Figure 1: Visualization of an example concurrent functional linear mixed model.

Al W Xin

Overview

We generated synthetic data to demonstrate how `fastFMM` might produce coefficient and confidence interval estimates.

Directory setup and function loading

We require the following packages to organize and plot data.

```
suppressMessages(library(dplyr))
suppressMessages(library(stringr))
suppressMessages(library(ggplot2))
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```
theme_set(theme_bw())
```

We saved the helper functions in the directory `fun/`.

```
# Data-generating functions
source("fun/fig1a.R")
# Function to create random draws from data-generating functions
source("fun/synthesize_data.R")
```

File outputs will be saved in `img/1-cflmm`.

```
img_dir <- paste0("img/1-cflmm")
if (!dir.exists(img_dir)) {
  dir.create(img_dir)
}
```

Synthetic data

Model-building will be demonstrated on synthetic data without a random slope.

The choice of number of clusters and number of replicates is arbitrary. For the figure, the point `s_pt` is also motivated by aesthetic choice and not experimental significance.

```r
# S comes from the synthetic data function
L <- length(S)
s_pt <- S[25]

set.seed(20250605)
dat_list <- synthesize_data(
  I = 40,
  J = 50,
  snr_B = 0.5,
  snr_sigma = 0.5,
  ptwise_snr = T,
  y_name = "y",
  x_name = "x",
  return_fixed = T,
  return_noise = T,
  asis_cols = F,
  fix_J = T
)

dat <- dat_list$data
```

Graphing a single example replicate

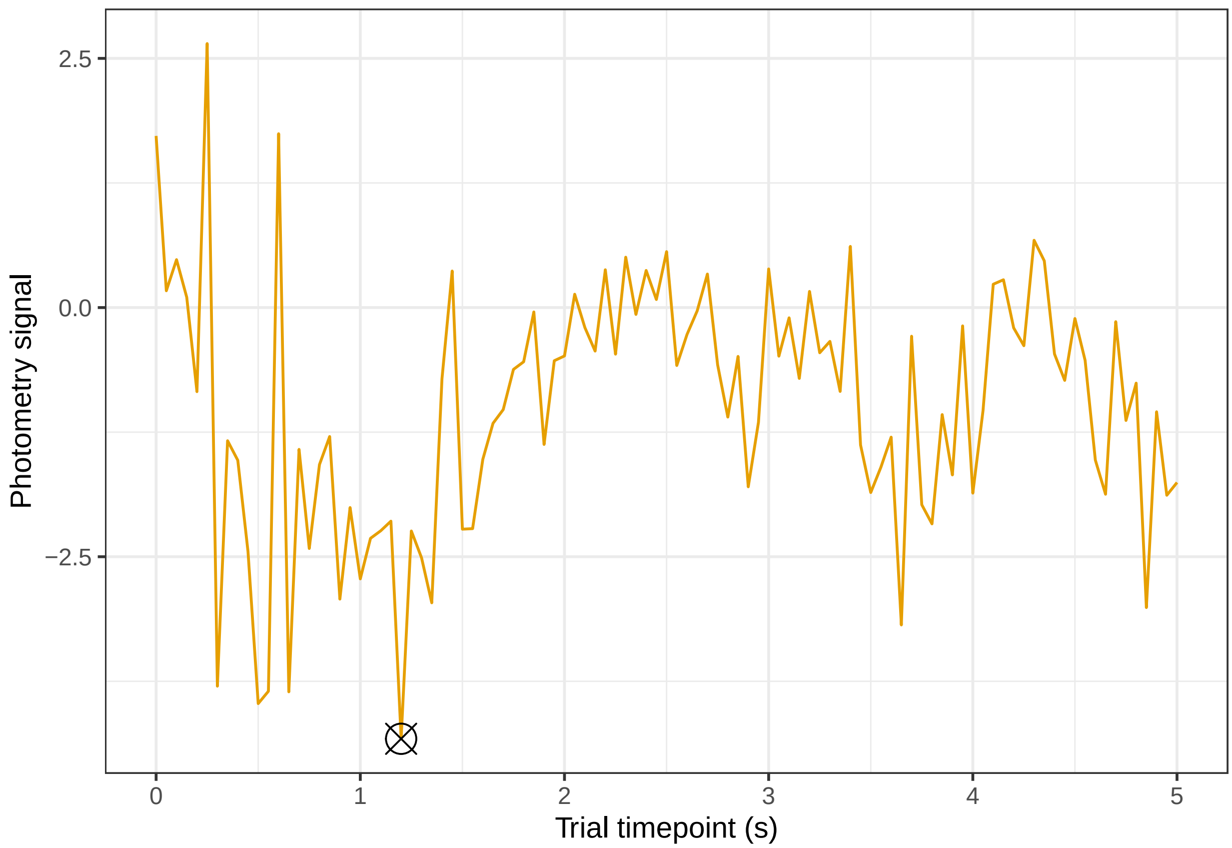
The data returned by `synthesize_data` is in a wide format and needs to be adjusted before being fed to `ggplot2`. To keep the graphs clear, we pick an arbitrary cluster (here, `ex_idx <- 16`).

```r
# Example chosen arbitrarily
ex_idx <- 16
# Filter for the example cluster
dat_ex_wide <- dat[ex_idx, ]
# Pivot to a long data frame
dat_ex <- data.frame(
  photo = as.numeric(
    dat_ex_wide[1, grep("^y", colnames(dat_ex_wide))]
  ),
  fun_cov = as.numeric(
    dat_ex_wide[1, grep("^x", colnames(dat_ex_wide))]
  )
) %>%
  tidyr::pivot_longer(
    photo:fun_cov,
    names_to = "data_type"
  ) %>%
  mutate(
    s = rep(S, each = 2)
  )
```

Filtering for `data_type` produces two separate graphs for the outcome and covariate.

```r
dat_ex %>%
  filter(data_type == "photo") %>%
  ggplot(aes(x = s, y = value)) +
```

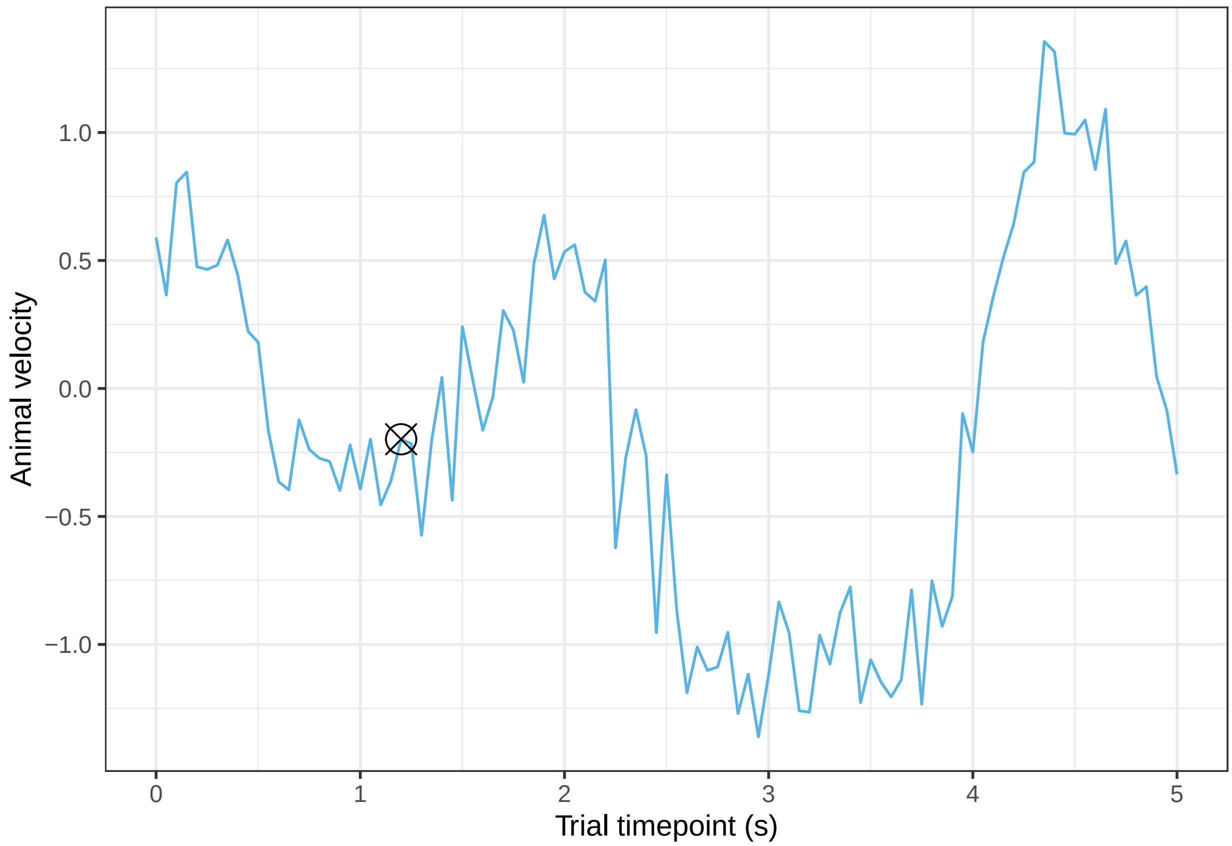
```
    geom_line(color = palette.colors()[2]) +
    labs(
      x = "Trial timepoint (s)",
      y = "Photometry signal"
    ) +
    geom_point(
      data = filter(dat_ex, data_type == "photo", s == s_pt),
      shape = 13,
      size = 5
    )
```




```
ggsave(
  paste0(img_dir, "exp_photo.png"),
  width = 3, height = 2.5,
  dpi = 300
)
```

```
dat_ex %>%
  filter(data_type == "fun_cov") %>%
  ggplot(aes(x = s, y = value)) +
    geom_line(color = palette.colors()[3]) +
    labs(
      x = "Trial timepoint (s)",
      y = "Animal velocity"
    ) +
  geom_point(
```

```r
    data = filter(dat_ex, data_type == "fun_cov", s == s_pt),
    shape = 13,
    size = 5
  )
```




```r
ggsave(
  paste0(img_dir, "exp_speed.png"),
  width = 3, height = 2.5,
  dpi = 300
)
```

Modeled fixed effects

```r
mod <- fastFMM::fui(
  y ~ x + (1 + x | id),
  data = dat,
  parallel = F,
  concurrent = T,
  randeffs = T
)
```

```
## Functional covariate(s): x

## Step 1: Fit Massively Univariate Mixed Models
```

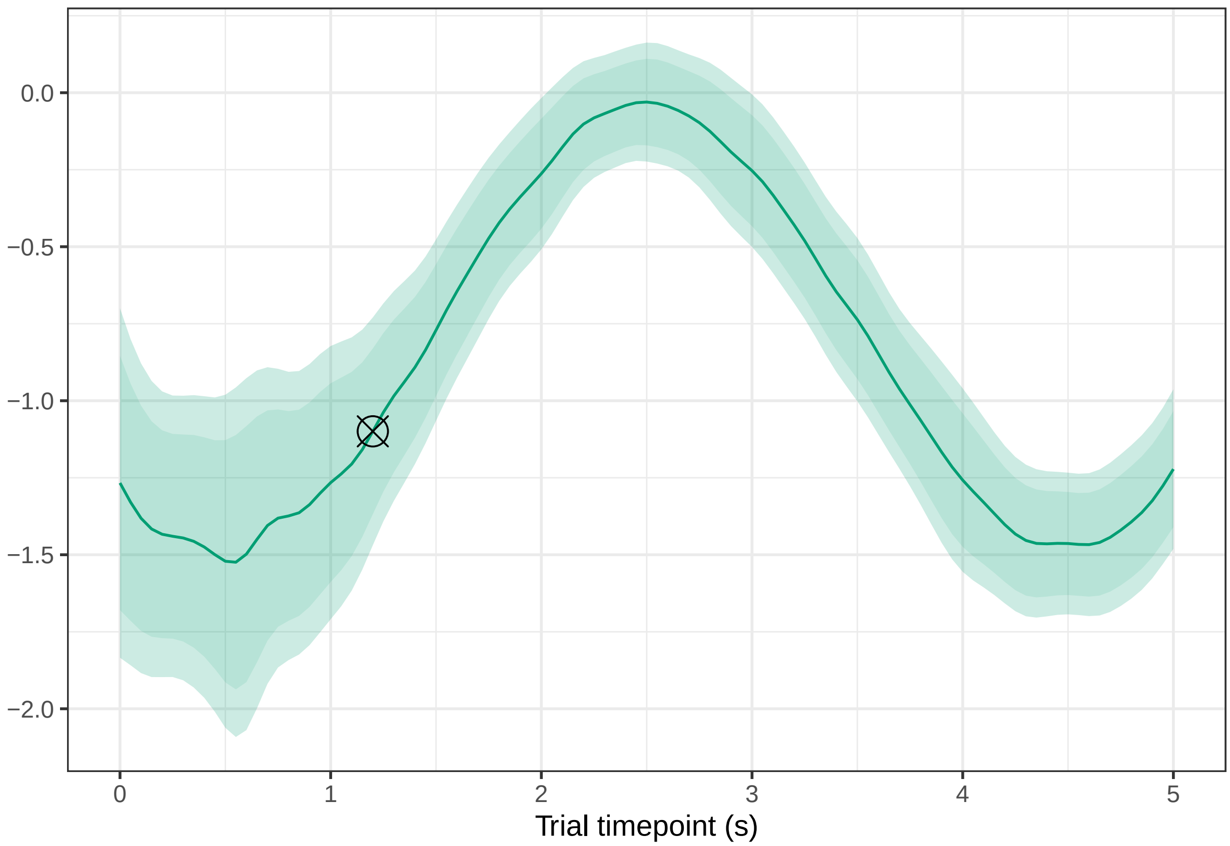
```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues

## Step 2: Smoothing

## Step 3: Inference (Analytic)

## Step 3.0: Preparation

## Step 3.1.1: Preparation B

## Step 3.1.1: Method of Moments Covariance Estimator

## Step 3.1.2: Smooth G

## Step 3.2: First step

## Step 3.2.1: First step

## Step 3.3: Second step

## Complete!
##  - Use plot_fui() function to plot estimates.
##  - For more information, run the command:  ?plot_fui
```

```r
# fastFMMconc::plot_fui(mod)
```

```r
source("fun/plottable_fmm.R")
fixfx <- plottable_fmm(mod, cov_names = c("Intercept", "Slope")) %>%
  mutate(s = (s - 1) / 20) %>%
  as.data.frame()
```

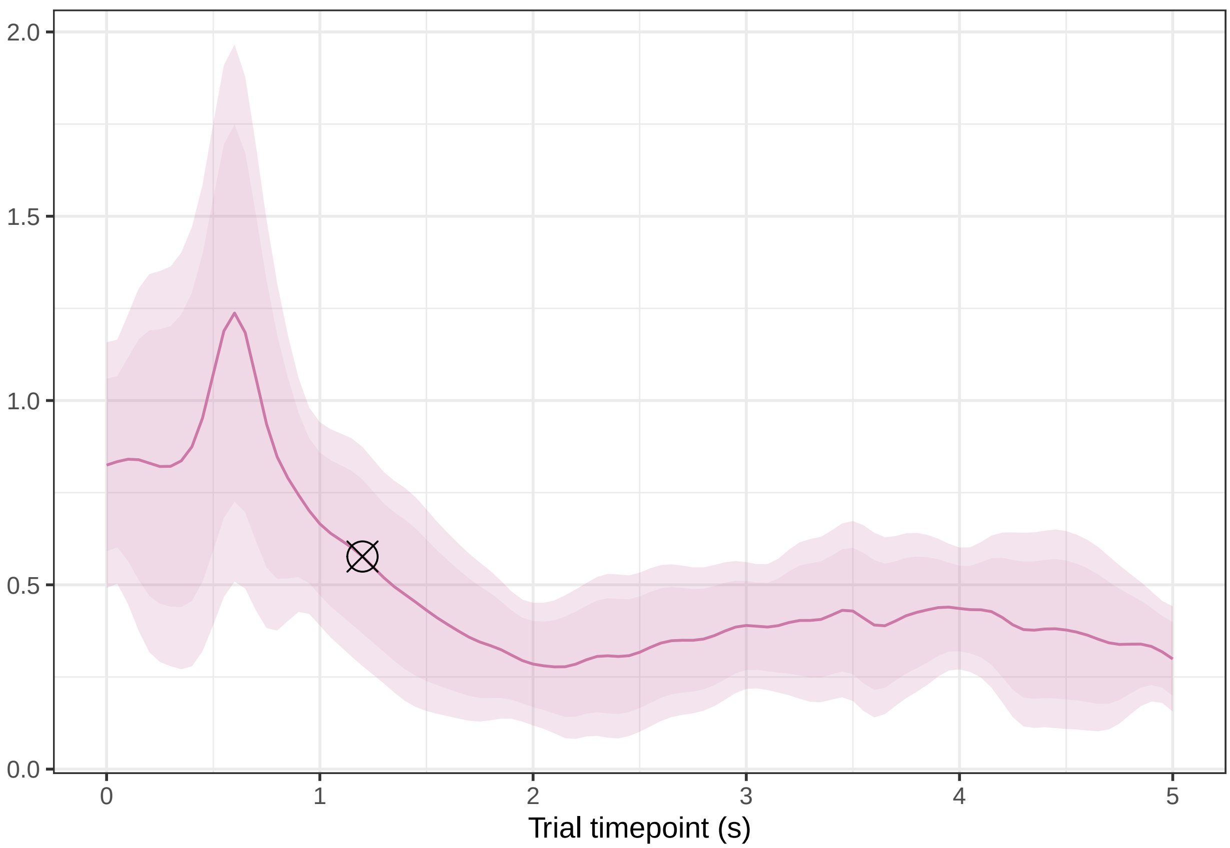
```r
s_pt <- 1.2

fixfx %>%
  filter(cov == "Intercept") %>%
  ggplot(aes(x = s, y = val)) +
    geom_line(color = palette.colors()[4]) +
    geom_ribbon(
      aes(ymin = val - ci, ymax = val + ci),
      fill = palette.colors()[4],
      alpha = 0.1
    ) +
    geom_ribbon(
      aes(ymin = val - joint_ci, ymax = val + joint_ci),
      fill = palette.colors()[4],
      alpha = 0.2
```

```
    ) +
    geom_point(
      data = filter(fixfx, cov == "Intercept", s == s_pt),
      shape = 13,
      size = 5
    ) +
    labs(
      x = "Trial timepoint (s)",
      y = NULL
    )
```




```
ggsave(
  paste0(img_dir, "beta_0.png"),
  width = 3, height = 2.5,
  dpi = 300
)

fixfx %>%
  filter(cov == "Slope") %>%
  ggplot(aes(x = s, y = val)) +
    geom_line(color = palette.colors()[8]) +
    geom_ribbon(
      aes(ymin = val - ci, ymax = val + ci),
      fill = palette.colors()[8],
      alpha = 0.1
    ) +
```

```
    geom_ribbon(
      aes(ymin = val - joint_ci, ymax = val + joint_ci),
      fill = palette.colors()[8],
      alpha = 0.2
    ) +
    geom_point(
      data = filter(fixfx, cov == "Slope", s == s_pt),
      shape = 13,
      size = 5
    ) +
    labs(
      x = "Trial timepoint (s)",
      y = NULL
    )
```



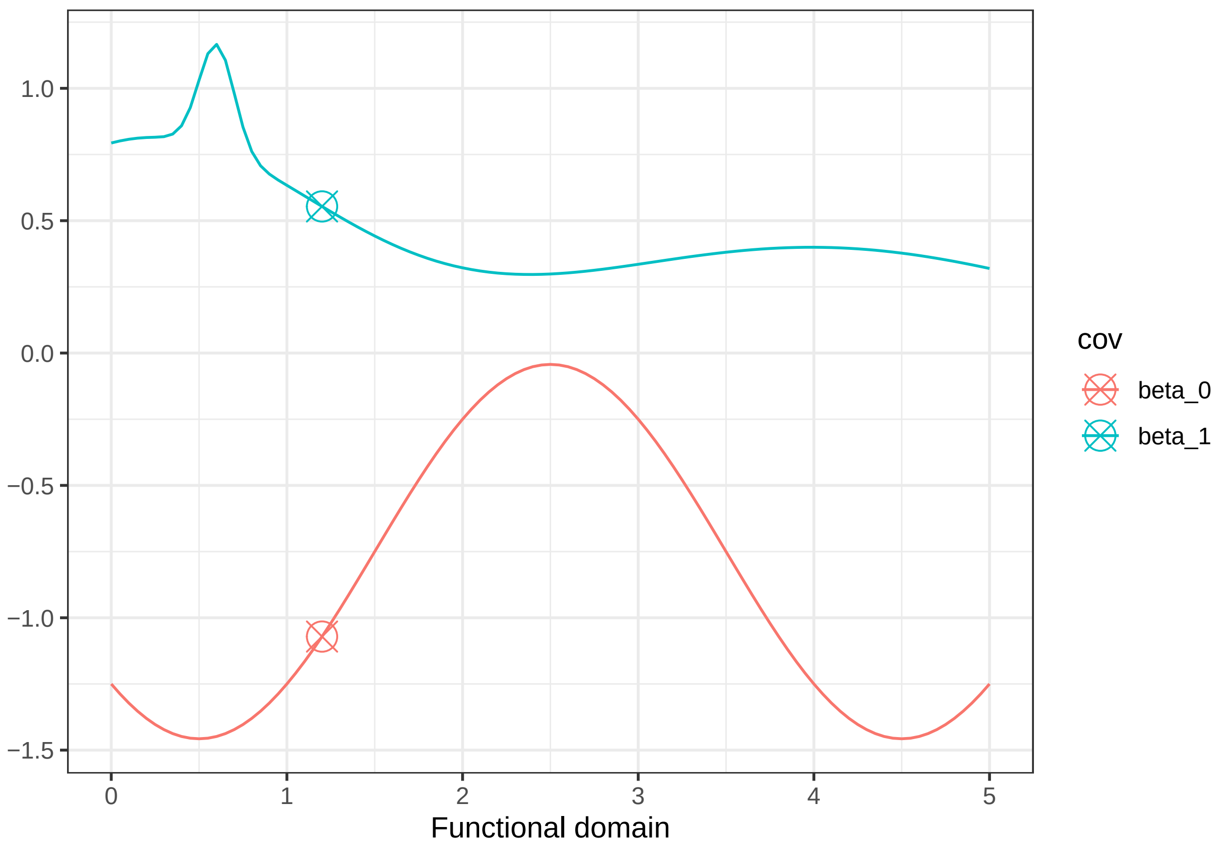

```
ggsave(
  paste0(img_dir, "beta_1.png"),
  width = 3, height = 2.5,
  dpi = 300
)
```

True fixed effects

```r
source("fun/fig1a.R")
source("fun/truth_from_syn.R")

set.seed(99)
L <- length(beta_0())
beta_df <- truth_from_syn("fig1a") %>%
  mutate(s = (s - 1) / 20)
```

The true fixed effects are shown below.

```r
ggplot(beta_df, aes(x = s, y = truth_val, color = cov)) +
  geom_line() +
  labs(
    x = "Functional domain",
    y = NULL
  ) +
  geom_point(
    data = filter(beta_df, s == s_pt),
    shape = 13,
    size = 5
  )
```

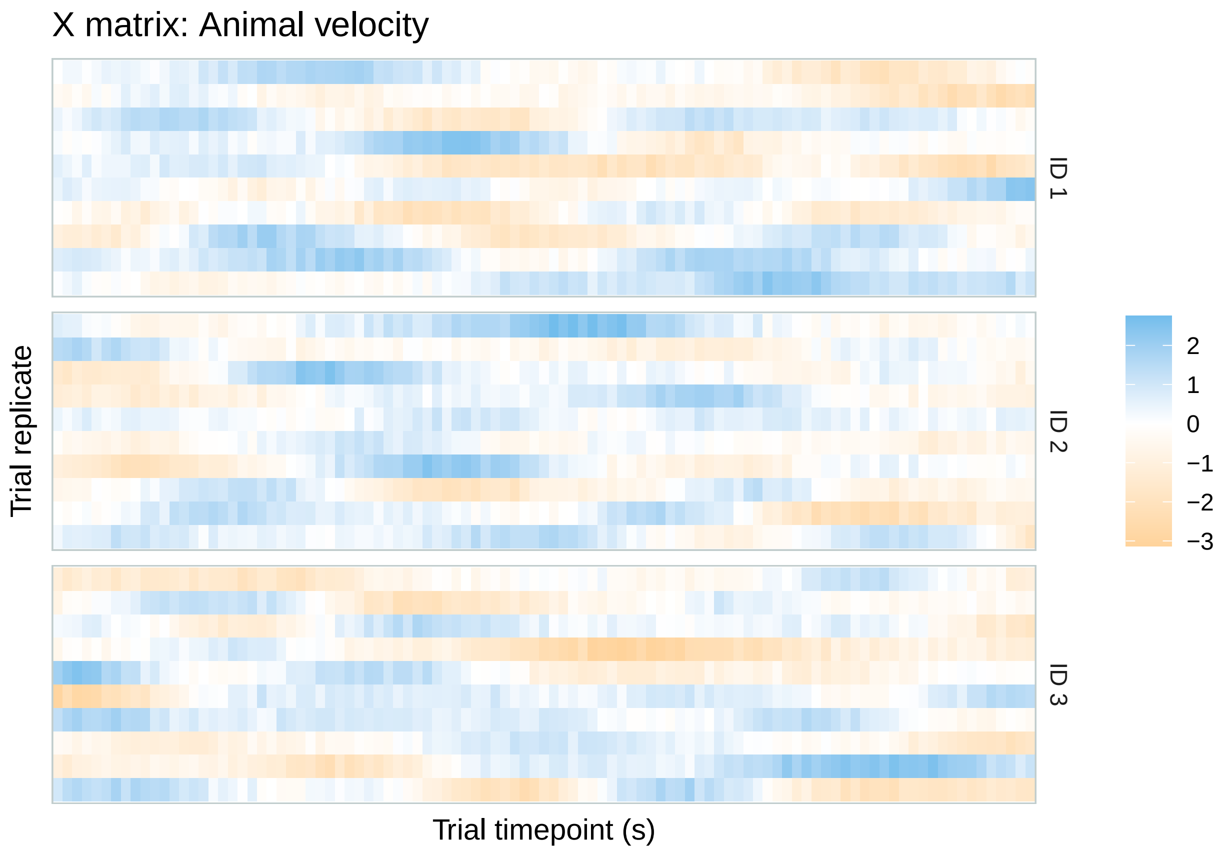



```r
ggsave(
  paste0(img_dir, "fixed_fx.png"),
  width = 7, height = 3, dpi = 300
)
```

Heat map

Functional covariate

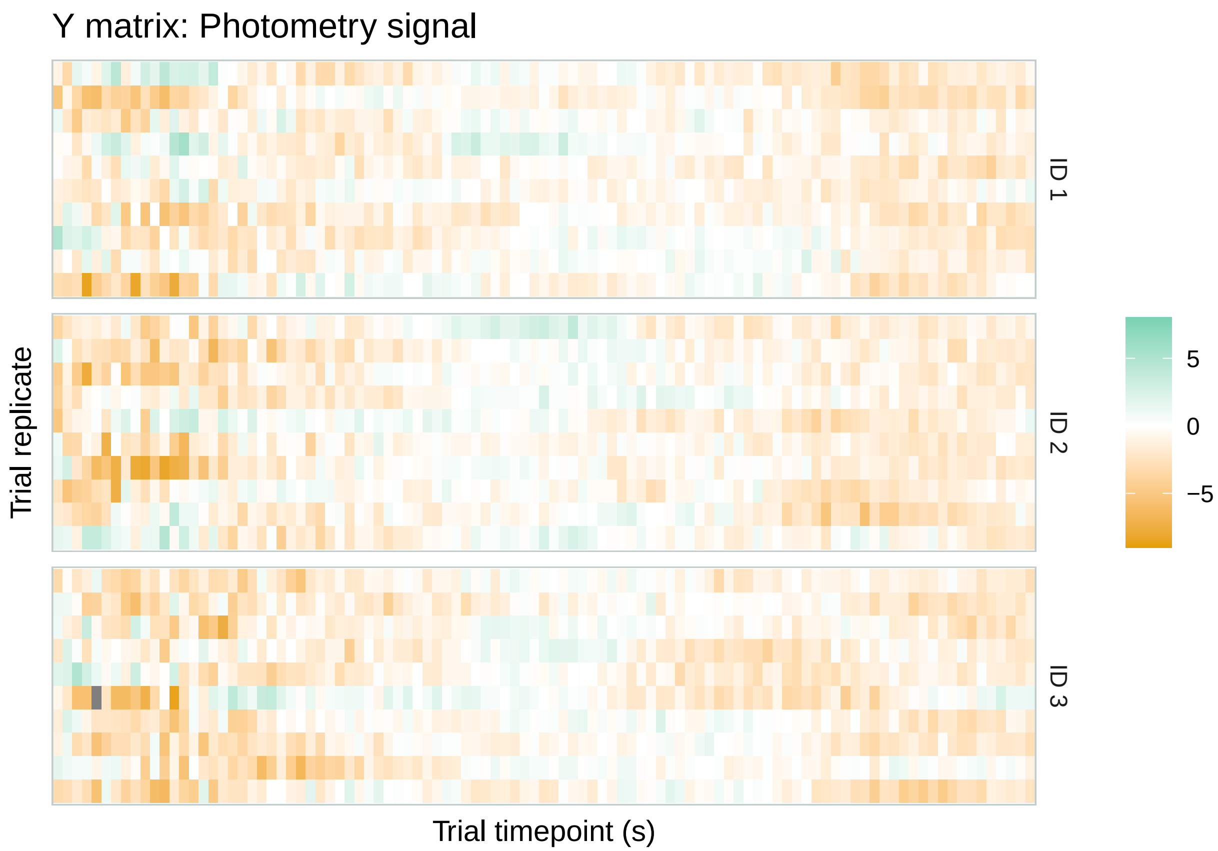
```
dat_x <- dat %>%
  select(x_1:x_101, id, trial) %>%
  filter(id %in% 1:3) %>%
  tidyr::pivot_longer(-id:-trial, names_to = "timepoint") %>%
  mutate(
    timepoint = factor(timepoint, levels = paste0("x_", 1:L)),
    trial = factor(trial, levels = max(trial):1),
    id = paste0("ID ", id)
  )

dat_x %>%
  filter(trial %in% 1:10) %>%
  ggplot(aes(x = timepoint, y = trial)) +
  geom_tile(aes(fill = value)) +
  facet_grid(rows = vars(id)) +
  scale_fill_gradient2(
    high = palette.colors()[3],
    mid = "white",
    low = "burlywood1"
  ) +
  # theme_minimal() +
  labs(
    x = "Trial timepoint (s)",
    # y = "Trial replicate",
    y = "Trial replicate",
    title = "X matrix: Animal velocity",
    fill = NULL
    # fill = "Speed"
  ) +
  theme_minimal() +
  theme(
    panel.border = element_rect(color = "azure3", fill = NA),
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    panel.grid = element_blank()
  )
```




```
ggsave(
  paste0(img_dir, "heat_speed.png"),
  width = 6, height = 2.4, dpi = 300
)
```

```
dat_y <- dat %>%
  select(y_1:y_101, id, trial) %>%
  filter(id %in% 1:3) %>%
  tidyr::pivot_longer(-id:-trial, names_to = "timepoint") %>%
  mutate(
    timepoint = factor(timepoint, levels = paste0("y_", 1:L)),
    trial = factor(trial, levels = max(trial):1),
    id = paste0("ID ", id)
  )


dat_y %>%
  filter(trial %in% 1:10) %>%
  ggplot(aes(x = timepoint, y = trial)) +
    theme_minimal() +
    theme(
      panel.border = element_rect(color = "azure3", fill = NA),
      axis.text = element_blank(),
      axis.ticks = element_blank(),
      panel.grid = element_blank()
    ) +
```

```r
    geom_tile(aes(fill = value)) +
    facet_grid(rows = vars(id)) +
    scale_fill_gradient2(
      low = palette.colors()[2],
      mid = "white",
      high = "aquamarine3",
      limits = c(-9, 8)
    ) +
    labs(
      x = "Trial timepoint (s)",
      # y = "Trial replicate",
      y = "Trial replicate",
      fill = NULL,
      title = "Y matrix: Photometry signal"
    )
```




```r
ggsave(
  paste0(img_dir, "heat_photo.png"),
  width = 6, height = 2.4, dpi = 300
)
```

Random effects

Extract and add random effects

```
randfx <- mod$randeffs
rand_0 <- as.data.frame(randfx[1:10, 1, ]) %>%
  mutate(id = as.factor(1:10)) %>%
  tidyr::pivot_longer(V1:V101, names_to = "s", values_to = "rfx") %>%
  mutate(s = as.numeric(stringr::str_remove(s, "V"))) %>%
  mutate(cov = "Intercept")
rand_1 <- as.data.frame(randfx[1:10, 2, ]) %>%
  mutate(id = as.factor(1:10)) %>%
  tidyr::pivot_longer(V1:V101, names_to = "s", values_to = "rfx") %>%
  mutate(s = as.numeric(stringr::str_remove(s, "V"))) %>%
  mutate(cov = "Slope")

randfx <- rbind(rand_0, rand_1) %>%
  mutate(s = (s - 1)/ 20)
```

Bind to the true values:

```
fix_rand <- left_join(randfx, fixfx, by = c("s", "cov")) %>%
  mutate(rand_total = rfx + val)
```

Smooth random effects

Although it's possible to extract the smoothed random effects from the model, it's easier to redo the smoothing with the correct number of splines.

```
set.seed(1)
rand_list <- lapply(
  1:10,
  function(i) {
    data.frame(
      x = 1:L,
      y = fix_rand %>%
        filter(
          cov == "Slope",
          id == i
        ) %>%
        .$rand_total
    )
  }
)

names(rand_list) <- 1:10

smooth_rand_list <- lapply(
  1:10,
  function(i) {
    dat <- rand_list[[i]]
```

```r
    temp <- stats::smooth.spline(
      x = dat$x,
      y = dat$y,
      nknots = round(L / 8)
    )
    data.frame(
      s = 1:L,
      val = temp$y,
      id = i
    )
  }
)

smooth_rand_1 <- do.call(rbind, smooth_rand_list) %>%
  mutate(
    s  = (s - 1) / 20,
    id = as.factor(id)
  )
```

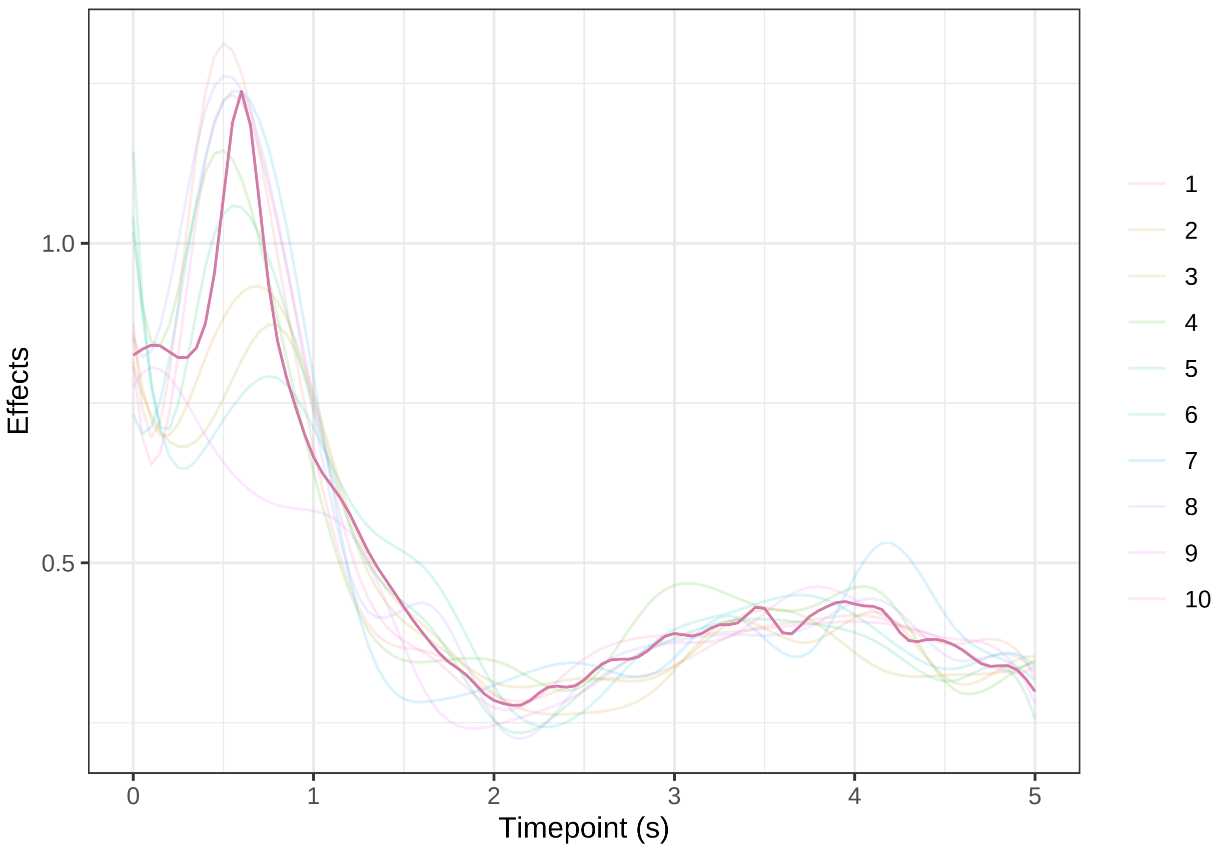
```r
set.seed(1)
rand_list <- lapply(
  1:10,
  function(i) {
    data.frame(
      x = 1:L,
      y = fix_rand %>%
        filter(
          cov == "Intercept",
          id == i
        ) %>%
        .$rand_total
    )
  }
)

names(rand_list) <- 1:10

smooth_rand_list <- lapply(
  1:10,
  function(i) {
    dat <- rand_list[[i]]
    temp <- stats::smooth.spline(
      x = dat$x,
      y = dat$y,
      nknots = round(L / 8)
    )
    data.frame(
      s = 1:L,
      val = temp$y,
      id = i
    )
  }
)
```

```r
smooth_rand_0 <- do.call(rbind, smooth_rand_list) %>%
  mutate(
    s  = (s - 1) / 20,
    id = as.factor(id)
  )
```

Graphing all random effects

The random effects are centered around the fixed effects.

```r
ggplot(smooth_rand_1, aes(x = s, y = val)) +
  geom_line(aes(color = id), alpha = 0.15) +
  geom_line(
    data = fixfx %>% filter(cov == "Slope"),
    aes(x = s, y = val),
    color = "#d37aa7"
  ) +
  labs(
    color = NULL,
    x = "Timepoint (s)",
    y = "Effects"
  )
```

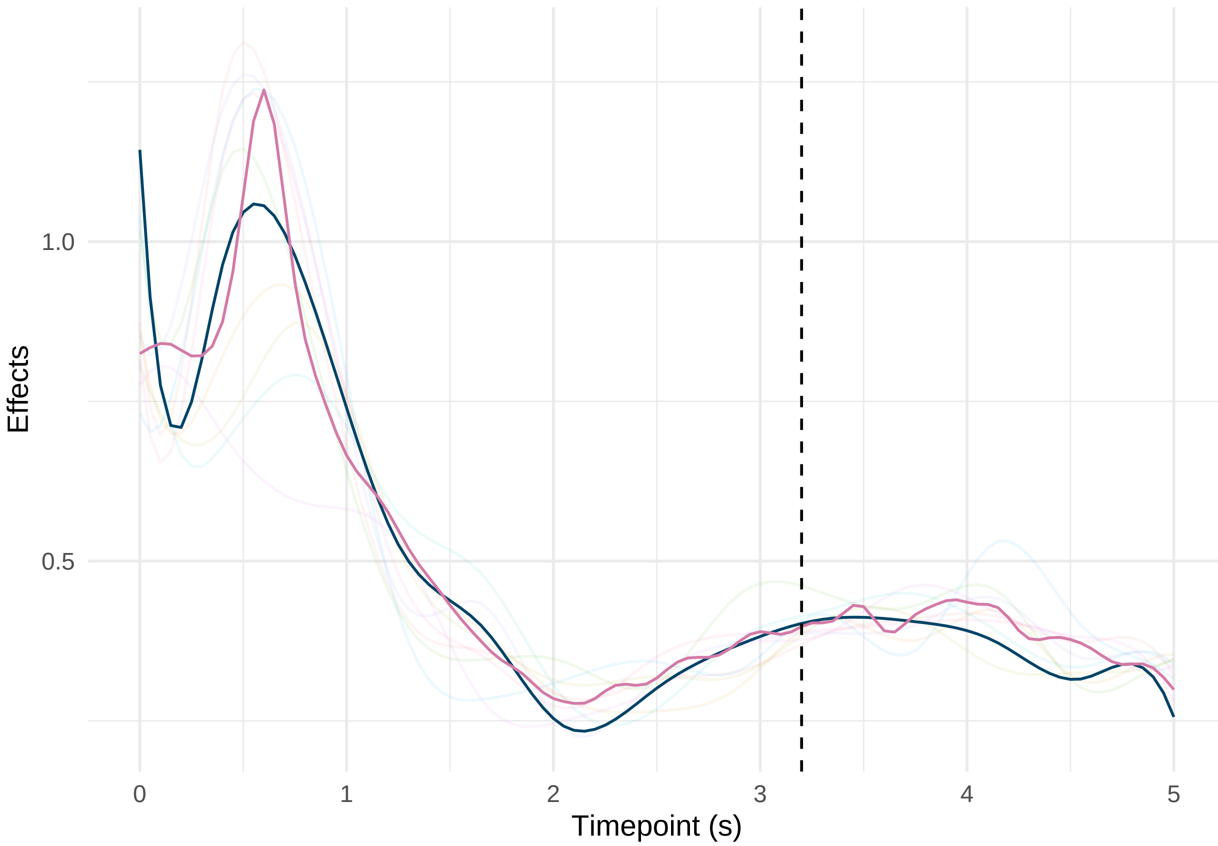


```r
ggsave(paste0(img_dir, "fixed_rand_fx.png"), width = 6, height = 4)
```

Highlighting one particular random effect

ID 5 was chosen as a contrast because of clear separation at around $s = 3.5$ seconds.

```r
smooth_rand_1 %>%
  ggplot(aes(x = s, y = val)) +
    geom_line(aes(color = id), alpha = 0.08) +
    geom_line(data = filter(smooth_rand_1, id == 5), aes(x = s, y = val), color = "#004266") +
    geom_line(data = filter(fixfx, cov == "Slope"), aes(x = s, y = val), color = "#d37aa7") +
    labs(
      color = NULL,
      x = "Timepoint (s)",
      y = "Effects"
    ) +
    geom_vline(xintercept = 3.2, linetype = "dashed") +
    theme_minimal() +
    theme(legend.position = "none")
```



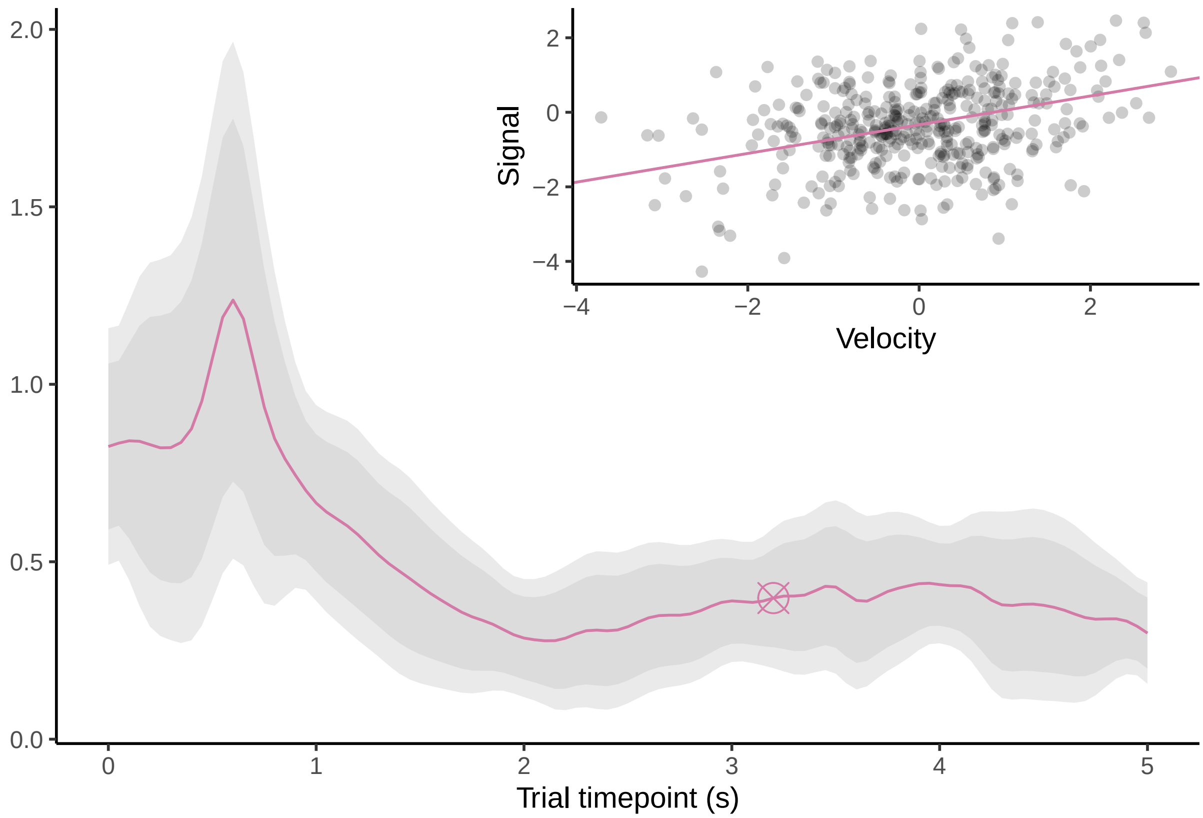
Insets

```r
# Linear model at time point
s_pt <- 3.2
x_pt <- s_pt * 20 - 1

fixed_corplot_df <- data.frame(
  x = dat[[paste0("x_", x_pt)]],
  y = dat[[paste0("y_", x_pt)]],
  id = dat$id,
  trial = dat$trial
) %>%
  filter(trial %in% 1:10)

fixed_corplot <- ggplot(fixed_corplot_df, aes(x = x, y = y)) +
  geom_point(alpha = 0.2) +
  geom_abline(
    slope = mod$betaHat[2, x_pt],
    intercept = mod$betaHat[1, x_pt],
    color = "#d37aa7"
  ) +
  labs(
    x = "Velocity",
    y = "Signal"
  ) +
  theme_classic()
```

Actual fixed effects The plot `fixed_corplot` can then be added into the dataset.

```r
fix_slope <- ggplot(fixfx %>% filter(cov == "Slope"), aes(x = s, y = val)) +
  geom_ribbon(aes(ymin = val - ci, ymax = val + ci), alpha = 0.07) +
  geom_ribbon(aes(ymin = val - joint_ci, ymax = val + joint_ci), alpha = 0.1) +
  # geom_line(color = "#d37aa7") +
  geom_line(color = "#d37aa7") +
  labs(
    x = "Trial timepoint (s)",
    y = NULL
  ) +
  geom_point(
    data = filter(fixfx, cov == "Slope", s == s_pt),
    shape = 13,
    size = 5,
    # color = "#d37aa7"
    color = "#d37aa7"
  ) +
  theme_classic()
```

```r
fix_slope +
  patchwork::inset_element(
    fixed_corplot,
    0.4, 0.55, 1, 1,
    align_to = 'full'
  )
```



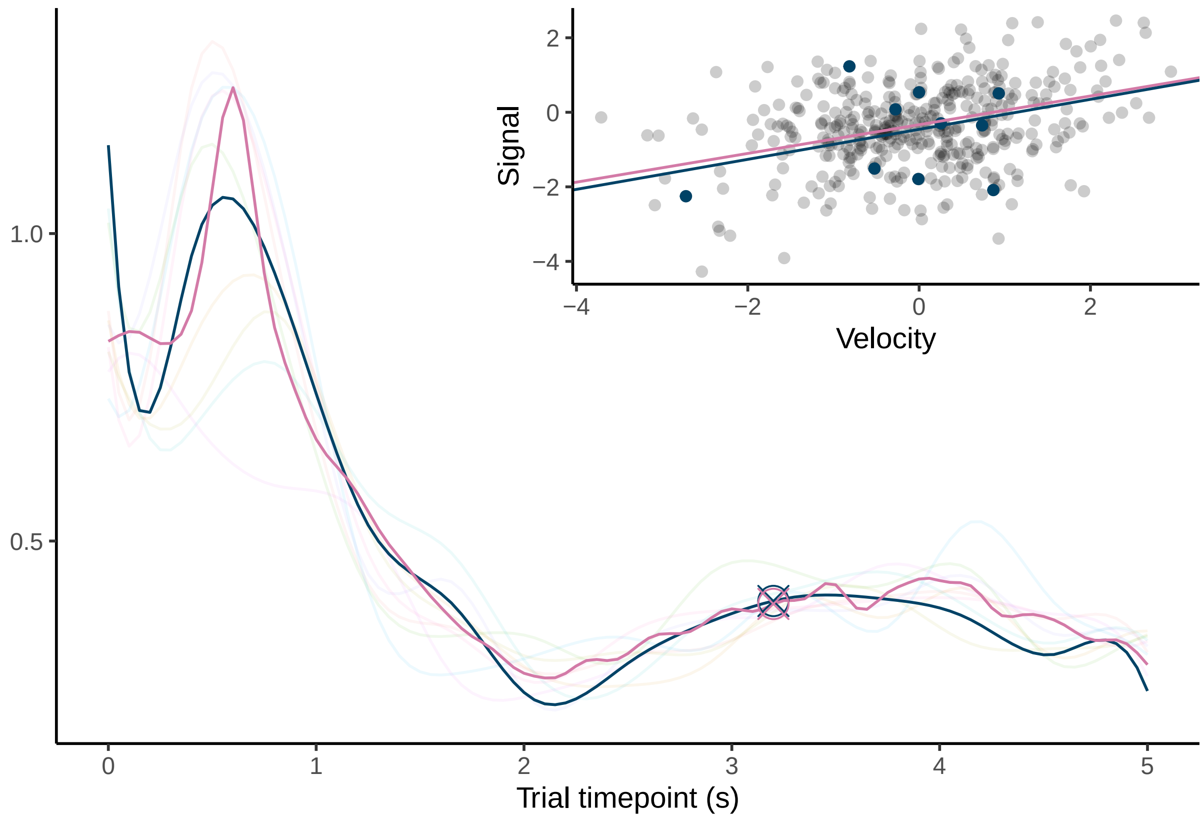

```
ggsave(paste0(img_dir, "inset_fixed_real.png"), width = 3.5, height = 3.5)
```

Actual random effects Insets can be used to illustrate the difference between fixed effects and a particular random effect:

```
rand_corplot <- ggplot(fixed_corplot_df, aes(x = x, y = y)) +
  geom_point(alpha = 0.2) +
  # ID 5 dots
  geom_point(data = fixed_corplot_df %>% filter(id == 5), color = "#004266") +
  # Fixed effect line
  geom_abline(
    slope = mod$betaHat[2, x_pt],
    intercept = mod$betaHat[1, x_pt],
    color = "#d37aa7"
  ) +
  # Rand effect line
  geom_abline(
    slope = smooth_rand_1 %>% filter(id == 5, s == s_pt) %>% .$val,
    intercept = smooth_rand_0 %>% filter(id == 5, s == s_pt) %>% .$val,
    color = "#004266"
  ) +
  labs(
    x = "Velocity",
    y = "Signal"
  ) +
  theme_classic()
```

```r
rand_slope <- smooth_rand_1 %>%
  ggplot(aes(x = s, y = val)) +
    geom_line(aes(color = id), alpha = 0.08) +
    geom_line(data = filter(smooth_rand_1, id == 5), aes(x = s, y = val), color = "#004266") +
    geom_line(data = filter(fixfx, cov == "Slope"), aes(x = s, y = val), color = "#d37aa7") +
    geom_point(
      data = filter(smooth_rand_1, s == s_pt, id == 5),
      shape = 13,
      size = 5,
      color = "#004266"
    ) +
    geom_point(
      data = filter(fixfx, cov == "Slope", s == s_pt),
      shape = 13,
      size = 5,
      color = "#d37aa7"
    ) +
    labs(
      color = NULL,
      x = "Trial timepoint (s)",
      y = NULL
    ) +
    theme_classic() +
    theme(legend.position = "none")
```

```r
rand_slope +
  patchwork::inset_element(
    rand_corplot,
    0.4, 0.55, 1, 1,
    align_to = 'full'
  )
```




```
ggsave(paste0(img_dir, "inset_rand_real.png"), width = 3.5, height = 3.5)
```

Synthetic insets

To make the trends clearer, it is possible to generate synthetic fixed and random effects.

Synthetic random effects

Synthetic random effects can be generated by adding a MVN of length L with covariance matrix given below.

```
rbfkernel <- function(x, y, sigma = 1, l = 15) {
  sigma^2 * exp(-(x - y)^2 / (2 * l^2))
}

fun_cov_sig <- matrix(0, nrow = L, ncol = L)

for (i in 1:L) {
  for (j in i:L) {
    fun_cov_sig[i, j] <- fun_cov_sig[j, i] <- rbfkernel(i, j)
  }
}

# Draws a single observation gamma_i, 1
gamma_add <- function(noisy = T, scalar = 0.1) {
  scalar * MASS::mvrnorm(n = 1, mu = rep(0, L), Sigma = fun_cov_sig)
```

```
}

rand_syn <- lapply(
  1:10,
  function(seed) {
    set.seed(seed)
    data.frame(
      id = seed,
      val = gamma_add() + mod$betaHat[2, ],
      s = (1:L - 1) / 20
    )
  }
)

rand_syn <- do.call(rbind, rand_syn) %>%
  mutate(id = as.factor(id))
rand_syn <- rbind(
  rand_syn,
  fixfx %>%
    filter(cov == "Slope") %>%
    select(-cov, -ci, -joint_ci) %>%
    mutate(id = "fixed")
)

# Sanity check
# ggplot(rand_syn, aes(x = s, y = val)) +
#   geom_line(aes(color = id), alpha = 0.2) +
#   geom_line(data = rand_syn %>% filter(id == "fixed"), color = "#d37aa7") +
#   geom_line(data = rand_syn %>% filter(id == 7), color = "#004266") +
#   theme_classic() +
#   theme(legend.position = "none")
```

Synthetic correlation plots

```
# Linear model at time point
s_pt <- 3.2
x_pt <- s_pt * 20 - 1
n_points <- 6
ids <- c(1:10, "fixed")

set.seed(1)
corpoints <- lapply(
  ids,
  function(i) {
    delta_y <- rand_syn %>%
      filter(id == i, s == s_pt) %>%
      .$val
    temp <- sample(fun_cov(), n_points)

    data.frame(
      x = temp,
      y = (temp * delta_y) + rnorm(n = n_points, mean = 0, sd = 0.2),
```

```r
      id = i
    )
  }
)

corpoints <- do.call(rbind, corpoints)

# Sanity check
# ggplot(corpoints, aes(x = x, y = y)) +
#   geom_point(aes(color = id))
```

Inset figures

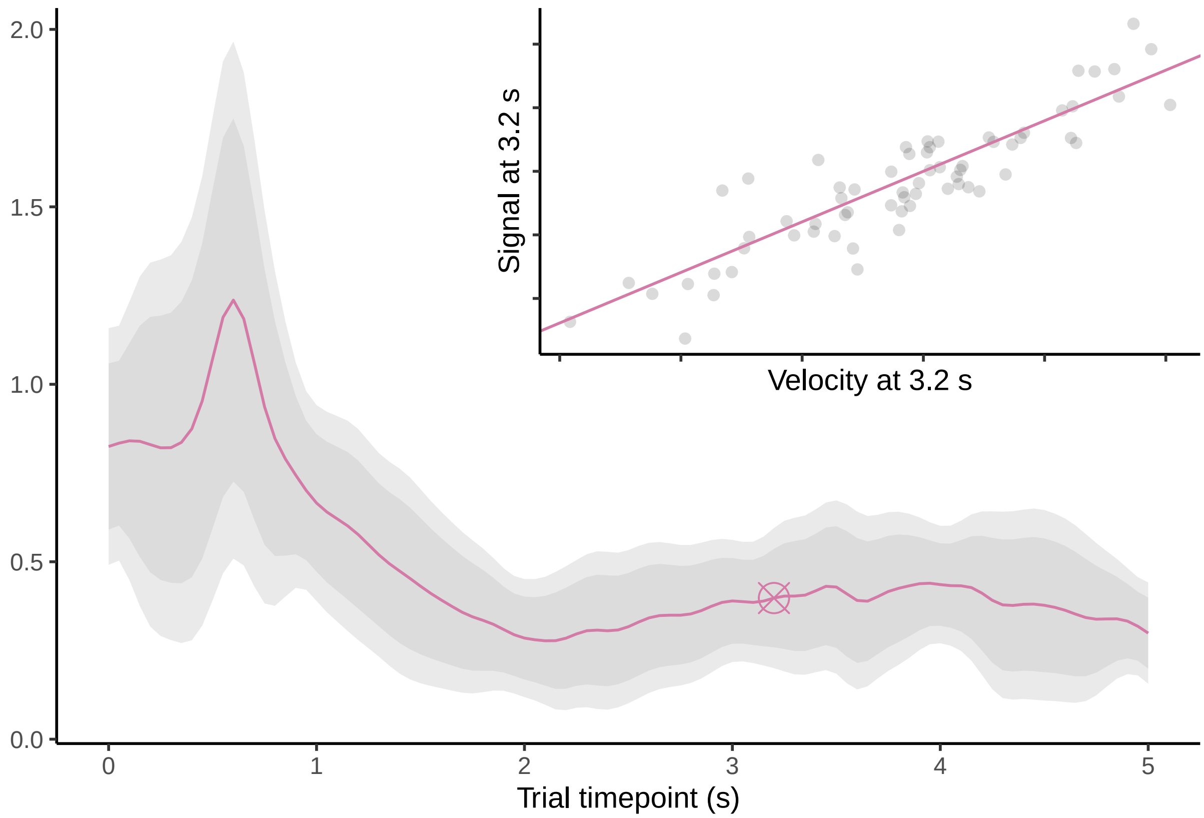
```r
fixed_corplot_syn <- ggplot(corpoints, aes(x = x, y = y)) +
  geom_point(alpha = 0.15) +
  # geom_point(data = corpoints %>% filter(id == "fixed"), color = "#d37aa7") +
  geom_abline(
    color = "#d37aa7",
    slope = rand_syn %>% filter(id == "fixed", s == s_pt) %>% .$val
  ) +
  labs(
    x = "Velocity at 3.2 s",
    y = "Signal at 3.2 s"
  ) +
  theme_classic() +
  theme(
    plot.background = NULL,
    axis.text = element_blank()
  )

rand_corplot_syn <- ggplot(corpoints, aes(x = x, y = y)) +
  geom_point(alpha = 0.1) +
  geom_abline(
    color = "#d37aa7",
    slope = rand_syn %>% filter(id == "fixed", s == s_pt) %>% .$val
  ) +
  #  #004266
  geom_point(data = corpoints %>% filter(id == 7), color = "#004266", alpha = 0.5) +
  geom_abline(
    color = "#004266",
    slope = rand_syn %>% filter(id == 7, s == s_pt) %>% .$val
  ) +
  labs(
    x = "Velocity at 3.2 s",
    y = "Signal at 3.2 s"
  ) +
  theme_classic() +
  theme(
    plot.background = NULL,
    axis.text = element_blank()
  )
```

Make insets

Make base plots The figure `fix_slope` is still fine as the base plot for the fixed effects inset.

```
rand_slope_syn <- rand_syn %>%
  ggplot(aes(x = s, y = val)) +
    geom_line(aes(group = id), alpha = 0.08) +
    geom_line(data = filter(rand_syn, id == 7), aes(x = s, y = val), color = "#004266") +
    geom_line(data = filter(rand_syn, id == "fixed"), aes(x = s, y = val), color = "#d37aa7") +
    geom_point(
      data = filter(rand_syn, s == s_pt, id == 7),
      shape = 13,
      size = 5,
      color = "#004266"
    ) +
    geom_point(
      data = filter(rand_syn, s == s_pt, id == "fixed"),
      shape = 13,
      size = 5,
      color = "#d37aa7"
    ) +
    labs(
      color = NULL,
      x = "Trial timepoint (s)",
      y = NULL
    ) +
    theme_classic() +
    theme(legend.position = "none")
```

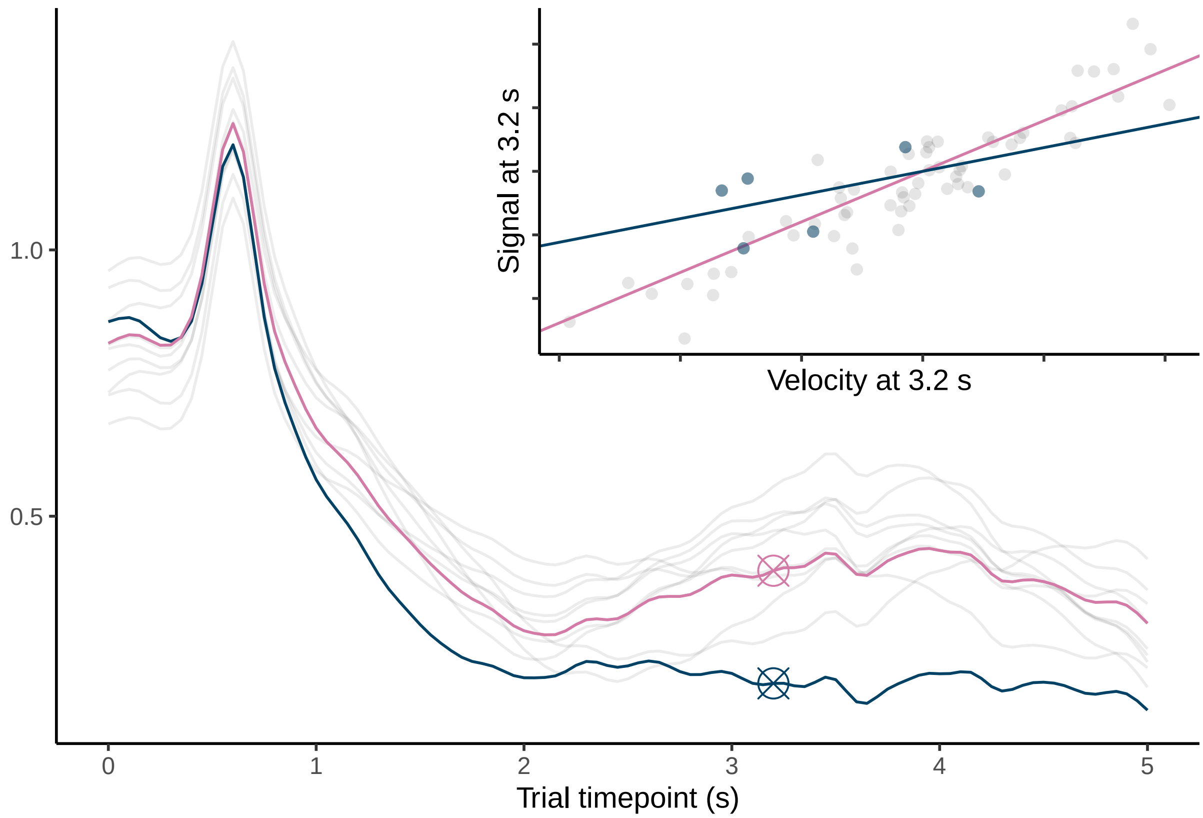
```
fix_slope +
  patchwork::inset_element(
    fixed_corplot_syn,
    0.4, 0.5, 1, 1,
    align_to = 'full'
  )
```



Join figures

```
ggsave(paste0(img_dir, "inset_fixed.png"), width = 3.5, height = 3.5)
```

```
rand_slope_syn +
  patchwork::inset_element(
    rand_corplot_syn,
    0.4, 0.5, 1, 1,
    align_to = 'full'
  )
```




```r
ggsave(paste0(img_dir, "inset_rand.png"), width = 3.5, height = 3.5)
```

Overall model

$$Y_{i,j}(s) = \beta_0(s) + \beta_1(s)X_{i,j}(s) + \gamma_i(s) + \epsilon_{i,j}(s)$$