Functionality of the FoSIntro package

Alexander Bauer

2017-10-23

Contents

1 Introduction

This vignette gives an overview of the functionality provided with the `FoSIntro` `R` package, which is the accompanying package to Bauer et al. (2017) - ''An Introduction to Semiparametric Function-on-scalar Regression".

All examples from the paper can be reproduced using the code in the `_examples` folder on the GitHub repository. This vignette instead does not reproduce each single plot of the paper, but instead gives an overview of the functions provided by the package, which are mainly convenient effect/prediction/residual plot functions for function-on-scalar models fitted with `pffr` (from the `refund` package), but also comprise functions to perform (non)parametric bootstrapping to retrieve confidence intervals for smooth effects from `pffr` models. Additionally, for the sake of completeness, also the overview of how to apply all hypothesis tests listed in the paper is given here.

More information on the package can be found in the GitHub repository.

2 Preparations

2.1 General preparations

```
# Install the FoSIntro package from GitHub
# devtools::install_github("bauer-alex/FoSIntro")

library(FoSIntro)
library(refund)
library(mgcv)
library(ggplot2)
library(dplyr)
library(tidyr)
library(magrittr)
```

```
theme_set(theme_bw()) # set ggplot2 theme
```

2.2 Data preparation

```
data <- FoSIntro::sample_data
yindex <- attr(data, "yindex")

# Manually create dummy variables for factor variables
# (currently necessary as pffr v.0.1-16 doesn't dummy code categorical variables correctly)
data <- cbind(data, with(data, model.matrix(~velocity))[,-1,drop=FALSE])

# reshape data for some visualizations
data_wide <- data %>%
  select(-ground_velocity) %>%
  bind_cols(data$ground_velocity) %>%
  gather(key = "time", value = "ground_velocity", starts_with("y")) %>%
  mutate(time = as.numeric(substring(time, 3)))
```

2.3 Fit the function-on-scalar model with pffr()

```
model <- pffr(ground_velocity ~ s(hypocentral_distance) + s(stat_friction_coef) +
                c(dyn_friction_coef) + c(s(slip_weakening)) +
                dir_background_stress + velocitysediment,
              family = Tweedie(p = 2, link = "log"), yind = yindex, data = data)
```

3 Effect plots

Note: 1D and 2D smooth effects can both be plotted using the default plot.pffr() function (based on plot.gam()). For better customization however we use our own functions based on ggplot2.

```
ylim <- c(-2.25,3) # use same effect scale for all effects for easier interpretation
# time-constant effect of slip weakening
plot_1D(model, select = 4, ylim = ylim, ylab = "estimate")
# time-varying effect of the direction of the background stress
plot_1D(model, select = 5, xlab = "time [s]", ylim = ylim, ylab = "estimate")
# time-varying effect of hypocentral distance
plot_2Dheatmap(model, select = 2, plot_ci = FALSE,
               xlab = "hypocentral distance", ylab = "time [s]", legend_limits = ylim)
plot_2D(model, select = 2,
        xlab = "hypocentral distance", ylab = "time [s]", zlab = "estimate")
```

4 Uncertainty quantification

Note: Currently only pointwise confidence/prediction intervals can be calculated with our code. A method for computing simultaneous/global confidence bands (and analogously intervalwise bands and global/intervalwise prediction bands) is shown in Krivobokova et al. (2010).

4.1 Confidence intervals after Marra & Wood (2012)

```r
# for 1D smooth effects
plot_1D(model, select = 4)
# for 2D smooth effects
plot_2Dheatmap(model, select = 2,
               xlab = "hypocentral distance", ylab = "time [s]")
```

4.2 Bootstrapped confidence intervals

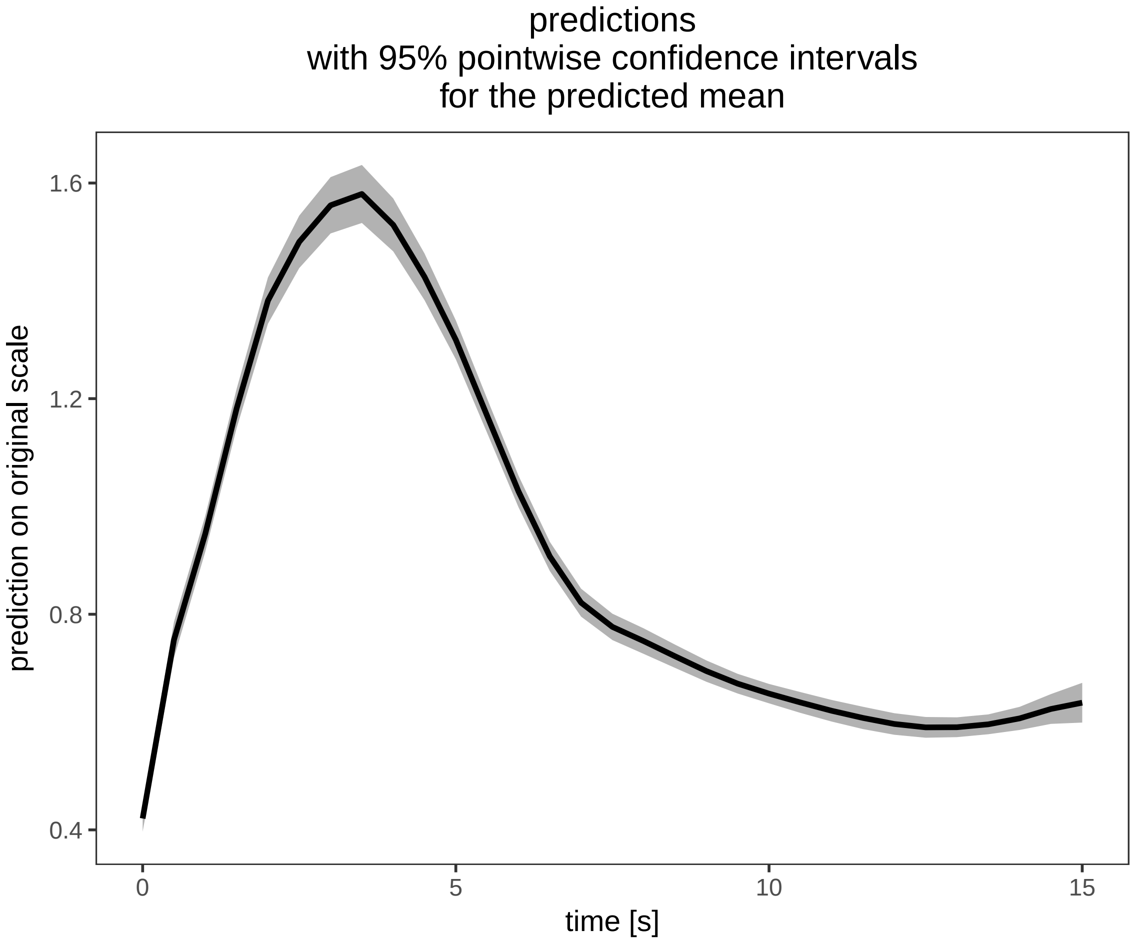
```r
B <- 1000 # bootstrap iterations (note that this takes several hours - for the
# vignette we use precalculated results)
n_cores <- 1 # for parallel computation (possible both for Linux-based systems and Windows)
formula <- ground_velocity ~ s(hypocentral_distance) + s(stat_friction_coef) +
  c(dyn_friction_coef) + c(s(slip_weakening)) + dir_background_stress +
  velocitysediment
```

```r
# Possibility 1: Parametric Bootstrap
bs <- bootstrap_pffr("parametric", B = B, cores = n_cores,
                     formula, data, model = model,
                     family = Tweedie(p = 2, link = "log"), yind = yindex,
                     param_yvar = "ground_velocity", param_simFun = "gamma", param_yMinValue = 0.01,
                     log_file = "bootstrap.log") # useful for parallel computation
# Possibility 2: Nonparametric bootstrap
bs <- bootstrap_pffr("nonparametric", B = B, cores = n_cores,
                     formula, data, model = model,
                     family = Tweedie(p = 2, link = "log"), yind = yindex,
                     log_file = "bootstrap.log") # useful for parallel computation
# Prepare bs for plotting
CIs_list <- calc_bootstrapCIs(bs, 0.05)
```

```r
plot_bootstrapCIs(model, CIs_list, select = 6, xlab = "time [s]", ylab = "estimate",
                  effect_label = "Time-varying effect of sediment velocity")
plot_bootstrapCIs(model, CIs_list, select = 2, ylab = "time [s]")
```

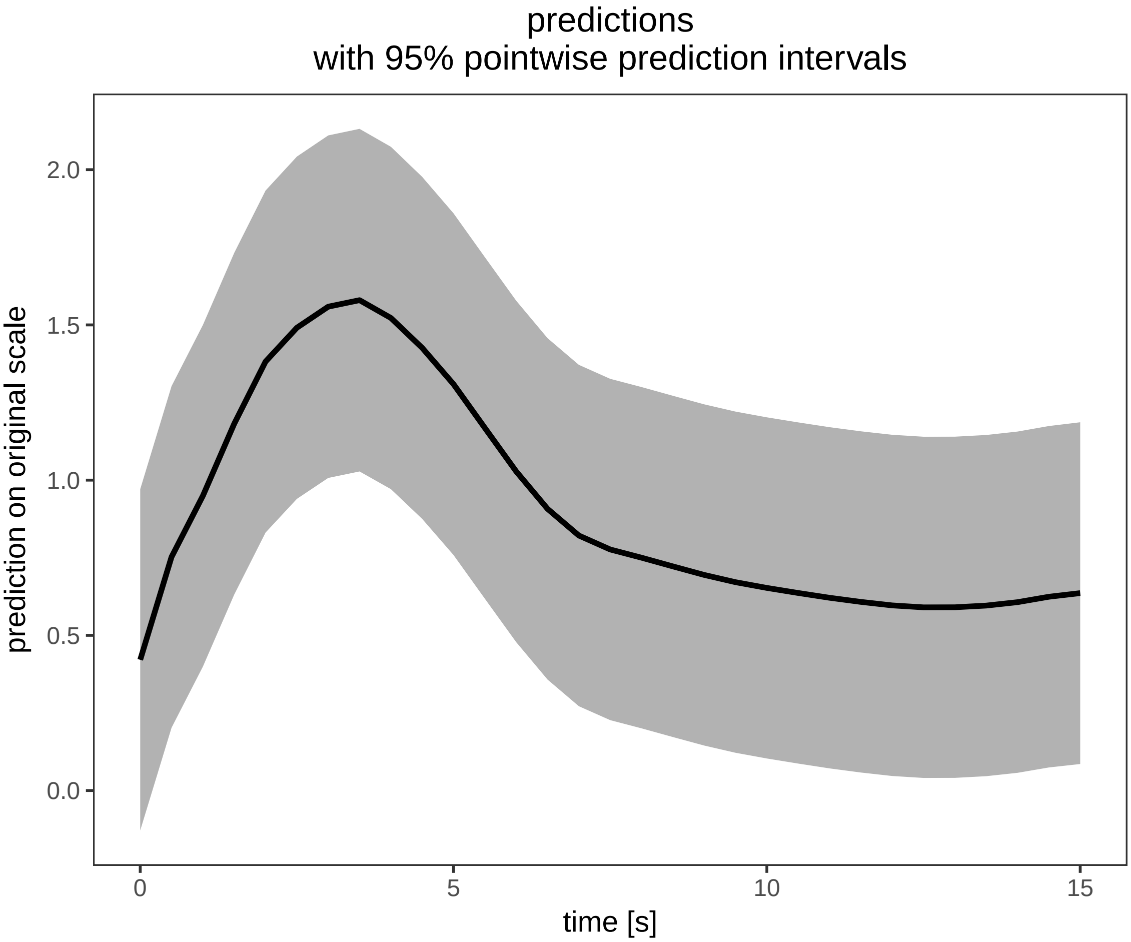
4.3 Confidence intervals for the predicted mean

```
newdata <- data[1,colnames(data) != "ground_velocity"]
plot_predictions(model, newdata, log10 = FALSE, ci_type = "ci",
                 xlab = "time [s]", ylab = "prediction on original scale")
```



4.4 Prediction intervals

```
plot_predictions(model, newdata, log10 = FALSE, ci_type = "pi",
                 xlab = "time [s]", ylab = "prediction on original scale")
```



5 Hypothesis testing

As noted in the text we didn't report p-values as our quite high-dimensional dataset led to all p-values being <0.0001. However, the following code shows how the p-values for all shown hypothesis tests can be extracted/computed.

```
# Save the model summary as we will extract some p-values from it
sm <- summary(model)
```

Likelihood-Ratio test

```
# Likelihood ratio test
# Example: Test if hypocentral_distance has a time-varying effect by comparing
#          the full model to 'model_2', which only contains a time-constant
#          effect of hypocentral distance.
# Note: as of a current bug in pffr v.0.1-16 we have to overwrite the class of
#       a pffr-object in order to make the anova.gam() function work properly
```

```
model_2 <- pffr(ground_velocity ~ c(s(hypocentral_distance)) + s(stat_friction_coef) +
                  c(dyn_friction_coef) + c(s(slip_weakening)) + dir_background_stress +
                  velocitysediment,
                family = Tweedie(p = 2, link = "log"), yind = yindex, data = data)
```

```
mtest <- model; mtest_2 <- model_2
class(mtest) <- class(mtest)[-1]
class(mtest_2) <- class(mtest_2)[-1]
anova(mtest, mtest_2, test = "F")
```

Other tests

```
### 1) Is the linear effect different from zero?
# 1a) for a metric or binary variable
sm$p.table
sm$p.pv["dyn_friction_coef"] # extract the p-value in 'sm$p.table' for one variable
# 1b) for a categorical variable with >2 categories
# -> Likelihood ratio test (see above)

### 2) Is the smooth effect different from zero?
# 2a) Globally
sm$s.table
sm$s.pv[2] # extract the p-value in 'sm$s.table' for one variable
# 2b) At a specific point
# -> Use bootstraped confidence intervals
# 2c) In a specific interval
# -> Use bootstrapped confidence intervals

### 3) Is at least one of multiple parameters different from zero?
# -> Likelihood ratio test (see above)

### 4) Are two linear effects different from one another?
# see explanation in paper
```

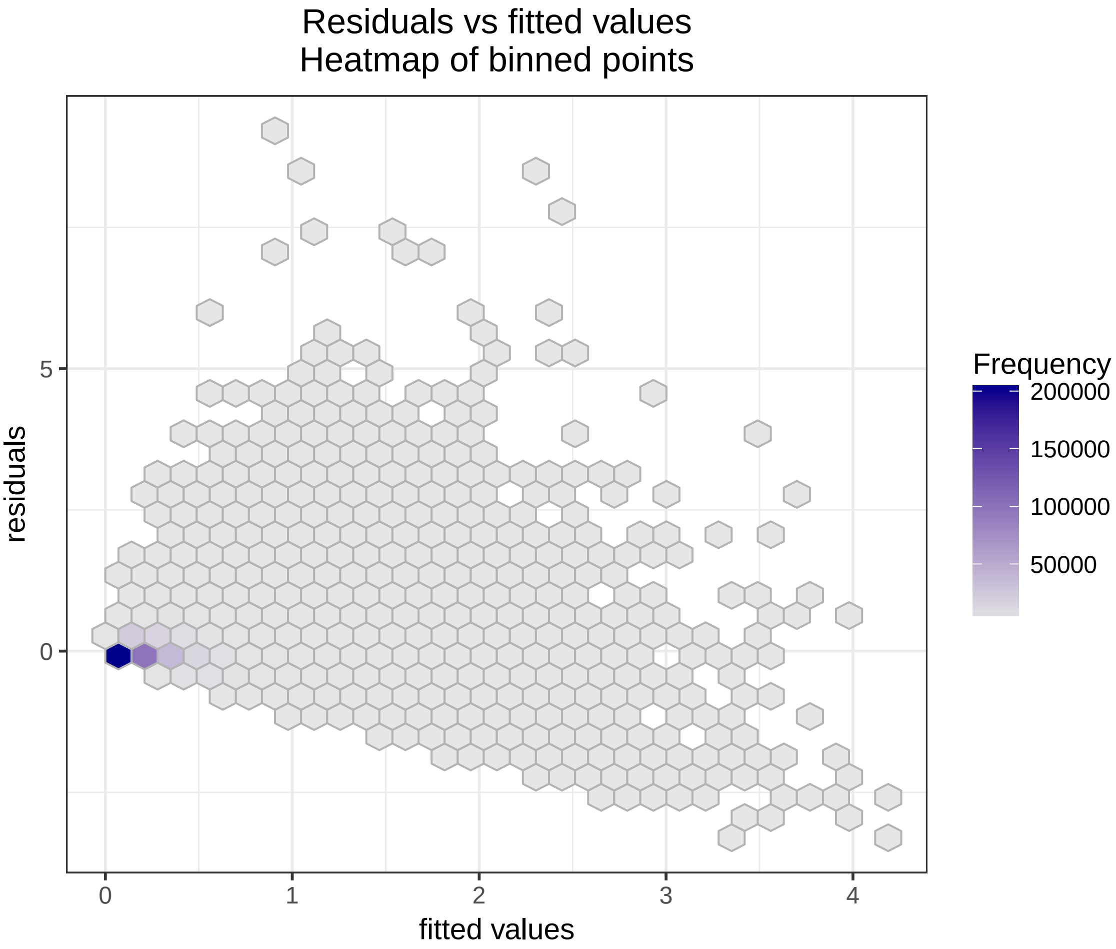
```
### 5) Are two smooth effects different from one another?
# see explanation in paper

### 6) Is the linear effect different depending on another variable?
# 6.1) both variables are metric or binary
#      6.1a) x_k is binary or the effect is varying linearly over the metric x_k
#      -> see hypothesis 1a)
#      6.1b) the effect is varying nonlinearly over the metric x_k
#      -> Likelihood ratio test (see above)
# 6.2) at least one of the variables is categorical with >2 categories
# -> Likelihood ratio test (see above)

### 7) Is the smooth effect different depending on another variable?
# 7a) x_k is binary
# -> see 2a)
# 7b) x_k is metric or categorical with >2 categories
# -> Likelihood ratio test (see above)

### 8) Is one model better than another model?
# -> Likelihood ratio test (see above)
```

6 Model evaluation

6.1 Residual plots

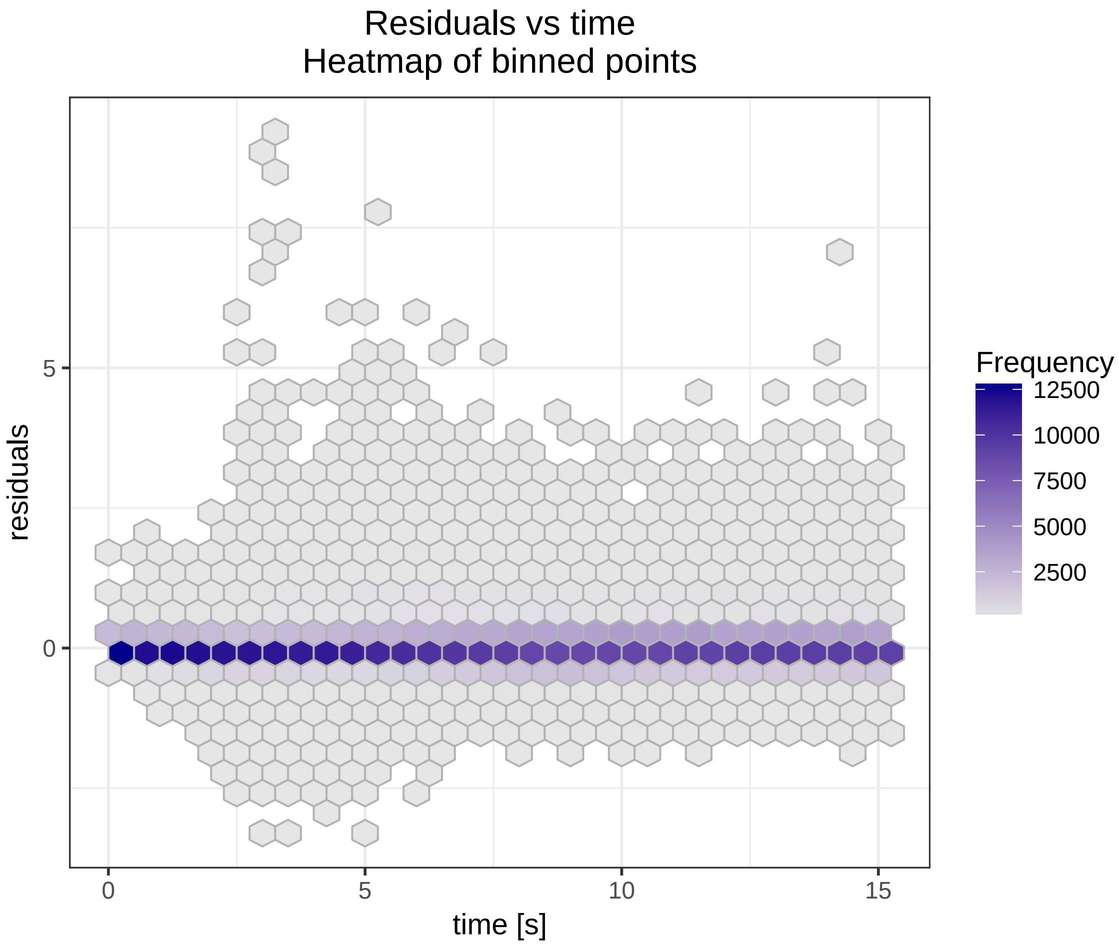
Residuals vs fitted values

```
plot_resVSfitted(model)
```



Residuals vs the function domain (here: time)

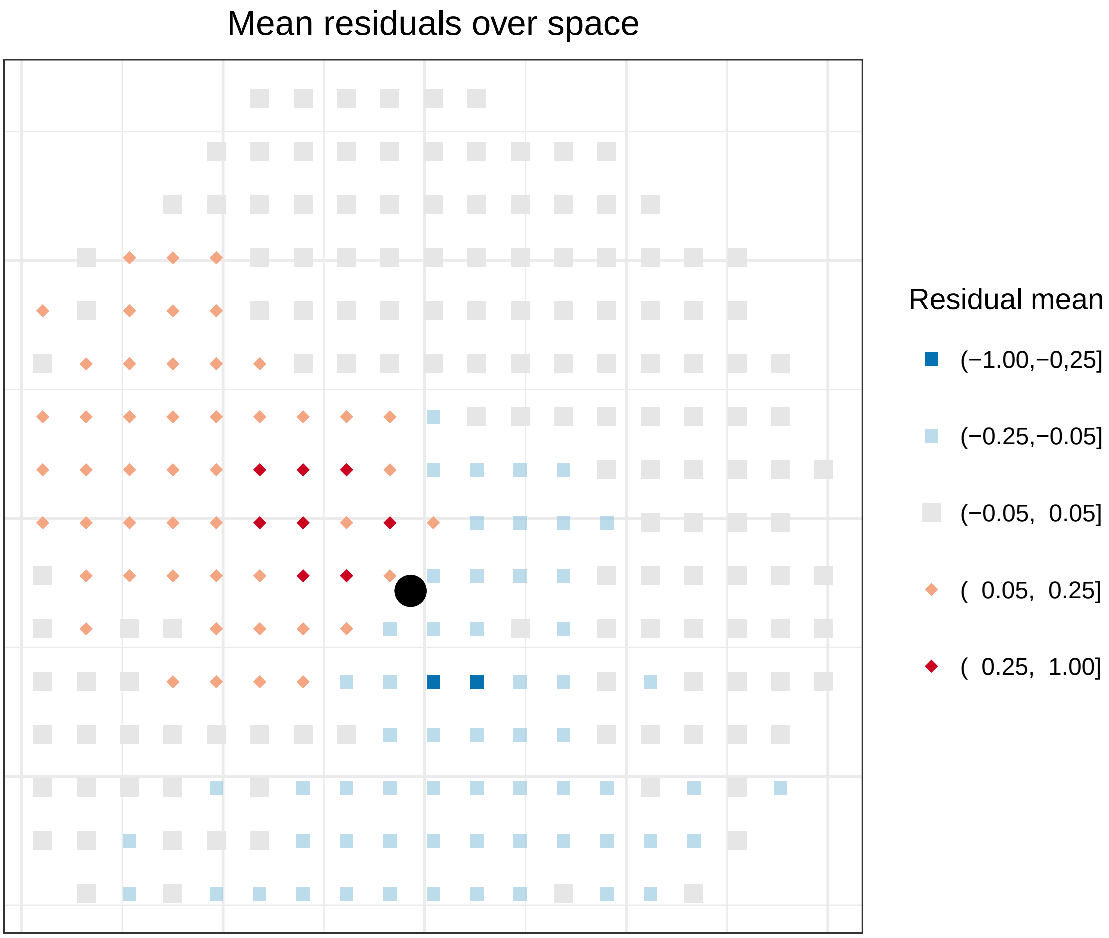
```
plot_resVSyindex(model, xlab = "time [s]",
                 main = "Residuals vs time\nHeatmap of binned points")
```



Residuals vs space

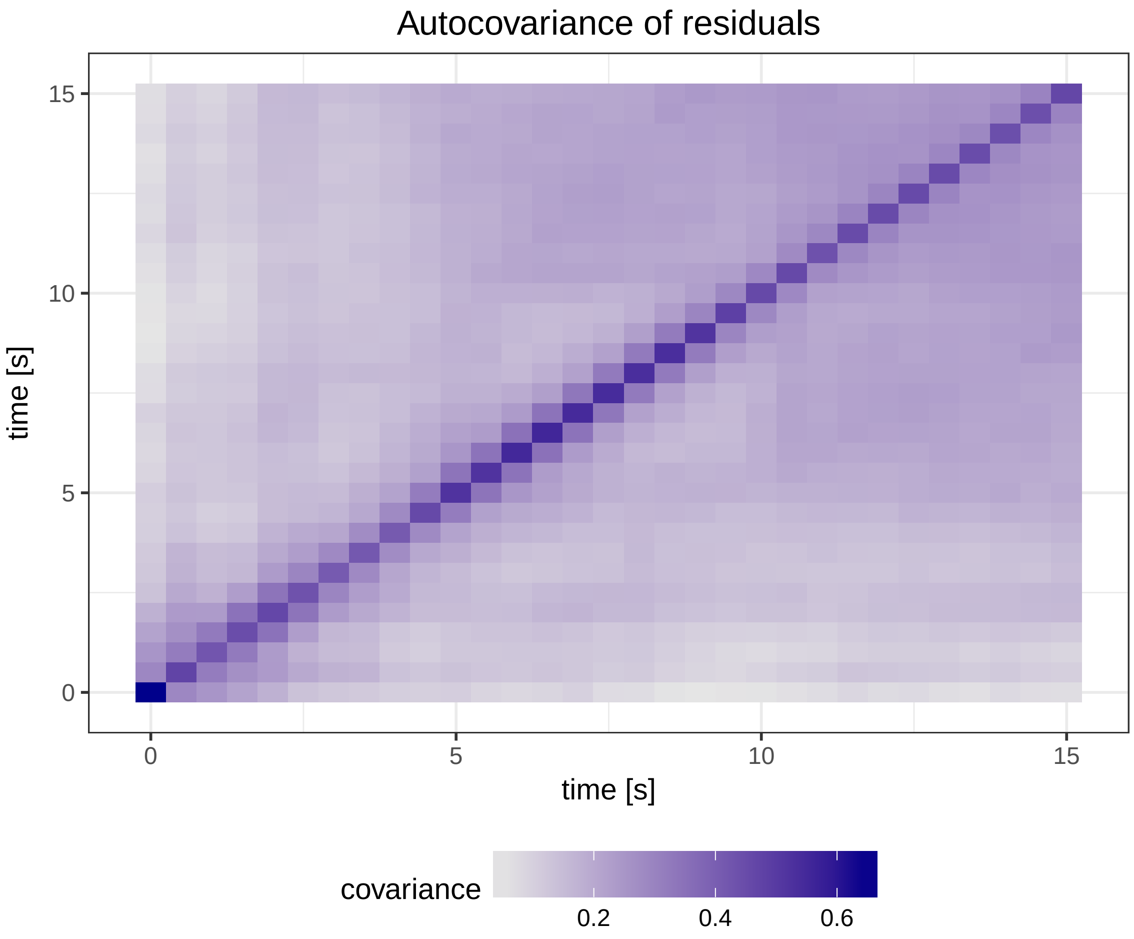
Note: the plot is not as highly resoluted as in the paper as we only work with a subset of the real data. For the figure in the paper, in order to show the visualization method properly, we predicted not just the subset-data with the model, but all seismographs from the original dataset of Bauer (2016).

```
dat_xy <- prepareData_residsVSxy(model, data, "ground_velocity",
                                 "seismometer_xCoord", "seismometer_yCoord")
breaks <- c(-1,-0.25,-0.05,0.05,0.25,1)
labels_cut <- c("(-1.00,-0,25]","(-0.25,-0.05]","(-0.05,  0.05]","(  0.05,  0.25]","(  0.25,  1.00]")
plot_residsVSxy(model, dat_xy, breaks, labels_cut,
                mark_location = which(dat_xy$seismometer == 62)) # Mark epicenter
```



Residual autocovariance

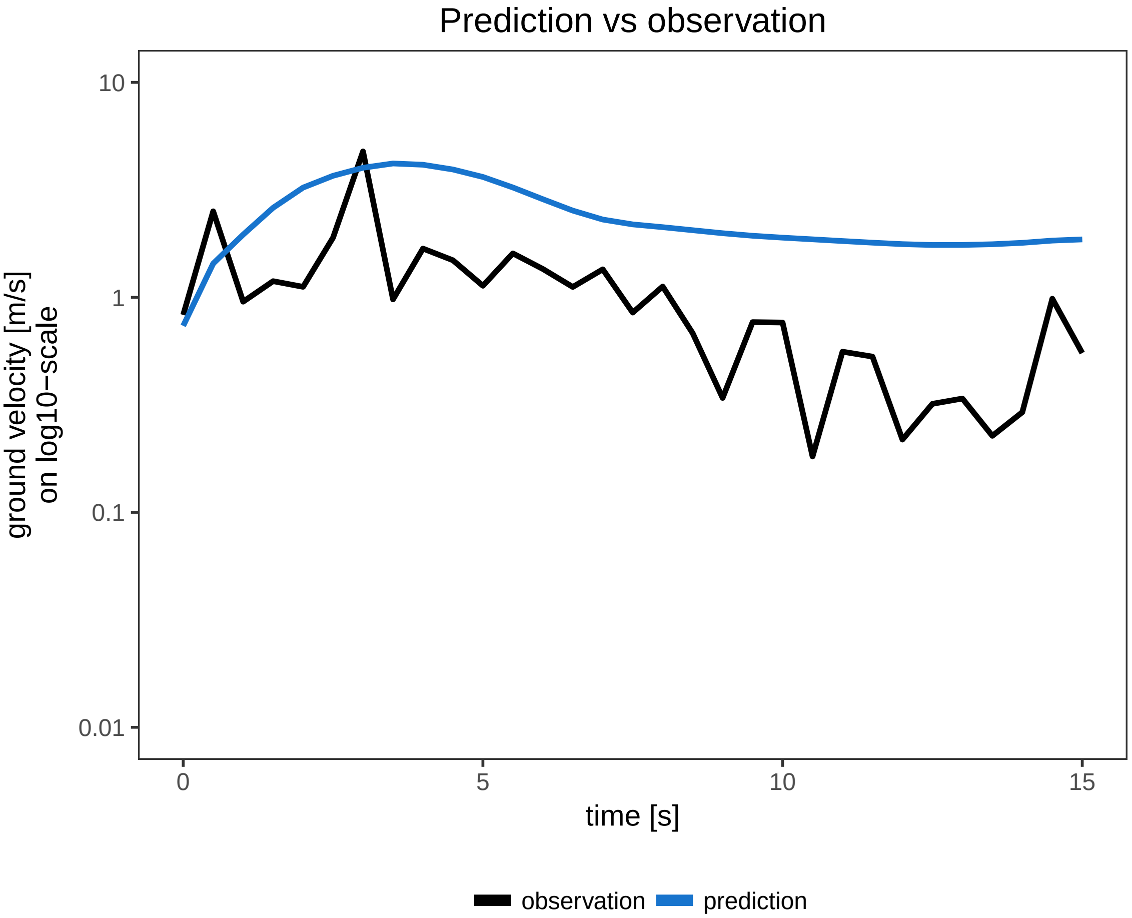
```r
plot_residAutocov(model)
```


Autocovariance of residuals
time [s]
15
10
5
0
0
5
10
15
time [s]
covariance
0.2
0.4
0.6

6.2 Prediction plots

Compare predictions with observations

```r
ybreaks <- 10^c(-2,-1,0,1)
plot_predVSobs(model, data[data$simulation == 88 & data$seismometer == 62,],
               yvar = "ground_velocity", yvar_label = "ground velocity [m/s]",
               log10 = TRUE, ylim = 10^c(-2,1), ybreaks = ybreaks, xlab = "time [s]")
```


Prediction vs observation
ground velocity [m/s]
on log10−scale
time [s]
observation
prediction