Simulating Time-Varying Performance

Andrew Glover

2023-10-04

Libraries

```
library(ggplot2)
library(tibble)
```

Functions used for analysis

```
params_mat <- function(k_1, tau_1, k_2, tau_2, change_days=NULL, days) {
  if (length(k_1) == length(tau_1) &&
      length(k_1) == length(tau_2) &&
      length(k_1) == length(k_2) &&
      length(k_1) == length(change_days)+1
  ) {}
    else {stop("check length of parameters")}
  out_matrix <- matrix(0, nrow = days, ncol = 4)
  colnames(out_matrix) <- c("k_1", "tau_1", "k_2", "tau_2")
  bound_1 <- 1; bound_2 <- days
  j <- 0 # counter for index of k_1, tau_1, etc
  for (elem in c(change_days, days)) {
    j <- j + 1
    bound_2 <- elem
    for (i in bound_1:bound_2) {
      out_matrix[i, ] <- c(k_1[[j]], tau_1[[j]], k_2[[j]], tau_2[[j]])
    }
    bound_1 <- elem
  }
  return(out_matrix)
}
```

```
#' Title
#'
#' @param params_mat a n by 4 matrix, where n is the number of days
#' @param training_load an n dimensional vector
#'
#' @return an n? dimensional vector with the performance
#' @export
#'
```

```r
#' @examples
perf_tv <- function(p_0, params_mat, training_load) {
  days <- nrow(params_mat)
  perf_out <- c(rep(NA, days))
  T_1 <- 0; T_2 <- 0
  for (i in 1:days) {
    T_1 <- exp(-1/params_mat[i, "tau_1"])*T_1 + training_load[[i]]
    T_2 <- exp(-1/params_mat[i, "tau_2"])*T_2 + training_load[[i]]
    perf_out[[i]] <- p_0 + params_mat[i, "k_1"]*T_1 - params_mat[i, "k_2"]*T_2
  }
  return(perf_out)
}
```

```r
perf_plot <- function(p_0,
                      k_1,
                      tau_1,
                      k_2,
                      tau_2,
                      change_days = NULL,
                      days,
                      training_stim) {
  training_load <- c()
  limit <- 0
  if (training_stim[[1]] == "constant") {
    training_load <- c(rep(training_stim[[2]], days))
    k <- length(k_1)

    # See justification for this below
    limit <- p_0 + training_stim[[2]]*k_1[[k]]/(1-exp(-1/tau_1[[k]])) -
      training_stim[[2]]*k_2[[k]]/(1-exp(-1/tau_2[[k]]))
  }
  tmp_matrix <- params_mat(k_1,
                           tau_1,
                           k_2,
                           tau_2,
                           change_days,
                           days)
  modeled_performance <- perf_tv(p_0, tmp_matrix, training_load)
  tmp_data <- tibble(
    "day" = c(0:days),
    "performance" = c(p_0, modeled_performance),
    "limit" = c(rep(limit, days + 1))
  )

  plot <- ggplot(tmp_data, aes(x = day)) +
    geom_line(aes(y = performance, color = "perf")) +
    geom_line(aes(y = limit, color = "lim"))

  #   scale_color_manual("Legend",
  #                      values = c("lim" = "#e31a1c", # this color comes from the theme "Paired"
  #                                 "perf" = "black"))
  plot
}
```

Computing the limit of the model

I would like to compute the limit of the predicted performance for the time-invariant model Under the assumption of constant training load. We have

$$p(t) = p_0 + k_1 \sum_{i=1}^{t-1} e^{\frac{t-i}{\tau_1}} w(i) + k_2 \sum_{i=1}^{t-1} e^{\frac{t-i}{\tau_2}} w(i)$$

If $w(i) = C$ for all i, then we can write

$$\sum_{i=1}^{t-1} e^{\frac{t-i}{\tau_1}} = e^{1/\tau_1} \sum_{i=1}^{t-1} e^{\frac{(t-1)-i}{\tau_1}} = e^{1/\tau_1} \left(-1 + \sum_{i=0}^{s} \left(e^{-1/\tau_1} \right)^i \right)$$

Finding the long-run limit of $p(t)$ then amounts to computing

$$\sum_{i=0}^{\infty} \left(e^{-1/\tau_1} \right)^i = \frac{1}{1 - e^{-1/\tau_1}}$$

Notice that this is a convergent geometric series, $e^{-1/\tau_1} < 1$ when $\tau_1 > 1$ (which we have assumed). Therefore,

$$\sum_{i=1}^{t-1} e^{\frac{t-i}{\tau_1}} = e^{1/\tau_1} \left(-1 + \frac{1}{1 - e^{-1/\tau_1}} \right) = e^{1/\tau_1} \left(\frac{e^{-1/\tau_1}}{1 - e^{-1/\tau_1}} \right) = \frac{1}{1 - e^{-1/\tau_1}}$$

Therefore

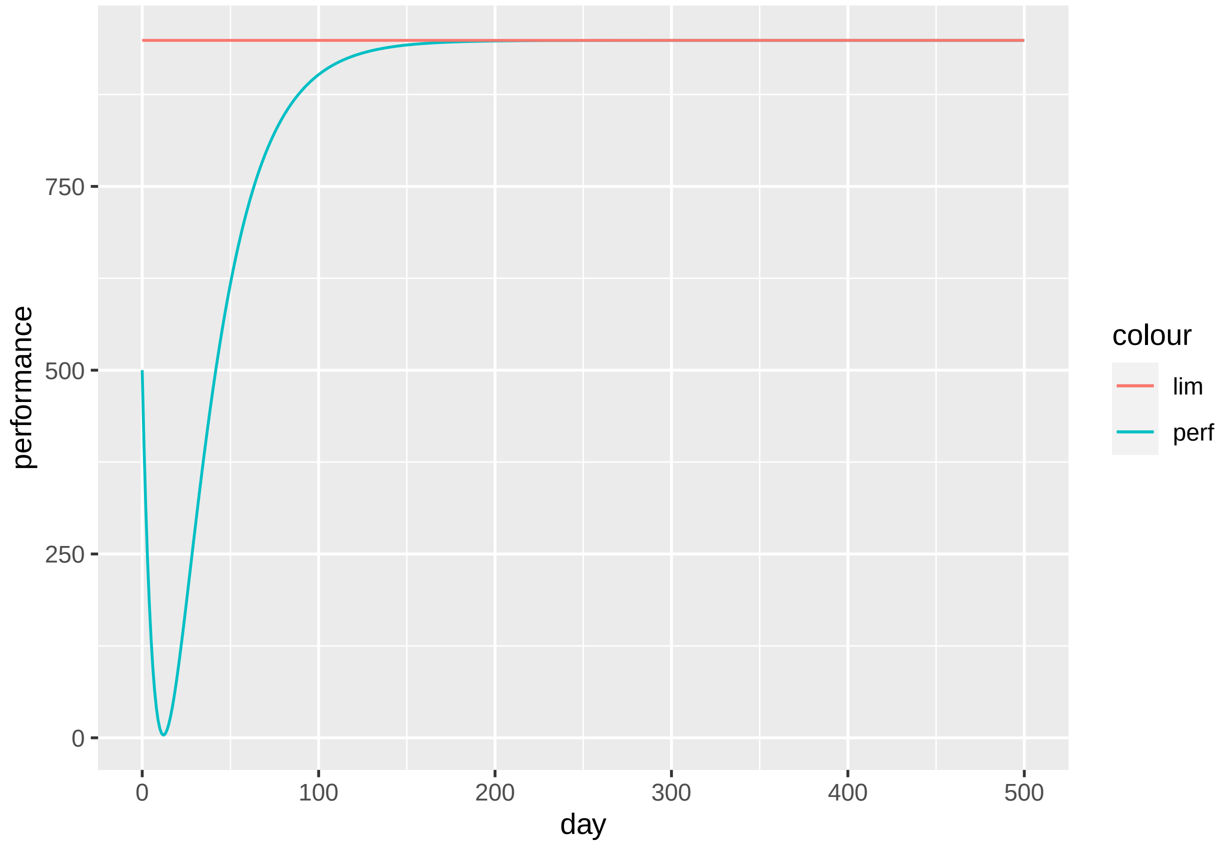
$$\lim_{t \to \infty} p(t) = p_0 + C \left(\frac{k_1}{1 - e^{-1/\tau_1}} - \frac{k_2}{1 - e^{-1/\tau_2}} \right)$$

This is how we compute the red line in `perf_plot`.

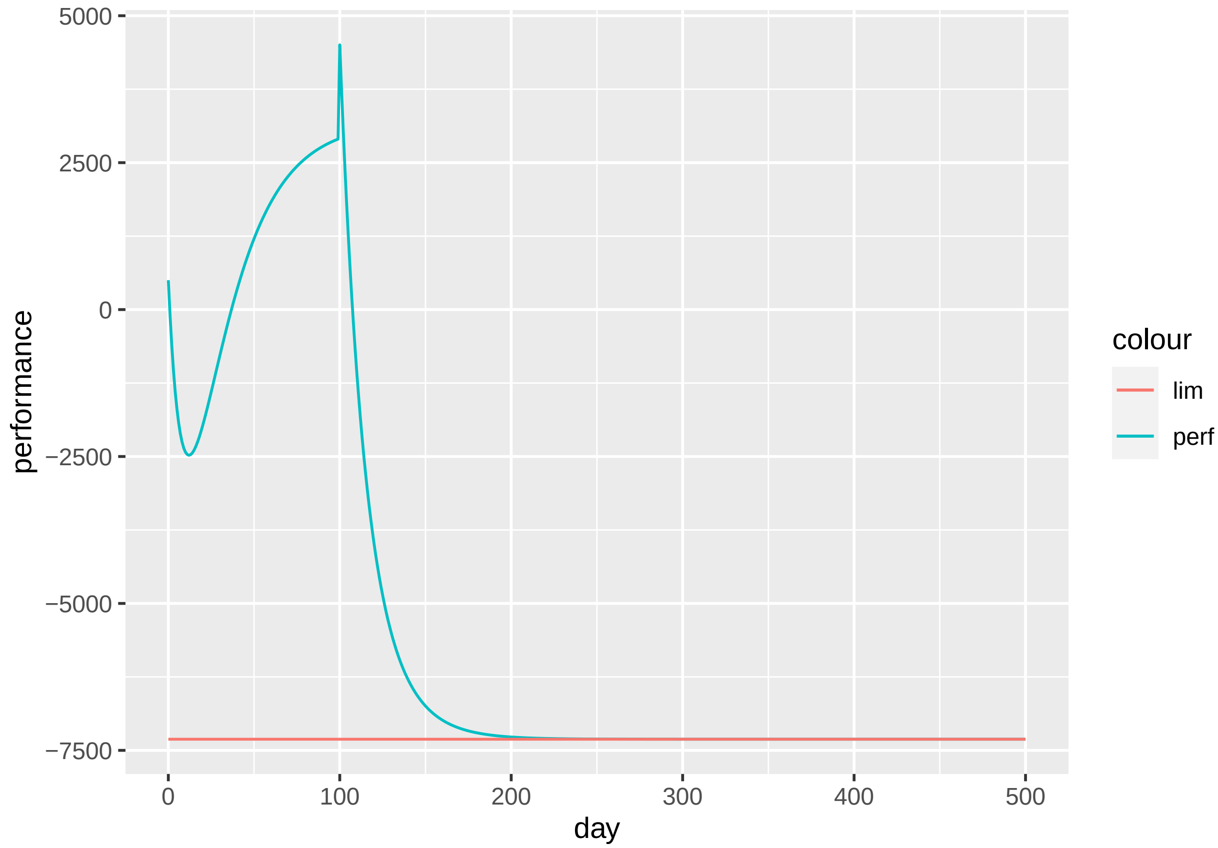
Exploring the time, varying model

This is a time-invariant plot with our new function, to check that things are working correctly

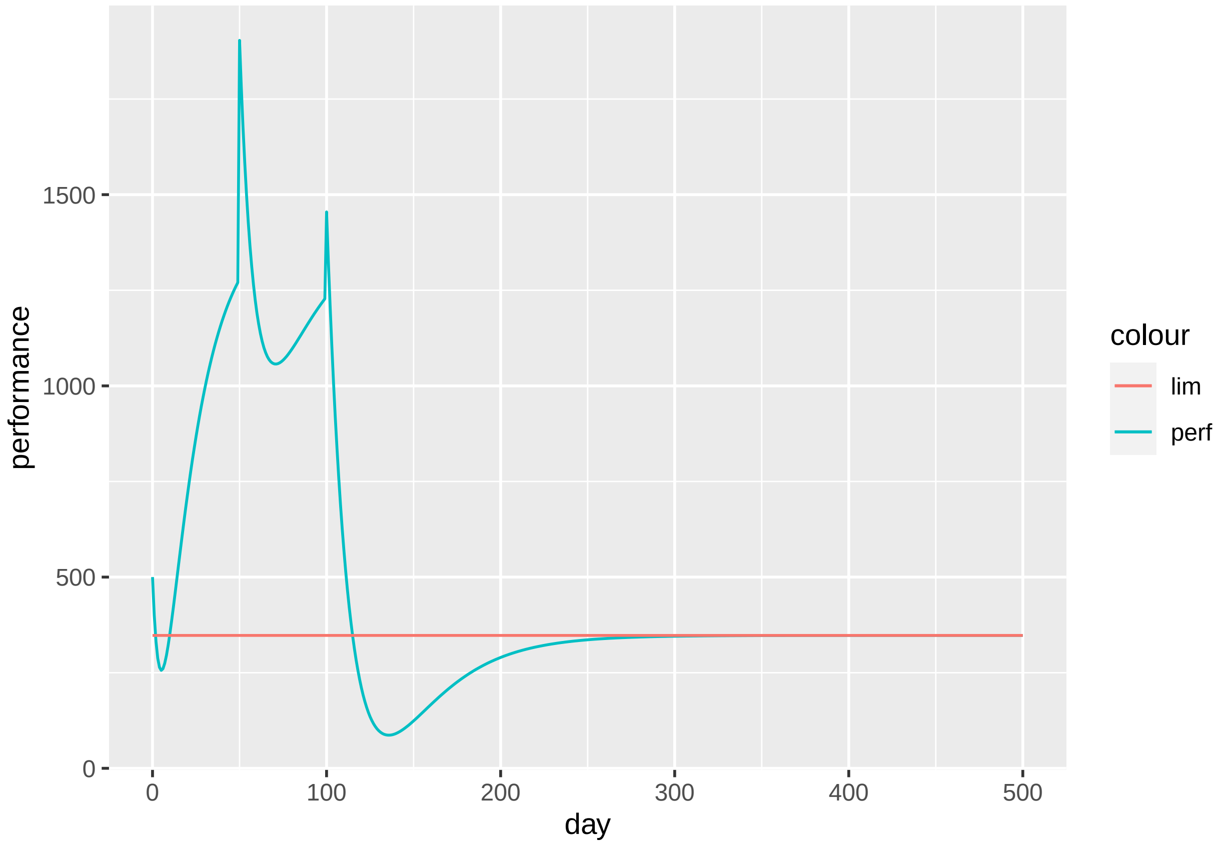
```
perf_plot(p_0 = 500,
          k_1 = 1,
          tau_1 = 25,
          k_2 = 2,
          tau_2 = 10,
          days = 500,
          training_stim = list("constant", 100))
```




```r
perf_plot(p_0 = 500,
          k_1 = c(1, 2),
          tau_1 = c(25, 20),
          k_2 = c(2, 4),
          tau_2 = c(10, 13) ,
          change_days = c(100),
          days = 500,
          training_stim = list("constant", 600))
```




```r
perf_plot(p_0 = 500,
          k_1 = c(1,2,3),
          tau_1 = c(20, 25, 30),
          k_2 = c(2, 4, 6),
          tau_2 = c(5,10, 15),
          change_days = c(50, 100),
          days = 500,
          training_stim = list("constant", 100))
```



To have the effects of an initial negative effect, and a long-run positive effect to a constant stimulus, it seems to be necessary that $k_1 < k_2$ and τ_1 has to be much bigger than