Using ddcurves2 to fit density profile models

Andrew Manderson

02 August, 2018

ddcurves2 is an R package for fitting the hierarchical model detailed in the forthcoming publication "Statistical estimation of temporal variability in vertical seawater density profiles". It also has another model for temporal variation of background density profiles, that may be discussed at a later date.

Overview of this document

There are three topics this document covers:

- The dependencies required by ddcurves2, some basic instructions on installing them, and where to documentation in case of installation difficulties.
- Fitting the hierarchical model to the data presented in the publication
- Demonstrating some of the built in plotting functionality inside the package.

Installation

R

You will need an installation of R (https://www.r-project.org/), and I recommend using RStudio (https://www.rstudio.com/) to run R. Both of these websites have installation documentation should you run into difficulties.

R packages

There are a number of R packages we require, which we can install with `install.packages()`. For now, we require `devtools`, `ggplot2` and `Rcpp`, which can be installed within R with the following command:

```
install.packages(c("devtools", "ggplot2", "Rcpp"))
```

Stan and rstan

This package makes use of Stan to fit the models. Installing it can be somewhat of a pain. The documentation for installing Stan and its dependencies is fairly comprehensive, and is available for either Windows or OSX / Linux at:

- Windows: https://github.com/stan-dev/rstan/wiki/Installing-RStan-on-Windows
- OSX / Linux: https://github.com/stan-dev/rstan/wiki/Installing-RStan-on-Mac-or-Linux

If the OSX installation instructions do no work for you, I can recommend running `xcode-select --install` from the terminal to get the c++ compiler (often the most error prone step), or checking the Stan forums (http://discourse.mc-stan.org/) for similar issues.

ddcurves2

Finally we can install ddcurves2, which makes use of `devtools::install_github()` to install the most recent version off of github.

```
devtools::install_github("hhau/ddcurves2") # should be public by the time this goes out.
```

iwaves

To make some of the figures in the paper, we make use of the python package iwaves: (https://bitbucket.org/mrayson/iwaves). Installation instructions can be found on the bitbucket page.

Running the hierarchical model

The data presented in the paper is included in the package. It is split into the two halves. We fit it as two separate models, as it makes minimal difference to the estimated posterior distributions of each curve, and the data has a variable number of depths at which density data is recorded.

For the first half of the data, we load the library and create the list of data to give to the fitting function, and call the fitting function with some additional options to Stan:

```
library(ddcurves2)
```

```
Warning: package 'Rcpp' was built under R version 3.5.1
```

```
stan_dat_one <- list(
  n_depths = length(Crux_KP150_Phs1$depths),
  n_times = nrow(Crux_KP150_Phs1$density_mat),
  depths = Crux_KP150_Phs1$depths,
  densities = Crux_KP150_Phs1$density_mat
)

# model needs initial values
n_t_one <- stan_dat_one$n_times
init_sub_one <- list(
  beta_zero = rep(1025, n_t_one),
  beta_one = rep(1, n_t_one),
  beta_three = rep(42, n_t_one),
  beta_six = rep(48, n_t_one),
  beta_midpoint = array(c(rep(77, n_t_one), rep(150, n_t_one)), dim = c(n_t_one, 2)),
  mean_beta_zero = 1025,
  mean_beta_one = 1,
  mean_beta_three = 42,
  mean_beta_six = 48,
  mean_beta_midpoint = array(c(77, 150), dim = 2),
  sigma_beta = array(c(0.18, 0.12, 26.1, 8.09, 12.6, 6.65), dim = 6),
  sigma_curve = 0.08
)

model_fit_one <- double_tanh_no_timeseries(
  stan_data_list = stan_dat_one,
  iter = 2500,
  warmup = 2000,
  save_warmup = FALSE,
  chains = 3,
  cores = 3,
  refresh = 25,
  control = list(adapt_delta = 0.95, max_treedepth = 16, stepsize = 1e-6),
  init = list(init_sub_one, init_sub_one, init_sub_one)
)
saveRDS(model_fit_one, file = "model-fit-one.rds")
```

And the same for the second half of the data:

```
stan_dat_two <- list(
  n_depths = length(Crux_KP150_Phs2$depths),
  n_times = nrow(Crux_KP150_Phs2$density_mat),
  depths = Crux_KP150_Phs2$depths,
  densities = Crux_KP150_Phs2$density_mat
)

n_t_two <- stan_dat_two$n_times
init_sub_two <- list(
  beta_zero = rep(1025, n_t_two),
  beta_one = rep(1, n_t_two),
  beta_three = rep(42, n_t_two),
  beta_six = rep(48, n_t_two),
  beta_midpoint = array(c(rep(77, n_t_two), rep(150, n_t_two)), dim = c(n_t_two, 2)),
  mean_beta_zero = 1025,
```

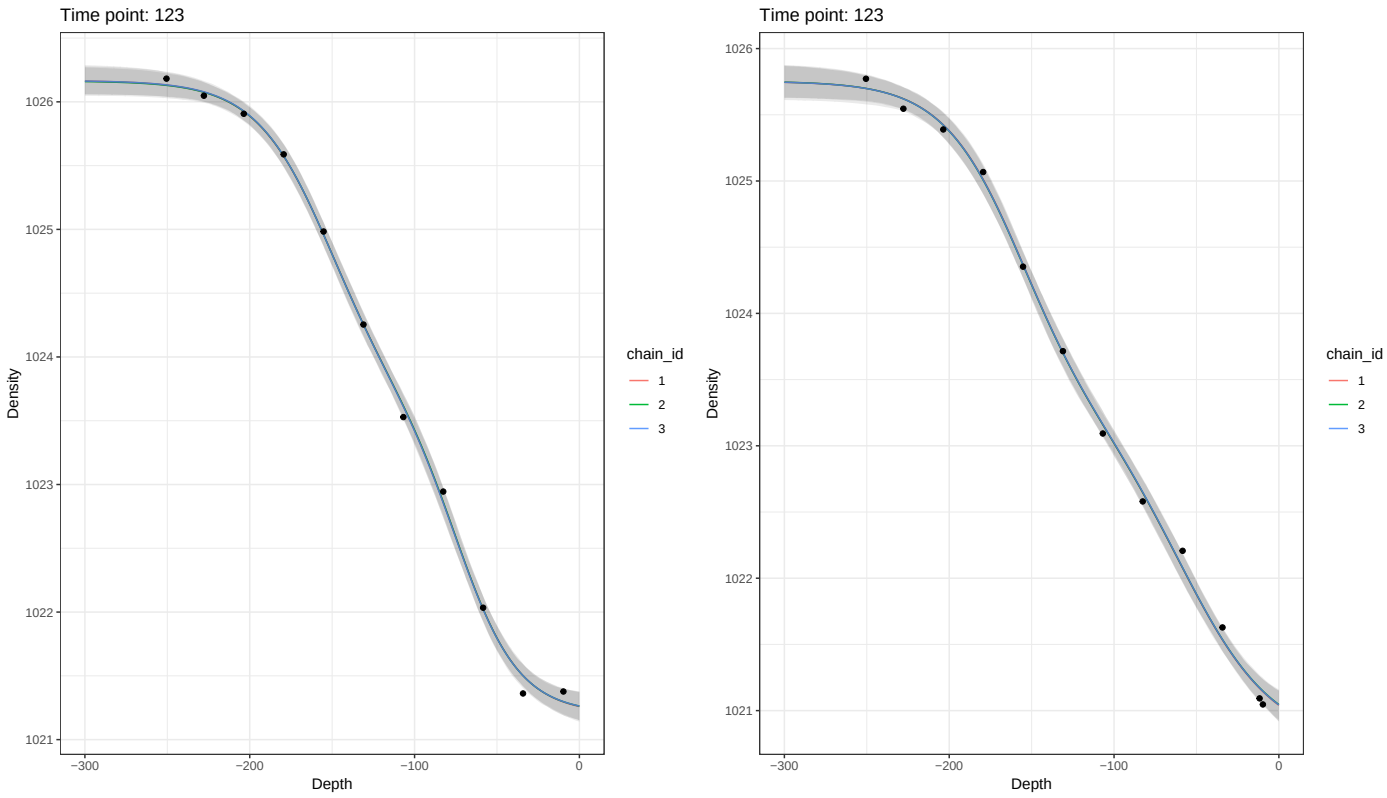
```
  mean_beta_one = 1,
  mean_beta_three = 42,
  mean_beta_six = 48,
  mean_beta_midpoint = array(c(77, 150), dim = 2),
  sigma_beta = array(c(0.18, 0.12, 26.1, 8.09, 12.6, 6.65), dim = 6),
  sigma_curve = 0.08
)

model_fit_two <- double_tanh_no_timeseries(
  stan_data_list = stan_dat_two,
  iter = 2500,
  warmup = 2000,
  save_warmup = FALSE,
  chains = 3,
  cores = 3,
  refresh = 25,
  control = list(adapt_delta = 0.95, max_treedepth = 16, stepsize = 1e-6),
  init = list(init_sub_two, init_sub_two, init_sub_two)
)
saveRDS(model_fit_two, file = "model-fit-two.rds")
```

Some output plots

Estimated density profiles

For a given time index, we can look at the estimated density profile:

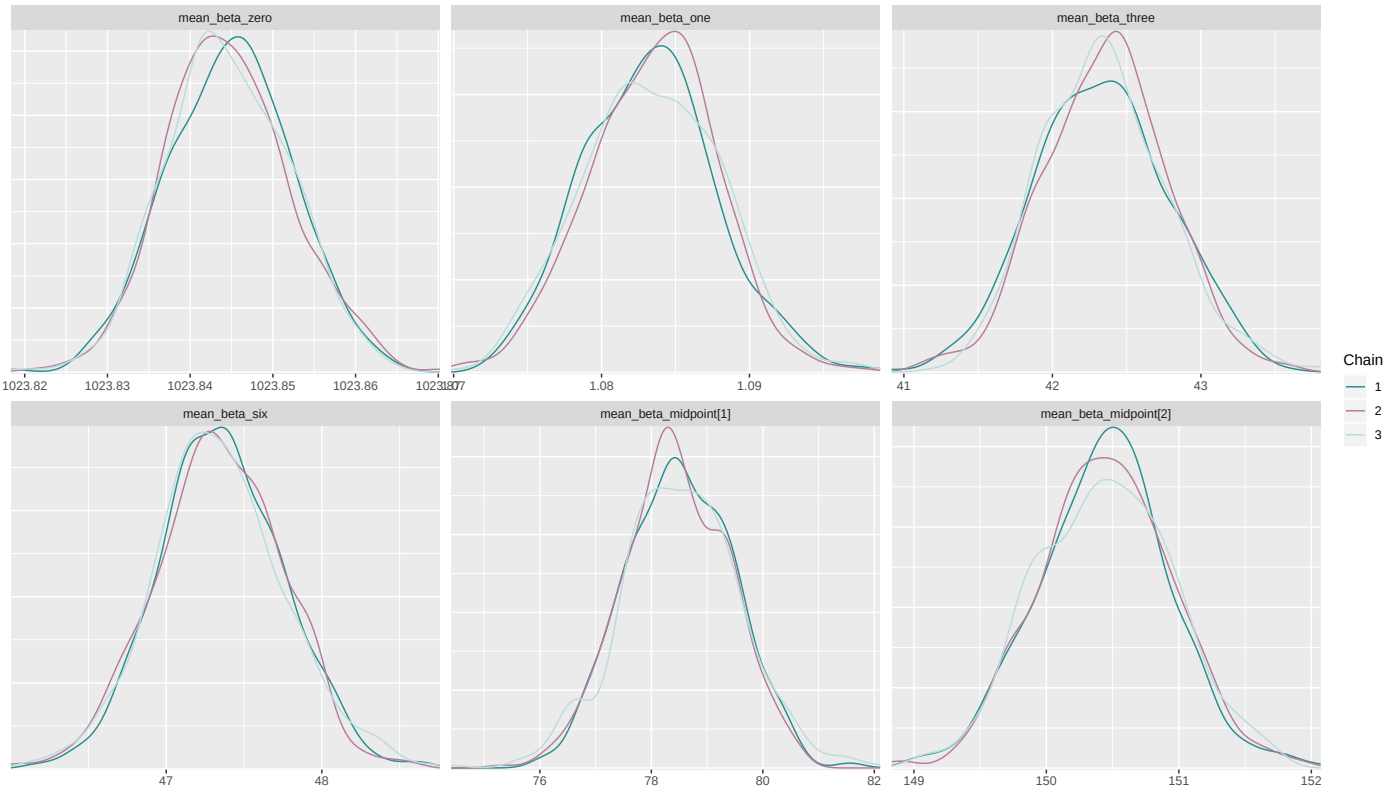
```
t_index <- 123

dens_plot_one <- ddcurves2::double_fitted_curve_plotter(
  model_fit_one,
  t_index,
  stan_dat_one
)

dens_plot_two <- ddcurves2::double_fitted_curve_plotter(
  model_fit_two,
  t_index,
  stan_dat_two
)

library(gridExtra)

# put both plots together
grid.arrange(arrangeGrob(
  dens_plot_one,
  dens_plot_two,
  nrow = 1
))
```



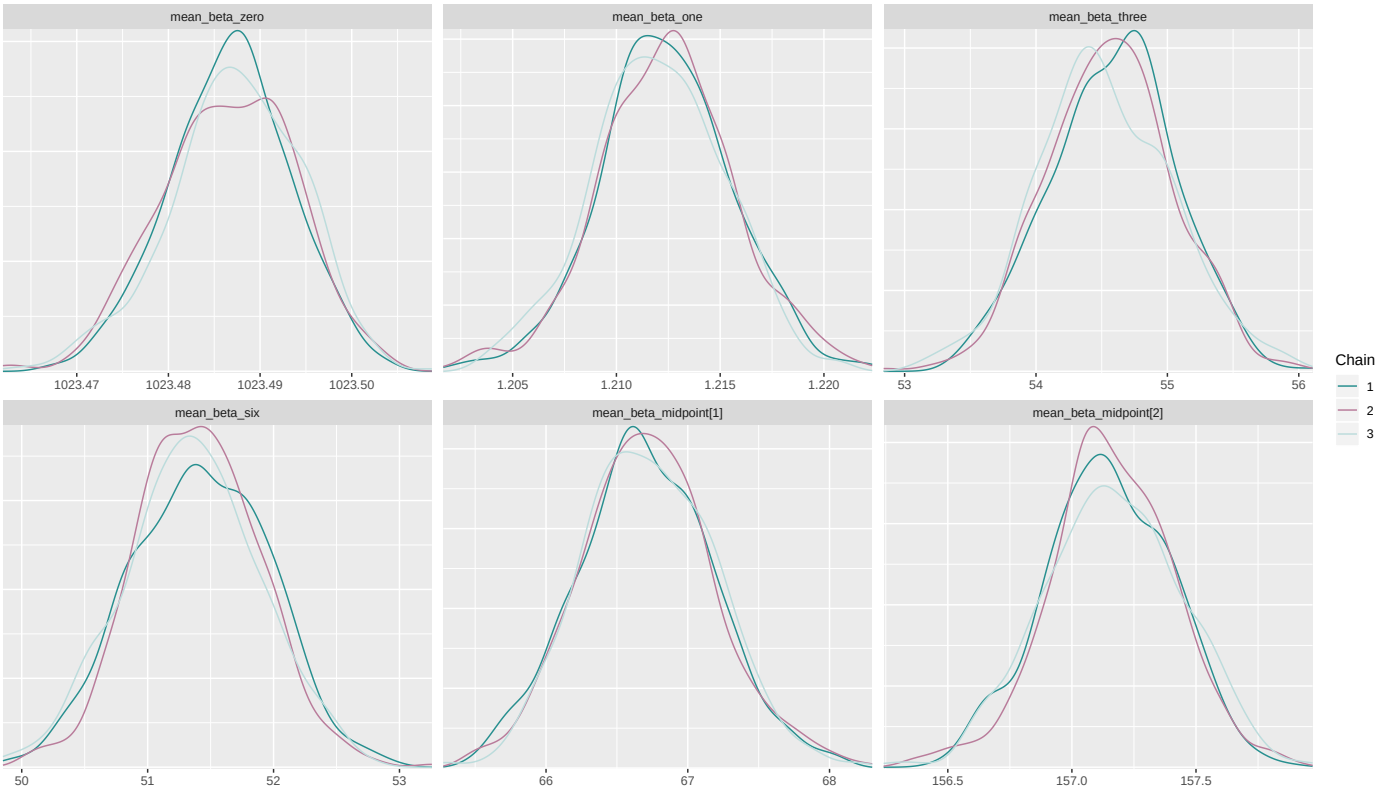
Here we are looking at the 123rd observation from the start of the first and second halves of the overall data set.

We can also look at the distribution of the means of the parameters of the curve, for which we will only plot the first model:

```
mean_beta_density_diag(model_fit_one)
```




```
mean_beta_density_diag(model_fit_two)
```



For more examples of the R package, as well as how we use the python package, we direct the reader to the code that produces the figures in the paper.