuisance factors, varying change points are useful for estimating, e.g., individual differences in memory capacity or other human performance discontinuities (Lindeløv, 2018). Modeling the data from all participants in one model increases the precision of the individual parameters over conventional per-participant analyses [Leibovich-Raveh et al. (2018); kruschke2018].

2.6 An R formula interface

R (R Core Team, 2019) provides a compact syntax to specify regression models. The coefficients are implicit so that `y ~ 1 + x` means $y_i = \beta_0 1 + \beta_1 x_i$. Python too has begun adopting this syntax via the `patsy` module.

Given a dataset with the response column `y` and the predictor column `x`, you can specify model (4) as a list of formulas:

```
model = list(
  y ~ 1,         # Plateau
    ~ 0 + x,       # Joined slope
    ~ 1 + I(x^2)   # Disjoined quadratic
)
```

The coefficients in the three formulas are estimated as are the two change points that define their boundaries. See a fit based on this model in Figure 2.

With `lme4`, the syntax `y ~ \ldots + (formula|group)` was added for mixed models (Bates et al., 2015). `brms` has extended the syntax to include missing values (`mi(x)`), monotonic effects (`mo(x)`), and many others (Bürkner, 2017). The `mcp` model specification builds on this tradition: You can infer changes in the variance using `sigma(formula)` terms. The following would apply the same regression model to the variance that the model above did for the mean:

```
model = list(
  y ~ 1 + sigma(1),        # Plateau
    ~ 0 + sigma(0 + x),      # Joined slope
    ~ 0 + sigma(1 + I(x^2)) # Disjoined quadratic
)
```

AR(N) take the format `ar(order, formula)` where `formula` defaults to an intercept (`1`). So adding `ar(2)` to a segment would model 2nd order autoregressive residuals for that segment and later segments, until otherwise specified.

The left-hand side of segment 2+ specifies the change point. It defaults to an intercept, i.e., an "intercept change point". For example, `~ x` is interpreted as `1 ~ x`. You can let a change point intercept vary by a categorical predictor. For example, the following models varying-by-id change points from a plateau to a joined slope where all other parameters are shared:

```
model = list(
  y ~ 1,              # Plateau
```

```
  1 + (1|id) ~ 0 + x  # Joined slope at cp_1 + delta_{1, id}
)
```

These expressions can be combined to characterize a given change point as a change in the mean, variance, autoregression, and varying effect all at once (see an example in Figure 4).

The response family, the link function, and the prior is specified in the call to `mcp::mcp()`. For example, the model above can be fitted using Gaussian residuals (the default), Poisson, Binomial, or Bernoulli:

```
fit_gauss = mcp(model, data)  # Defaults to gaussian()
fit_poiss = mcp(model, data, family = poisson(link = "log"))
```

Examples of these modeling options are visualized in Figure 4.

3 Priors

3.1 Default priors for change points

Uninformative priors are assigned to all parameters in the absence of user-provided priors. The default priors should be suitable for estimation in the absence of strong prior knowledge. While setting priors for intercepts, slopes, etc. follow the same principles as other (non-change-point) regression problems, setting priors for the change points presents a challenge of its own. Due to the non-interchangeable segment order in `mcp`, the prior for the change points should be ordered monotonically in the observed range ($\min(\mathbf{x}) < \Delta_1 < \Delta_2 < \ldots < \Delta_K < \max(\mathbf{x})$) while otherwise remaining as uninformative as possible.

The ideal would probably be the Dirichlet prior described below and hence it will be described first. However, it samples 10-100 fold less efficiently than many other priors, so `mcp` defaults to using an approximation called the "t-tail" prior. The Dirichlet and the t-tail prior is visualized for $K = 2$ and $K = 5$ in Figure 1.

3.1.1 Dirichlet prior

`mcp` supports a scaled and shifted Dirichlet prior on the difference between change points - an approach that is also used to model monotonic effects in brms (Bürkner and Charpentier, 2018). The Dirichlet is itself a simplex: $\Delta_i > 0$ ensures ordering and $\sum_{k=1}^{K} \Delta_i = 1$ so that it can be shifted and scaled to the observed range. Furthermore, the sum of the probability density functions (PDFs) is Uniform$(0, 1)$ (or Beta$(1, 1)$), meaning that it represents equal prior credence that a change occurs at any point in the interval $[0, 1]$. The Dirichlet prior is scaled to the observed range of $\mathbf{x}$ using $\Delta_k = \min(\mathbf{x}) + (\max(\mathbf{x}) - \min(\mathbf{x}))\Delta_{dirich_k}$.

The resulting PDFs for the individual change points are Beta distributions. Specifically, if all $\alpha_k = 1$ the marginal priors are $\Delta_k \sim \text{Beta}(k, K + 1 - k)$ before scaling (see appendix for more details). When there is just one change point, this simplifies to $\Delta_1 \sim \text{Beta}(1, 1)$, i.e., a uniform prior in the observed range.

To illustrate, the marginal PDFs for a five-change points model are visualized in figure 1. An axis for an example data scale is added when $\min(\mathbf{x}) = 50$ and $\max(\mathbf{x}) = 150$.

3.1.2 t-tail prior

For models with one change point ($K = 1$), the default prior is simply

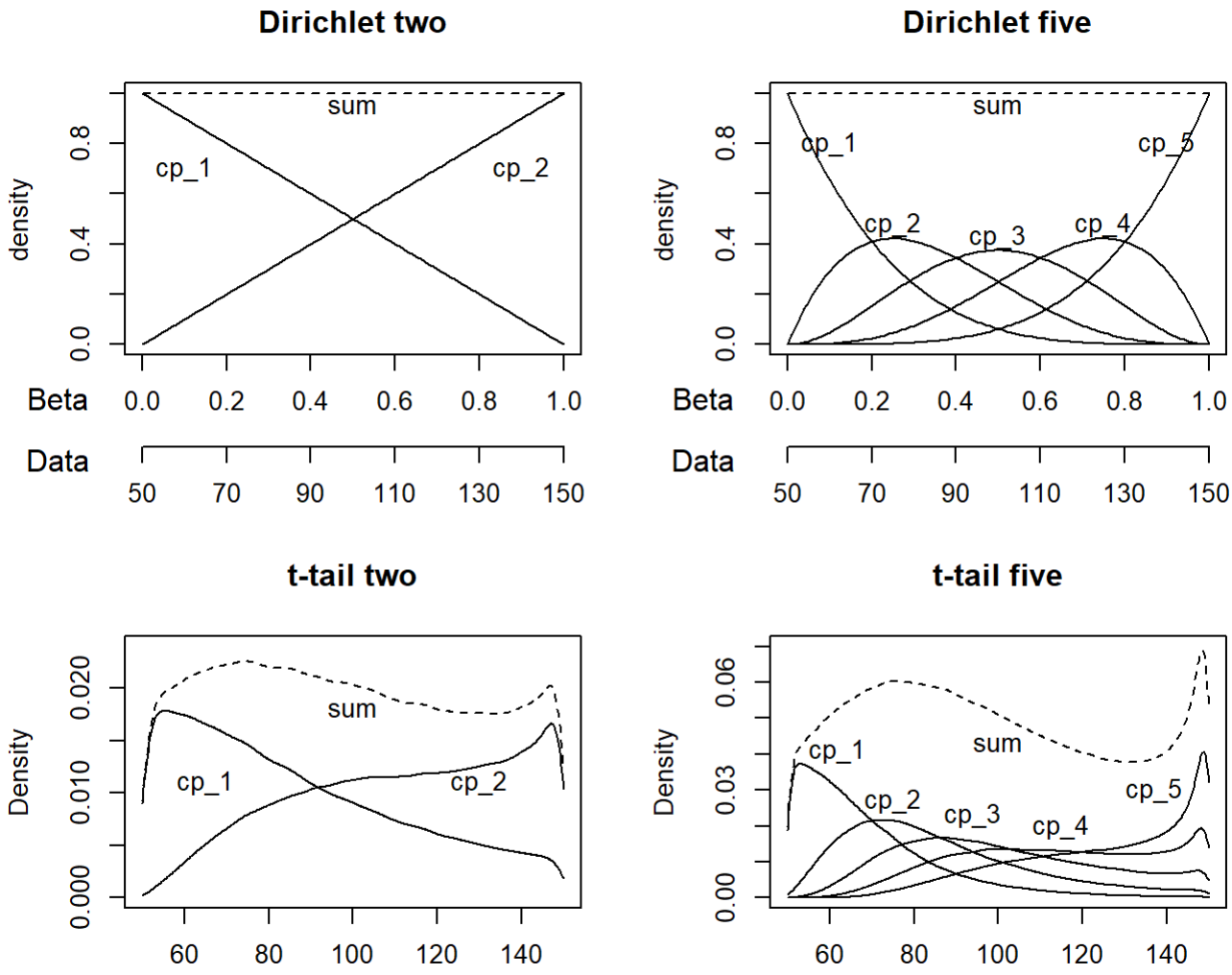

Figure 1: Dirichlet and t-tail priors for two and five change points respectively. Here shown on data where $\min(\mathbf{x}) = 50$ and $\max(\mathbf{x}) = 150$. Both priors are flat at $K = 1$. As K increase, so does the informativeness of the prior for any given change point.

$$\Delta_1 \sim \text{Uniform}(\min(\mathbf{x}), \max(\mathbf{x})) \tag{10}$$

which is identical to the Dirichlet prior for $K = 1$. For models with $K > 1$ a "t-tail" prior is the default for all change points. It (a) yields posteriors that are virtually identical to the Dirichlet, (b) has good sampling efficiency, and (c) is reasonably uninformative.

The t-tail prior is a series of t-distributions where ordering is imposed via left-truncation, i.e., leaving just the tails:

$$\Delta_k \sim t(\mu = 0, \sigma = \frac{1}{K}, \nu = K - 1), \Delta_k \in (\Delta_{k-1}, 1) \tag{11}$$

before scaling and $\Delta_0 = 0$. As K increases, the degrees of freedoms ν increase and the variance σ decreases to narrow the priors in a way that keeps the sum of the densities approximately flat. After shifting to start at $\min(\mathbf{x})$ and scaling to the range $\max(\mathbf{x}) - \min(\mathbf{x})$, we get

$$\Delta_k \sim t(\min(\mathbf{x}), \frac{\max(\mathbf{x}) - \min(\mathbf{x})}{K}, K - 1), \Delta_k \in (\Delta_{k-1}, \max(\mathbf{x})) \tag{12}$$

where $\Delta_0 = \min(\mathbf{x})$. While the t-distribution "base" is identical for all Δ_k, the cumulative left-truncation(s) imposes a conditionality that make them differ. Δ_1 is a half-t while Δ_{2+} are t-tails.

3.1.3 Priors for varying effects

The default prior for varying change points are normals that are truncated to lie between the two adjacent change points, i.e., $\Delta_{k-1} < \delta_{k,j} < \Delta_{k+1}$.

3.2 Default priors for other coefficients

The default priors for intercepts and slopes in Gaussian families are normals that are vaguely based on data to ensure that they remain uninformative across any order of magnitude. They mimic the priors used in `brms` (Bürkner, 2017).

For Gaussian families, the prior for the intercept is $t(0, 3 * \text{sd}(y), 3)$ and the prior for the slope is $t(0, \text{sd}(y)/(\max(\mathbf{x}) - \min(\mathbf{x})), 3)$, i.e., a proportional *change* over the observed range of $\mathbf{x}$. The prior for σ_1s is $\mathcal{N}(0, \text{sd}(y))$ truncated to positive values. A Uniform$(-1, 1)$ prior is used on autoregressive coefficients to ensure stationarity (Fuller and Hasza, 1981). $\mathcal{N}(0, 10)$ is used for Poisson intercepts and slopes reflecting an expected value of 1 and a 68.3% prior credence counts between $\exp(-10) = 1/22026$ and $\exp(10) = 22026$ and likewise for the total *change* in counts over the observed range of $\mathbf{x}$. A $\mathcal{N}(0, 3)$ is used for Bernoulli and binomial intercepts on a logit scale reflecting 68.3% prior credence in probabilities between 4.7% and 95.3%, and $\mathcal{N}(0, 3/(\max(\mathbf{x}) - \min(\mathbf{x}))$ for slopes reflecting similar prior credence for the *change* in probabilities over the observed $\mathbf{x}$.

3.3 Setting and using priors

Users can specify priors using named lists. The names are the parameter names and the values are JAGS code. Consider this:

```
prior = list(
  cp_1 = "dunif(10, 50)",
  x_2 = "dnorm(0, 3) T(0, )",
  int_3 = "int_1"
)
```

Here, the first change point `cp_1` is restricted to lie in the interval [10, 50] with uniform probability. The slope on `x` in segment 2 (`x_2`) is a positive half-normal. The intercept in segment 3 (`int_3`) is shared with the intercept in segment 1 and they are modeled as one parameter. You can also set constants using e.g., `int_3 = 20` so `int_3` will not be inferred. The two latter priors are true priors: they express a 100% belief in the identity and the value respectively.

A priori knowledge about more likely change point locations can be specified by setting $\alpha_k > 1$ for one or several change points, e.g., `cp_2 = "dirichlet(2)"`. Uniform priors and truncation can be used to exclude modeling change points for certain `x`, e.g., `prior = list(cp_1 = "dunif(20, 50)", cp_2 = "dnorm(80, 30) T(cp_1 + 10, MAXX)")`. When no truncation is specified on change points by the user, `mcp` adds the order restriction (`T(cp_i-1, MAXX)`).

4 Using mcp

The mcp website contains numerous worked examples for all `mcp` features. Here is a simple example that follows a procedure which scales well from simple to complex inference problems.

First, specify a model as a list of formulas. For now, we use the previously discussed three-segment model.

4.1 Simulate data

Second, simulate data to evaluate the model's performance. `mcp` can simulate data for all supported models, including regression on variance and autocorrelation terms. Start by instantiating an `mcpfit` S3 class without sampling. This object contains all model information, including a `simulate()` function to simulate data for this model given coefficient values.

Here we simulate 50 data points from some illustrative coefficients. The simulated data is visualized in figure 2.

```
# Set up
library(mcp)
options(mc.cores = 3)

# Get mcpfit object without sampling
empty = mcp(model, sample = FALSE)

# Simulate data
set.seed(42)
df = data.frame(x = 1:50)  # x_i
df$y = empty$simulate(     # associated y_i
```

```
  x = df$x,
  cp_1 = 12, cp_2 = 30,
  int_1 = 5, int_3 = 5,
  x_2 = 0.5, x_3_E2 = -0.005,
  sigma_1 = 2
)
```

4.2 Set priors and inspect the prior predictive

Using the priors discussed in "Setting and using priors", we can sample the prior and do a prior predictive check that the priors jointly covers a reasonable region - not too narrow and not too wide (Conn et al., 2018) (see figure 2).

```
fit_prior = mcp(model, data = df, prior = prior, sample = "prior")
print(fit_prior$prior)
```

4.3 Results and interpretation

When you are satisfied with your model and priors, fit it to the simulated data to learn whether the parameters can be recovered. For the present example, we instruct `mcp()` to sample both the prior and the posterior (`sample = "both"`) because we will need the prior to compute a Savage-Dickey Density Ratio later. `mcp()` defaults to sampling the posterior only.

```
fit = mcp(model, data = df, prior = prior, sample = "both")
```

The `plot()` function visualizes the inferred model (see Figure 2. Fitted and/or predictive quantiles of the highest-density region can be added as well, defaulting to 95% intervals.

```
library(ggplot2)
plot(fit, q_fit = TRUE) +
  ggtitle("Posterior fit") +

plot(fit_prior, lines = 0, prior = TRUE, q_predict = c(0.2, 0.8)) +
  ggtitle("Prior predictive")
```

`summary()` summarises the posterior means and the highest density intervals for each parameter. When the data are simulated, the values used for simulation are included as well to ease model evaluation (columns `sim` and `match` which is "OK" if `lower < sim < upper`). We see that all parameters were well recovered. The summary includes diagnostics of the MCMC sampling as well. A Gelman-Rubin convergence diagnostic below 1.1 is usually taken as a good indication that the MCMC chains have converged well (Gelman and Rubin, 1992). The effective sample size expresses the amount of information that the MCMC chains carry for each parameter (Martino et al., 2017). It is inversely related to the MCMC autocorrelation. These diagnostics are computed using the `coda` package (Plummer et al., 2006).

```
summary(fit)
```

```
## Family: gaussian(link = 'identity')
```

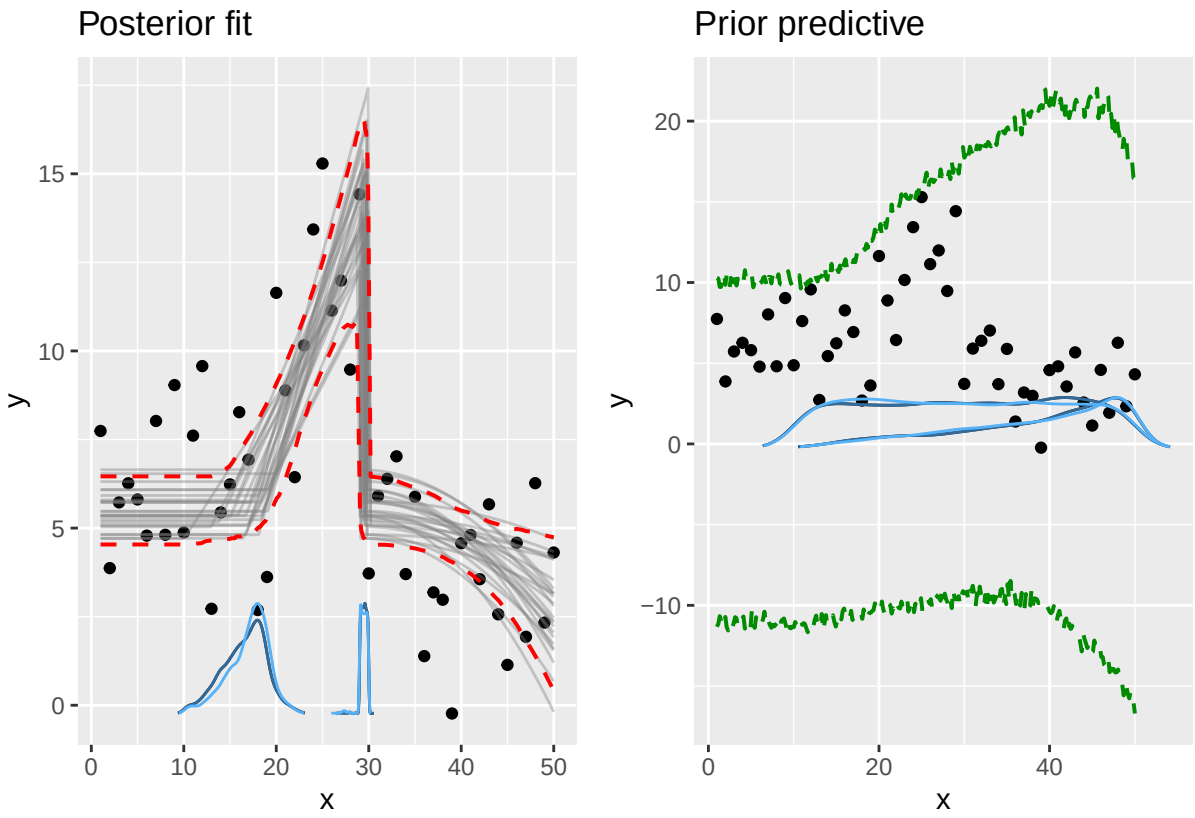

Figure 2: Prior predictive and posterior fits pertaining to the model specified in this section (see also (4)), plotted using `plot()`. **Left:** Raw data (points, simulated in the main text) with 25 draws from the joint posterior (grey lines) and 95% highest density interval (dashed red lines). Posterior distributions of the change points are shown in blue - one line for each chain. **Right:** The same as left, but for prior samples and with a 60% prediction interval (green lines). Note that the uniform prior on the first change point was set manually in the section on the prior API and is not an mcp default.

```
## Iterations: 9000 from 3 chains.
## Segments:
##   1: y ~ 1
##   2: y ~ 1 ~ 0 + x
##   3: y ~ 1 ~ 1 + I(x^2)
##
## Population-level parameters:
##     name match    sim    mean  lower   upper Rhat n.eff
##     cp_1    OK 12.000 16.6919 11.468 20.6705  1.0   834
##     cp_2       30.000 29.4674 29.001 29.9618  1.1   511
##    int_1    OK  5.000  5.5182  4.523  6.3986  1.0  1754
##    int_3    OK  5.000  5.5182  4.523  6.3986  1.0  1754
##  sigma_1    OK  2.000  2.3395  1.899  2.8687  1.0  3871
##      x_2    OK  0.500  0.7117  0.349  1.0935  1.0  1022
##   x_3_E2    OK -0.005 -0.0073 -0.014 -0.0011  1.0  2783
```

While `plot()` visualizes the whole model, `plot_pars()` visualizes individual parameters. The default plot includes a posterior density overlay with one curve for each chain, and a trace plot to inspect convergence history and mixing. Under the hood, this leverages the `bayesplot` package (Gabry et al., 2019) and supports many different plot types.

Notice how the change point posteriors do not conform to any readily known distribution, highlighting the need for a computation approach. This propagates to parameters of the following segment(s) as can be seen in the joint distribution between the change point to the second segment and its slope (see 3).

```
plot_pars(fit, pars = c("cp_1", "cp_2", "x_2")) +
  plot_pars(fit, pars = c("cp_1", "x_2"), type = "hex")
```

4.4 Testing parameter values

Hypotheses about parameter values can be tested with the `hypothesis()` function which is heavily inspired by the corresponding function in `brms` (Bürkner, 2017).

The Savage-Dickey density ratio is the ratio of the prior and posterior density at a particular value and corresponds to a point Bayes Factor (Verdinelli and Wasserman, 1995). Point Bayes Factors are as much an expression of the prior as the posterior, so careful specification of the prior is warranted.

In the current example, we may want to test whether $x_2 = 0.5$, i.e., corresponding to a one-sample t-test. We can do this using a Savage-Dickey test. The Savage-Dickey density ratio is the ratio of the prior and posterior density at a particular value, and corresponds to a point Bayes Factor (Verdinelli and Wasserman, 1995). Point Bayes Factors are as much an expression of the prior as the posterior, so careful specification of the prior is warranted.

The output includes the estimate of the deviance from the hypothesis. It also includes `p` and `BF` will express the evidence *in favor* of the specified hypothesis. The inverse (i.e., $1 - p$ and $1/BF$) would then be the evidence in favor of the alternative. The Savage-Dickey test yields around $BF = 6$ strengthening the belief in this point hypothesis by a factor of around 6 relative to the prior belief.

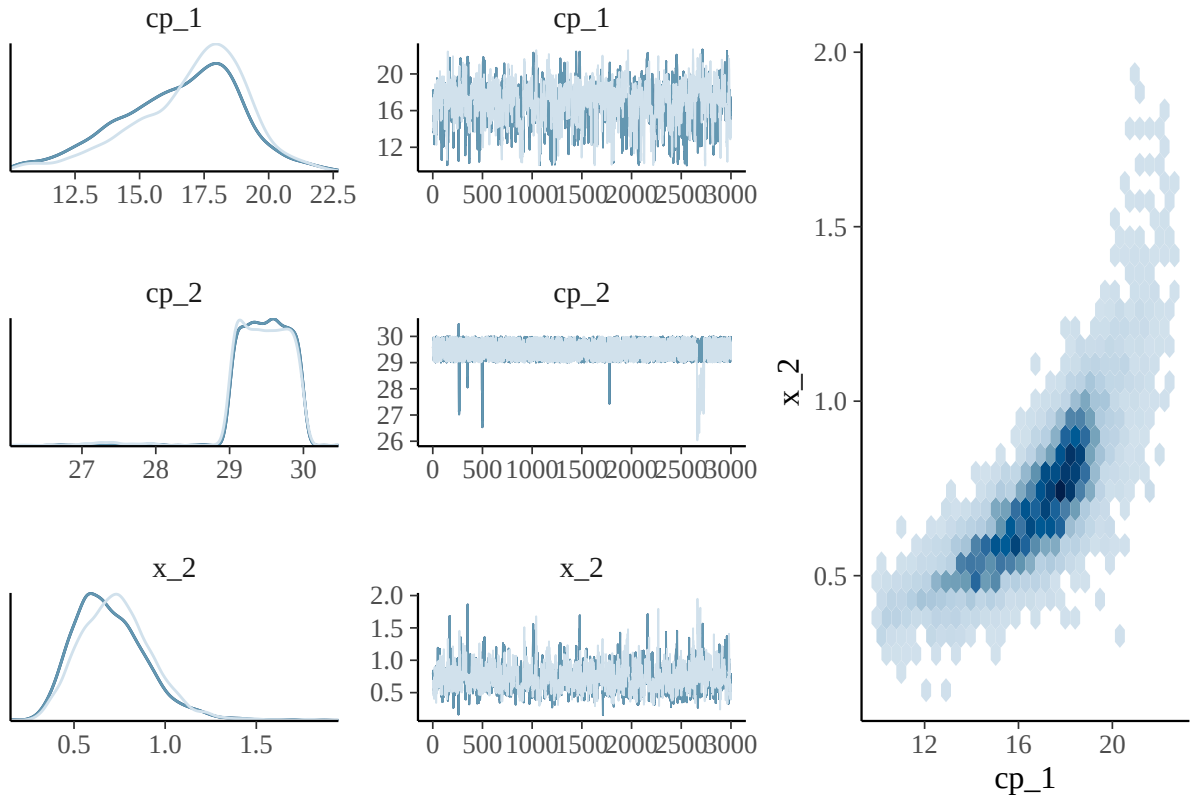

Figure 3: **Left:** Posteriors and trace plots for some selected individual parameters (rows) and three chains (colors) of the model from the section on the model API (see also Figure 2). The mixing and convergence is satisfactory here, as also indicated in the summary in the main text. **Right:** A joint density plot for the first change point and the slope in the following segment. These plots are the output of `plot_pars()`. The change point posteriors rarely conform to well-known distributions. Note that the posteriors were also plotted in Figure 2.

Directional hypotheses operate entirely on the posterior, comparing the proportion of posterior density (number of MCMC samples) above or below a particular value. This can be exploited to test interval hypotheses like ROPE (Kruschke, 2011), e.g., whether $x\_2 \in [0.2, 0.8]$. Here, the Bayes Factor is around $BF = 2.5$.

Hypotheses can also concern relationships between parameters, e.g., akin to two-sample t-tests. For example, we can test whether $\Delta_1 < \Delta_2/2$. The evidence seems to favor the alternative hypothesis by a factor of around $1/0.25 = 4$.

Since the values of the model's 7 parameters were inferred from just 50 data points, the likelihoods (and hence the posteriors) are relatively wide. They will tend to narrow with larger sample sizes resulting in more extreme (Savage-Dickey) Bayes Factors.

```
hypothesis(fit, c("x_2 = 0.5",
                  "x_2 > 0.2 & x_2 < 0.8",
                  "cp_1 < cp_2 / 2"))
```

```
##              hypothesis mean lower upper    p   BF
## 1         x_2 - 0.5 = 0 0.21 -0.15  0.59 0.86 5.95
## 2 x_2 > 0.2 & x_2 < 0.8   NA    NA    NA 0.70 2.33
## 3   cp_1 - cp_2 / 2 < 0 1.96 -3.26  6.02 0.20 0.26
```

4.5 Model comparison

`mcp` supports model comparison via cross-validation via the `loo` package (Vehtari et al., 2017), i.e., by comparing the out-of-sample predictive accuracy of models. Continuing our example, we can test the very existence of the first change point by comparing it to a null model where the first two segments are collapsed into one linear segment.

Briefly, `loo` sums the log-density of the posterior predictive a data point when the model is fitted on the remaining data points. This is repeated for all data points. Summing log-densities corresponds to multiplying the actual predictive densities. The Estimated Log-Predictive Density (ELPD) indicates how well the model predicts out-of-sample data. One would favor a model to the degree that it has superior predictive performance (greater ELPD) and the column `elpd_diff` shows this difference (Vehtari et al., 2017; Gelman et al., 2013). Like any model comparison approach, Bayesian cross-validation is often useful but there are (edge) cases where the predictive performance of a model is underestimated (Gronau and Wagenmakers, 2019). Hopefully, future versions of `mcp` will include more options for model comparison, e.g., via the `bridgesampling` package (Gronau et al., 2018), so that convergence between multiple methods can bolster inferential conclusions.

Here, the existence of the change point is favored over the null model but only weakly. Again, as the number of observed samples increase, the out-of-sample predictive performance of the models will be better discriminable.

```
# Define the null model
model_null = list(
  y ~ 1 + x,      # One linear segment
    ~ 1 + I(x^2)  # Disjoined quadratic
)
```

```
prior = list(x_1 = "dnorm(0, 3) T(0, )", int_2 = "int_1")  # Same prior

# Sample it
fit_null = mcp(model_null, data = df, prior = prior)
```

```
# Compute loos and compare
fit$loo = loo(fit)
fit_null$loo = loo(fit_null)
loo::loo_compare(fit$loo, fit_null$loo)  # Can take more loos
```

```
##        elpd_diff se_diff
## model1  0.0       0.0
## model2 -1.4       4.2
```

4.6 Apply to real data

The previous steps served to crystallize an analysis plan. For confirmatory research, it is recommended to pre-register the analysis plan and the associated code at this point (Munafò et al., 2017). Running the analysis on actual data is then merely a matter of "plugging it in" and running the analyses.

5 Strengths and limitations

5.1 Inferring change points and a note on intervals

`mcp` uses JAGS (Plummer, 2003) for inference and this is a notable deviation from other packages. Multiple change point problems with unknown parameters are analytically intractable ((Stephens, 1994; Carlin et al., 1992), but see (Jensen, 2013)). Therefore, change points are typically inferred using variations over two types of search algorithms:

- User-specified number of change points: iterate over possible locations of the K change points and return the fit that minimizes the cost of the $K + 1$ regression models.
- An unknown number of change points: Compute a statistic that is sensitive to changes. Return a change point every time the statistic passes a threshold. CUSUM (Page, 1954; Lee et al., 2003) is one such an algorithm which is sensitive to changes in the intercept.

These procedures are practical and fast. However, their procedural nature also means that the statistical properties of the inferred change points are unknown and hence confidence intervals cannot readily be computed. One exception is the `segmented` package which implements a score-based confidence interval (Muggeo, 2017). Care should be taken not to interpret frequentist intervals as a Bayesian posterior. The posteriors are often multi-modal and asymetric (see e.g. Figure 3, Figure 4 and (Raftery and Akman, 1986)). Therefore, simple intervals (whether frequentist or Bayesian) do not necessarily represent credible change point locations.

Change points can be inferred using a Gibbs sampler (and later optimizations) in the absence of an analytical solution (Stephens, 1994; Carlin et al., 1992). Since the advent of the BUGS language for Gibbs samplers in the 1990s (Lunn et al., 2012), Gibbs samplers have been used to identify change points in Generalized Linear

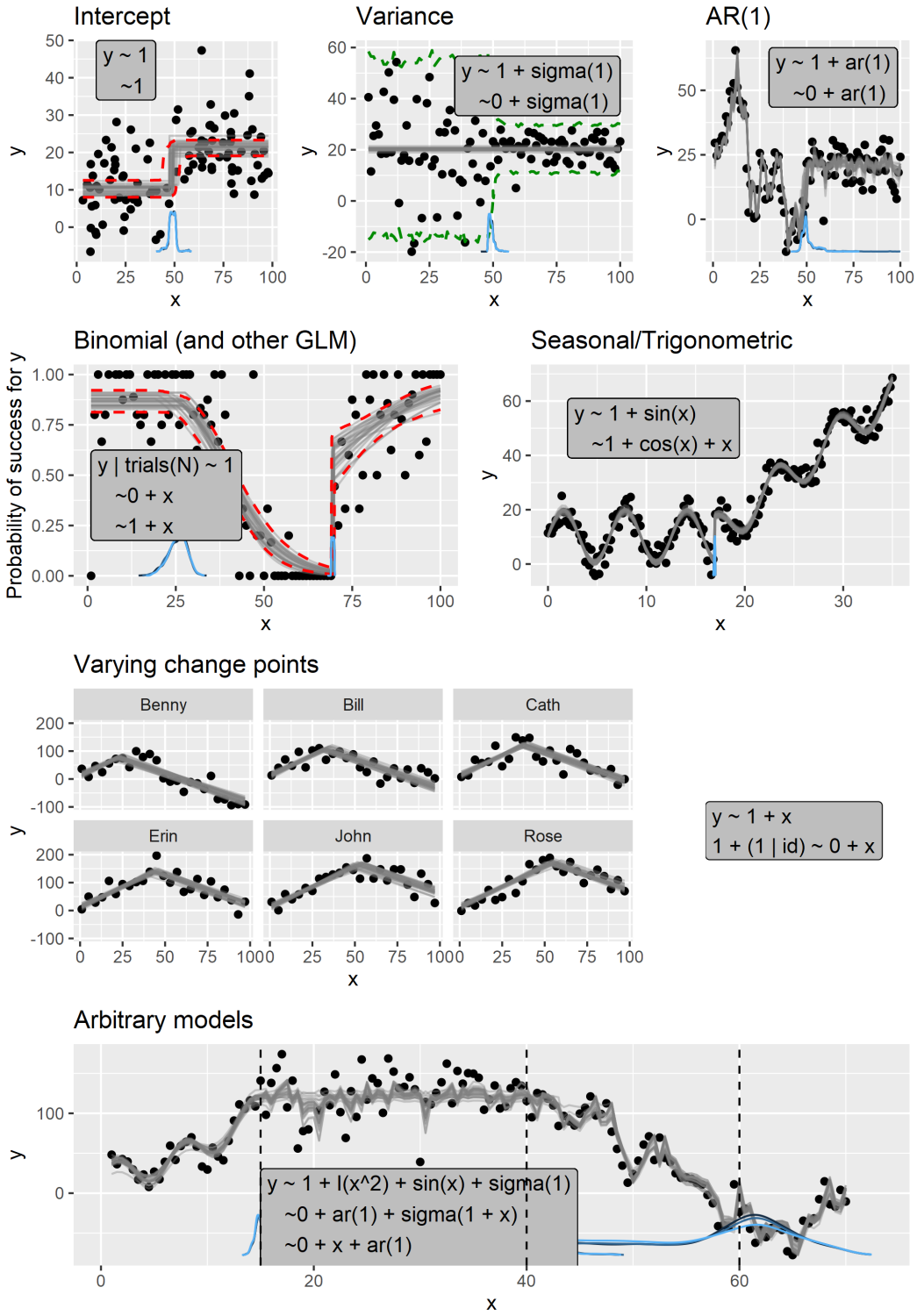

Figure 4: Some example mcp models with data (black dots), 25 posterior draws (grey lines), change point posteriors (blue densities), 95% fit intervals (red dashed lines), and 95% prediction intervals (green dashed lines). All plots were made with `plot()` and the data was simulated using the `simulate()` function. Other capabilities of `mcp` include Poisson regression and regression on `sigma()`/`ar()` (see the section on this API).

Mixed Models (GLMM) and Time Series, among many others. I refer to(Stephens, 1994; Carlin et al., 1992) for a detailed exposition of how Gibbs samplers infer change points.

`mcp` uses JAGS (Plummer, 2003) but is built with an abstraction layer that represents the model in a sampler-agnostic way so that other samplers can easily be supported in the future. JAGS was chosen for the initial versions of `mcp` because sampling starts immediately, the install procedure is similar across operating systems, and the JAGS code is highly similar to R code, making it easier for R users to inspect and learn. `stan` (Carpenter et al., 2017) may be supported in the future since it is faster once the model has compiled and is expected to be superior for large datasets, complex models, and the Dirichlet prior.

5.2 Strengths

The mcp website contains a detailed comparison of R packages for multiple change point analysis, including `mcp`, `segmented` (Muggeo, 2008), `strucchange` (Zeileis et al., 2002, Zeileis et al. (2003)), `ecp` (James and Matteson, 2015), `bcp` (Erdman and Emerson, 2007), `changepoint` (Killick and Eckley, 2014), `changepoint.np` (Haynes and Killick, 2019), `TSMCP` (Li and Jin, 2018), `cpm` (Ross, 2015), `EnvCpt` (Killick et al., 2018), `wbsts` (Korkas and Fryzlewicz, 2018), and others.

`mcp` is unique in the following ways: `mcp` supports a larger number of regression models while most packages model either (joined) slope changes or intercept-only changes. mcp allows regression models to differ between segments while other packages assume identical segment structures. `mcp` is the only package to model varying change points and to detect changes in autoregression. It is also the only package that can do regression on σ (variance) and ψ (autocorrelation) for change point problems. `mcp` is the only package to compute posteriors for the change points (and other parameters). Lastly, `mcp` can simulate data for all these models and directly assess parameter recovery.

`mcp` provides many plot options and returns the plots as `ggplot2` objects so that the user can customize the plots further.

`mcp` can include prior knowledge about individual parameters, model constant and identical coefficients (e.g., shared between segments), and do point-, directional-, and interval-tests on individual parameters. Because of this modeling flexibility, a greater number of models can be compared. Taken together, we may say that the strength of `mcp` is for *informed* analysis, i.e., when prior knowledge can be flexibly incorporated as specific model(s) or prior distributions on parameters.

When applying all change point packages to a simple three-intercept problem, half of the packages found the two change points without quantifying uncertainty (`EnvCpt`, `strucchange::breakpoints`, `cpm`, `changepoint.np`, and `ecp::e.divisive`), while the other half did not recover them (`segmented`, `changepoint`, most methods in `ecp`, `TSMCP`, and `wbsts`).

`mcp` aims to be user-friendly. It includes more than 100 stop conditions with helpful error messages. For example, if JAGS throws an error (e.g., when the user-provided prior results in a directed cycle), this error is returned along with an empty `mcpfit` object containing the model code so that it can be inspected by the user. The test suite for `mcp` currently includes more than 1.700 tests that cover 88% of the code, reducing the chance that the end-user encounters a bug.

The source code is hosted on GitHub. It is less than 2.000 lines making `mcp` relatively lightweight to maintain, extend, and collaborate on.

5.3 Limitations

Where `mcp` falls short, I find that `segmented`, `EnvCpt`, and `bcp` are recommendable packages if the data conforms to the smaller range of models supported by these packages.

`mcp` cannot detect the number of change points in a dataset. `mcp` can use `loo` to compare N models, e.g., with $K = 1, 2, \ldots, N$ change points but this grows intractable for large K.

Due to the order-restriction of the change point priors in `mcp`, each prior quickly becomes highly localized for larger K. Specifically, the posteriors may "shrink" towards these priors in undesirable ways if the data set is not large enough that the likelihood will dominate the prior, e.g., $K = 10$ for 100 data points. While `mcp` quantifies the credence that a *given* change point occurs at x_i, questions for large K may better be posed as the credence that *any* change point occurs at x_i.

`mcp` is the slowest of the reviewed packages and may be too computationally heavy for time-critical analyses of large datasets, e.g., 100.000 data points. To speed up inference, `mcp` supports parallel processing via the `future` and `future.apply` packages (Bengtsson, 2019). Furthermore, fewer posterior samples are likely needed for large datasets.

Lastly, `mcp` does not currently model multivariate data (`ecp`, `bcp`, or `TSMCP` are recommended). Furthermore, `mcp` does not currently support survival models, Moving-Average/ARIMA autocorrelation, or more than one predictor variable. `segmented` may be superior in these respects.

6 Conclusion

`mcp` aims to fill in a missing piece in the change point world: flexible per-segment regression models with rich inference information. In concert with the packages doing automatic detection of the number of change points, along with more efficient (but also more limited) packages for a known number of change points, R users now have a great plethora of options that should cover most needs.

7 Appendix

7.1 Marginal Dirichlet Priors

Before shifting and scaling, the prior for change point k of total K change points will correspond to:

$$\Delta_k \sim \text{Beta}(\sum_{z=1}^{K} \alpha_z, \sum_{z=k+1}^{K+1} \alpha_z)$$

When all $\alpha = 1$, this simplifies to:

$$\Delta_k \sim \text{Beta}(k, K + 1 - k)$$

##Likelihood for a multiple change point model Generalizing equation (2) from Carlin et al. (1992) to multiple change points and inserting (2) and (3), we get the likelihood:

$$\mathcal{L}(\mathbf{y}) = \prod_{k=1}^{K} \prod_{i=\Delta_{k-1}}^{\Delta_k} [x_i > \Delta_k] \cdot P_k(\max\{x_i, \Delta_k\} - \Delta_{k-1} | \beta_{\mathbf{k}})$$

where P_k is the density associated with f_k and the family in question (e.g., Gaussian or Binomial).

References

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, 67(1).

Bengtsson, H. (2019). Future: Unified parallel and distributed processing in r for everyone.

Bürkner, P.-C. (2017). Brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1):1–28.

Bürkner, P. C. and Charpentier, E. (2018). Modeling Monotonic Effects of Ordinal Predictors in Bayesian Regression Models. Preprint, PsyArXiv.

Camos, V. and Tillmann, B. (2008). Discontinuity in the enumeration of sequentially presented auditory and visual stimuli. *Cognition*, 107(3):1135–1143.

Carlin, B. P., Gelfand, A. E., and Smith, A. F. M. (1992). Hierarchical Bayesian Analysis of Changepoint Problems. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(2):389–405.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, 76(1).

Chen, J. and Gupta, A. K. (1997). Testing and Locating Variance Changepoints with Application to Stock Prices. *Journal of the American Statistical Association*, 92(438):739–747.

Conn, P. B., Johnson, D. S., Williams, P. J., Melin, S. R., and Hooten, M. B. (2018). A guide to Bayesian model checking for ecologists. *Ecological Monographs*, 88(4):526–542.

Cowan, N. (2000). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and brain sciences*, 24(01):87–114. 00000.

Erdman, C. and Emerson, J. W. (2007). Bcp: An R Package for Performing a Bayesian Analysis of Change Point Problems. *Journal of Statistical Software*, 23(1):1–13.

Fuller, W. A. and Hasza, D. P. (1981). Properties of Predictors for Autoregressive Time Series. *Journal of the American Statistical Association*, 76(373):155–161.

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., and Gelman, A. (2019). Visualization in Bayesian workflow. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 182(2):389–402.

Gelman, A., Hwang, J., and Vehtari, A. (2013). Understanding predictive information criteria for Bayesian models. *arXiv:1307.5928 [stat]*.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2018). Bridgesampling: An R Package for Estimating Normalizing Constants. *arXiv:1710.08162 [stat]*.

Gronau, Q. F. and Wagenmakers, E.-J. (2019). Limitations of Bayesian Leave-One-Out Cross-Validation for Model Selection. *Computational Brain & Behavior*, 2(1):1–11.

Haynes, K. and Killick, R. (2019). Changepoint.np: Methods for nonparametric changepoint detection.

James, N. A. and Matteson, D. S. (2015). Ecp: An R Package for Nonparametric Multiple Change Point Analysis of Multivariate Data. *Journal of Statistical Software*, 62(1):1–25.

Jensen, G. (2013). Closed-Form Estimation of Multiple Change-Point Models. Preprint, PeerJ PrePrints.

Killick, R., Beaulieu, C., Taylor, S., and Hullait, H. (2018). EnvCpt: Detection of structural changes in climate and environment time series.

Killick, R. and Eckley, I. A. (2014). Changepoint: An R Package for Changepoint Analysis. *Journal of Statistical Software*, 58(1):1–19.

Knuth, D. E. (1992). Two Notes on Notation. *The American Mathematical Monthly*, 99(5):403–422.

Korkas, K. and Fryzlewicz, P. (2018). Wbsts: Multiple change-point detection for nonstationary time series.

Kruschke, J. K. (2011). Bayesian assessment of null values via parameter estimation and model comparison. *Perspectives on Psychological Science*, 6(3):299–312.

Lee, S., Ha, J., Na, O., and Na, S. (2003). The Cusum Test for Parameter Change in Time Series Models. *Scandinavian Journal of Statistics*, 30(4):781–796.

Leibovich-Raveh, T., Lewis, D. J., Kadhim, S. A.-R., and Ansari, D. (2018). A New Method for Calculating Individual Subitizing Ranges. *Journal of Numerical Cognition*, 4(2):429–447.

Li, Y. and Jin, B. (2018). TSMCP: Fast two stage multiple change point detection.

Lindeløv, J. (2018). Using performance discontinuities to estimate individual Working-Memory Capacities in serial recall tasks. *Journal of Vision*, 18(10):698–698.

Lunn, D., Jackson, C., Best, N., Spiegelhalter, D., and Thomas, A. (2012). *The BUGS Book: A Practical Introduction to Bayesian Analysis*. Chapman and Hall/CRC.

Martino, L., Elvira, V., and Louzada, F. (2017). Effective Sample Size for Importance Sampling based on discrepancy measures. *Signal Processing*, 131:386–401.

Muggeo, V. M. (2008). Segmented: An R package to fit regression models with broken-line relationships. *R news*, 8(1):20–25.

Muggeo, V. M. R. (2017). Interval estimation for the breakpoint in segmented regression: A smoothed score-based approach. *Australian & New Zealand Journal of Statistics*, 59(3):311–322.

Munafò, M. R., Nosek, B. A., Bishop, D. V. M., Button, K. S., Chambers, C. D., du Sert, N. P., Simonsohn, U., Wagenmakers, E.-J., Ware, J. J., and Ioannidis, J. P. A. (2017). A manifesto for reproducible science. *Nature Human Behaviour*, 1(1):1–9.

Page, E. S. (1954). Continuous Inspection Schemes. *Biometrika*, 41(1/2):100–115.

Pesti, D. G. M., Vedenov, D., Cason, J. A., and Billard, L. (2009). A comparison of methods to estimate nutritional requirements from experimental data. *British Poultry Science*, 50(1):16–32.

Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, volume 124, page 125. Vienna, Austria.

Plummer, M., Best, N., Cowles, K., and Vines, K. (2006). CODA: Convergence diagnosis and output analysis for MCMC. *R news*, 6(1):7–11.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. Vienna, Austria.

Raftery, A. E. and Akman, V. E. (1986). Bayesian analysis of a Poisson process with a change-point. *Biometrika*, 73(1):85–89.

Ross, G. J. (2015). Parametric and Nonparametric Sequential Change Detection in R: The cpm Package. *Journal of Statistical Software*, 66(1):1–20.

Stephens, D. A. (1994). Bayesian Retrospective Multiple-Changepoint Identification. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 43(1):159–178.

Vehtari, A., Gelman, A., and Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, 27(5):1413–1432.

Verdinelli, I. and Wasserman, L. (1995). Computing Bayes Factors Using a Generalization of the Savage-Dickey Density Ratio. *Journal of the American Statistical Association*, 90(430):614–618.

Zeileis, A., Kleiber, C., Krämer, W., and Hornik, K. (2003). Testing and dating of structural changes in practice. *Computational Statistics & Data Analysis*, 44(1-2):109–123.

Zeileis, A., Leisch, F., Hornik, K., and Kleiber, C. (2002). Strucchange: An R Package for Testing for Structural Change in Linear Regression Models. *Journal of Statistical Software*, 7(1):1–38.