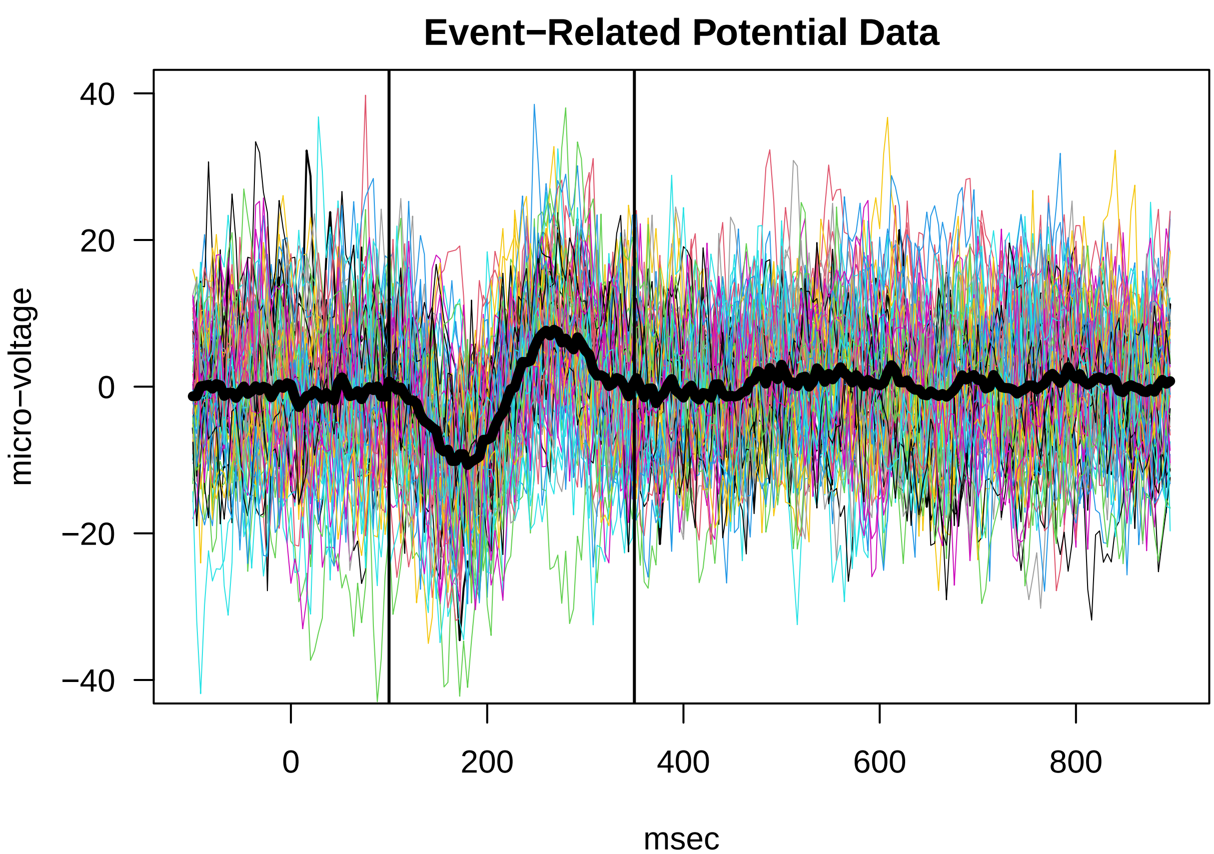
ERP Data Analysis

The raw real ERP data set is saved in `data-raw/Raw_ERP.csv`.

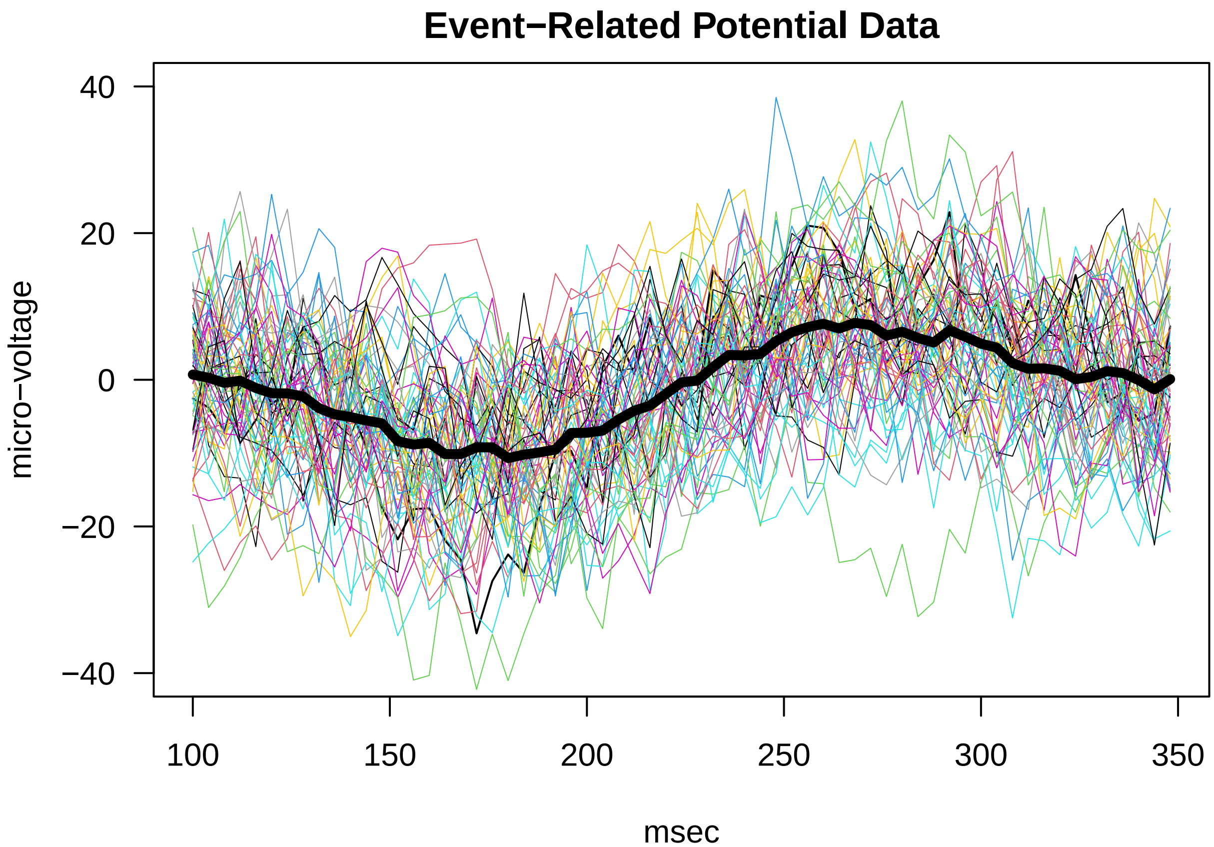
```
dataset <- read.csv("../data-raw/Raw_ERP.csv", header = TRUE)
names(dataset) <- c("id", "group", "region", "electrode", "condition",
                    "trial", seq(-100, 896, 4))
TT <- 250
erp_data <- dataset[, 7:(TT+6)]
```

The data are plotted as below. The focused time interval is $[100, 350]$ msec.

```
par(mar = c(4, 4, 2, 1))
plot(seq(-100, 896, 4), erp_data[1, ], type = "l", ylim = c(-40, 40),
     xlab = "msec", ylab =  "micro-voltage",
     main = "Event-Related Potential Data", las = 1)
for (i in 2:70) {
    lines(seq(-100, 896, 4), erp_data[i, ], col = i, lwd = 0.5)
}
lines(seq(-100, 896, 4), apply(erp_data, 2, mean), col = 1, lwd = 5)
abline(v = c(100, 350), lwd = 1.5)
```



```r
plot(seq(100, 350, 4), erp_data[1, as.character(seq(100, 350, 4))],
     type = "l", ylim = c(-40, 40),
     xlab = "msec", ylab =  "micro-voltage",
     main = "Event-Related Potential Data", las = 1)
for (i in 2:70) {
    lines(seq(100, 350, 4), erp_data[i, as.character(seq(100, 350, 4))], col = i, lwd = 0.5)
}
lines(seq(100, 350, 4), apply(erp_data[, as.character(seq(100, 350, 4))], 2, mean), col = 1, lwd = 5)
```




```r
## time bound index
lwr_idx <- which(names(erp_data) == 100)
upr_idx <- which(names(erp_data) == 348)
bd <- lwr_idx:upr_idx
msec_idx <- seq(-100, 896, 4)
n_trial <- 1:72

## sample size
n_erp <- length(seq(-100, 896, 4)[bd])
```

The following code shows how the posterior density of latency is computed.

```r
library(dgp)
library(emulator)
```

```
## Loading required package: mvtnorm
```

```r
################################################
### average two trials
```

```r
################################################

## ERP waveform averaged over 2 trials
k <- 1
avg_erp <- apply(as.matrix(erp_data)[(2*k-1):(2*k), ], 2, mean)

## time points
erp_x <- seq(0.004, 1, length = 250)

H0_erp <- outer(erp_x[bd], erp_x[bd],
                FUN = function(x1, x2) (x1 - x2))
x_a_erp <- min(erp_x[bd])
x_b_erp <- max(erp_x[bd])
grid_t_erp <- seq(x_a_erp, x_b_erp, length.out = 400)

erp_log_post_prob_t <- rep(0, length(grid_t_erp))
erp_post_prob_t <- rep(0, length(grid_t_erp))

x_test <- seq(x_a_erp, x_b_erp, length.out = 100)
x_test_msec <- seq(min(seq(-100, 896, 4)[bd]), max(seq(-100, 896, 4)[bd]),
                   length.out = 100)

len_trial <- length(n_trial)

## parameters of beta prior
sh1 <- 1
sh2 <- 1

## Optimizing hyperparameters
gp_res <- Rsolnp::solnp(pars = c(.5, .5, .5), fun = log_mar_lik_gp,
                        LB = c(0.0001, 0.0001, 0.0001),
                        UB = c(1 / 0.0001, 1 / 0.0001, 1 / 0.0001),
                        control = list(TOL = 1e-5, trace = 0),
                        y = avg_erp[bd], H0 = H0_erp)

erp_n <- length(erp_x[bd])
sig_gp <- gp_res$par[1]
tau_gp <- gp_res$par[2]
h_gp <- gp_res$par[3]
lambda_gp <- sig_gp ^ 2 / (erp_n * tau_gp ^ 2)
Kff_gp <- se_ker(H0 = H0_erp, tau = 1, h = h_gp)
A_gp <- Kff_gp + diag((erp_n * lambda_gp), erp_n)


## Posterior distribution of local extrema
for (i in 1:length(grid_t_erp)) {
    erp_log_post_prob_t[i] <- log_post_t_theory(t = grid_t_erp[i],
                                                y = avg_erp[bd],
                                                x = erp_x[bd],
                                                Kff = Kff_gp,
                                                A = A_gp,
                                                lambda = lambda_gp,
                                                h = h_gp,
```

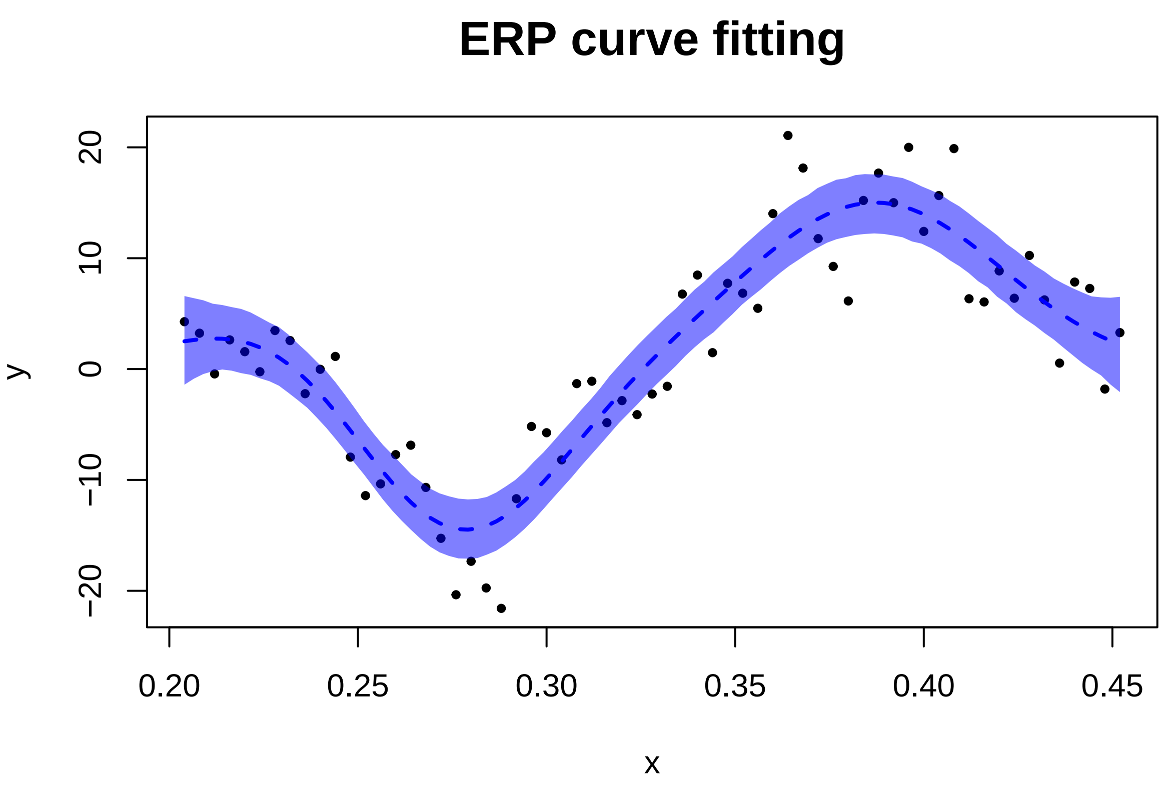
```
                                                        sig2 = sig_gp ^ 2,
                                                        shape1 = sh1, shape2 = sh2,
                                                        a = x_a_erp, b = x_b_erp)
}
erp_post_prob_t <- exp(erp_log_post_prob_t - max(erp_log_post_prob_t))
```

To obtain the fitted curve and uncertainty bands, we can use the function `get_pred_ci_gp()`.

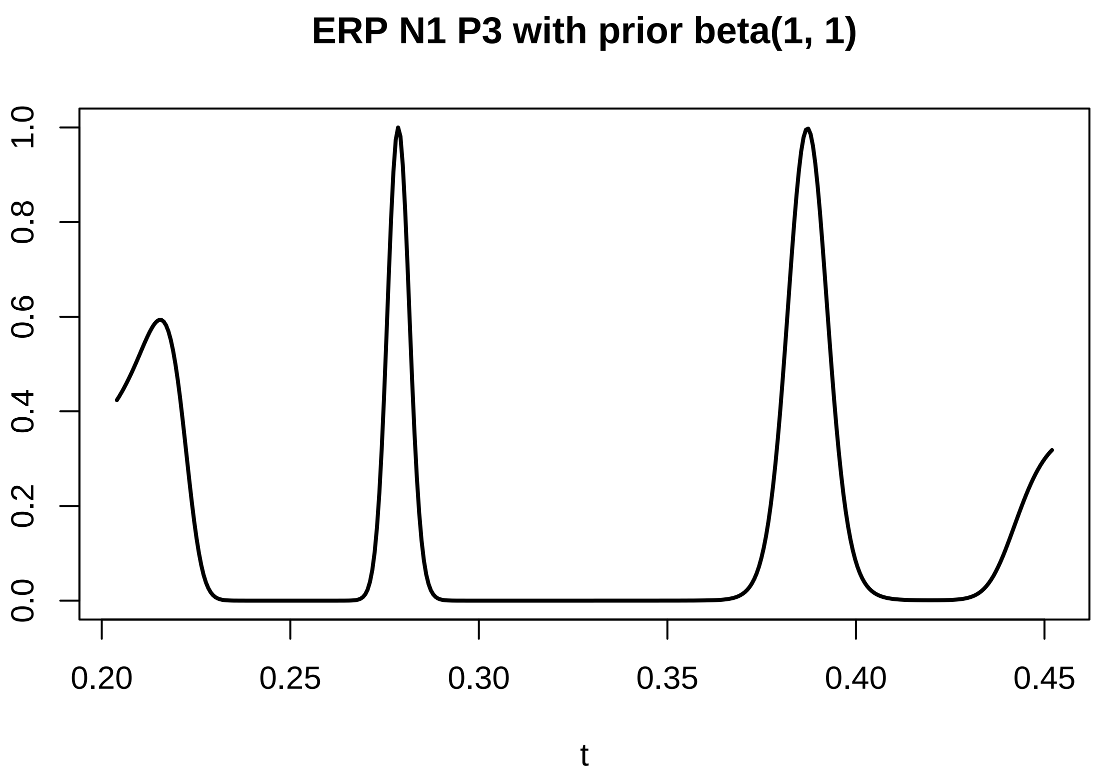
```
pred_erp <- get_pred_ci_gp(eb_par = gp_res$par, x = erp_x[bd], x_test = x_test,
                           y = avg_erp[bd])
```

The function `plot_pred_gp_f_y()` plots the fitted waveform as well as the uncertainty intervals.

```
## Curve fitting plotting
plot_pred_gp_f_y(x = erp_x[bd], y = avg_erp[bd],
                 x_test = x_test,
                 mu_test = pred_erp$mu_test,
                 CI_Low_f = pred_erp$ci_low,
                 CI_High_f = pred_erp$ci_high,
                 ylim = range(avg_erp[bd]),
                 xlim = c(x_a_erp, x_b_erp), is.der.line = FALSE,
                 is.true.fcn = FALSE, cex = 0.5,
                 plot.type = "p", col.poly = rgb(0, 0, 1, 0.5),
                 pred_lwd = 2, title = "ERP curve fitting", is.legend = FALSE)
```




```
## Posterior density plotting
plot(grid_t_erp, erp_post_prob_t, ylab = "", xlab = "t", type = 'l',
     main = paste0("ERP N1 P3 with prior beta(", sh1, ", ", sh2,")"),
     ylim = c(0, 1), lwd = 2)
```




```r
## HPD interval
(erp_hpd <- get_hpd_interval_from_den(erp_post_prob_t, grid_t = grid_t_erp,
                                       target_prob = 0.95))
```

```
## $no_cluster
## [1] 4
##
## $ci_lower
## [1] 0.2040000 0.2723709 0.3755489 0.4395689
##
## $ci_upper
## [1] 0.2251328 0.2848020 0.3991679 0.4520000
##
## $prob_value
## [1] 0.9596019
##
## $den_value
## [1] 0.1012871
```

```r
## estimated number of stationary points
(erp_map <- get_map(post_den = erp_post_prob_t, grid_t = grid_t_erp, hpdi = erp_hpd))
```

```
## [1] 0.2158095 0.2785865 0.3873584 0.4520000
```