Woods\_Thesis\_App\_Final.R

Phil

2022-11-06

## Masters Thesis: Stretch activation and fatigue   
## App for analyzing Amplitude and Rates  
## Philip C. Woods  
## Created: 3/11/22

library(shiny)  
library(shinythemes)  
library(dygraphs)  
library(tidyverse)  
library(readxl)  
library(RcppRoll)  
library(RColorBrewer)  
library(writexl)  
library(minpack.lm)  
library(ggpubr)  
library(broom)  
theme\_set(theme\_classic())  
  
get\_seperate\_phases <- function(mdl\_tidy, time0){  
 opt\_a <- filter(mdl\_tidy, term == 'a')  
 opt\_b <- filter(mdl\_tidy, term == 'b')  
 opt\_c <- filter(mdl\_tidy, term == 'c')  
 opt\_d <- filter(mdl\_tidy, term == 'd')  
 opt\_e <- filter(mdl\_tidy, term == 'e')  
 opt\_g <- filter(mdl\_tidy, term == 'g')  
   
 p2 <- opt\_a$estimate \* exp(-opt\_b$estimate \* time0)  
 p3 <- opt\_c$estimate \* (1 - exp(-opt\_d$estimate \* time0))  
 p4 <- opt\_e$estimate \* exp(-opt\_g$estimate \* time0)  
   
 phase2 <- data.frame(time0 = time0,  
 Force\_One = p2,  
 phase = '2')  
   
 phase3<- data.frame(time0 = time0,  
 Force\_One = p3,  
 phase = '3')  
 phase4 <- data.frame(time0 = time0,  
 Force\_One = p4,  
 phase = '4')  
   
 rbind(phase2,phase3,phase4)  
}  
  
  
ui <- fluidPage(  
 theme = shinytheme("yeti"),  
   
 ## Conditional panel section, including all action buttons  
 titlePanel("Stretch Activation and Fatigue - Woods' Masters Thesis"),  
 sidebarPanel(  
 fileInput(inputId = "file",  
 label = "Select a file"),  
 actionButton("load\_file",   
 "Load File")),  
   
   
 conditionalPanel(  
 condition = "input.tabselected==1",  
 actionButton("set\_phase\_3",   
 "Set Phase 3"),  
 downloadButton("download\_amp",  
 "Download Amplitude"),  
 downloadButton("download\_amp\_values",  
 "Download Phase 3 Values")),  
  
 conditionalPanel(  
 condition = "input.tabselected==2",  
 actionButton("set\_rate\_phases",   
 "Set Phases 2-4"),  
 downloadButton("download\_rate",  
 "Download Rates"),  
 downloadButton("download\_rate\_values",  
 "Download Rate Values")),  
   
   
 # mainPanel (what will show up in center after actionbuttons are clicked)  
 mainPanel(  
 dygraphOutput("interactive\_plot"),  
   
 tabsetPanel(id = "tabselected",  
 tabPanel("Amplitude", value = 1,  
 plotOutput("phase\_3"),  
 tableOutput("amp\_datatable")),  
   
 tabPanel("Rates", value = 2,  
 plotOutput("fit"),  
 plotOutput("fit\_split"),  
 tableOutput("rates"))  
 )  
 )  
)  
  
  
  
server <- function(input, output){  
   
 ## creating reactiveValues that will be stored following certain actions   
 user <- reactiveValues()   
   
   
 ## loading in file and ploting dygraph  
 observeEvent(input$load\_file, {  
   
 user$data <- read\_excel(input$file$datapath, skip = 29)  
 })  
   
 output$interactive\_plot <- renderDygraph({  
 validate(need(user$data, "Please upload data to begin"))  
 df <- data.frame(Seconds = user$data$Time,  
 Force = user$data$Force\_One)  
 dygraph(df, xlab = "Seconds", ylab = "Force") %>%   
 dyRangeSelector()  
 })  
   
   
 ## Instructions on setting Phase 3 amplitude after clicking Set Phase 3 button  
 observeEvent(input$set\_phase\_3, {  
 req(user$data)   
 if(!is.null(input$interactive\_plot\_date\_window)){  
   
 user$phase\_3\_boundaries <- c(input$interactive\_plot\_date\_window[[1]],  
 input$interactive\_plot\_date\_window[[2]])  
   
 user$data$force\_one\_smooth <- RcppRoll::roll\_meanl(x = user$data$Force\_One, n = 16)  
   
 user$phase\_3\_data <- user$data %>%   
 filter(Time >= user$phase\_3\_boundaries[[1]] & Time <= user$phase\_3\_boundaries[[2]])  
   
 user$phase\_3\_max\_force <- max(user$phase\_3\_data$force\_one\_smooth)[[1]]  
   
 user$phase\_3\_max\_x\_index <- user$data[which(user$data$force\_one\_smooth == user$phase\_3\_max\_force), ]  
   
 user$phase\_3\_total\_time <- user$phase\_3\_boundaries[[2]] + 0.1  
   
 user$amp\_parameters <- data.frame(user$phase\_3\_boundaries[[1]],  
 user$phase\_3\_boundaries[[2]],  
 user$phase\_3\_max\_force,  
 round(user$phase\_3\_max\_force, 6)\*1000,  
 user$phase\_3\_max\_x\_index$Time)  
   
 m <- list("Phase 3 Boundary 1",  
 "Phase 3 Boundary 2",  
 "Phase 3 Max Force, mN",  
 "Phase 3 Max Force, mN\*1000",  
 "Phase 3 Max Index")  
   
 names(user$amp\_parameters) <- m  
   
 colorz <- RColorBrewer::brewer.pal(8, "Dark2")  
   
 df1 <- filter(user$data, Time <= user$phase\_3\_total\_time)   
   
 user$plot\_amp <- ggplot() +  
   
 geom\_line(data = df1,  
 aes(x = Time,  
 y = Force\_One)) +  
   
 geom\_line(data = df1,  
 aes(x = Time,  
 y = force\_one\_smooth),  
 size = 1,  
 color = colorz[[1]]) +  
   
 geom\_errorbarh(aes(xmin = user$phase\_3\_boundaries[[1]],  
 xmax = user$phase\_3\_boundaries[[2]],  
 y = user$phase\_3\_max\_force),  
 height = 0.001,  
 color = colorz[[2]],  
 size = 1) +  
   
 geom\_point(aes(x = user$phase\_3\_max\_x\_index$Time,  
 y = user$phase\_3\_max\_force),  
 color = colorz[[2]],  
 size = 4) +  
   
 theme\_linedraw(20)  
 }  
 })  
   
   
 ## Rate fitting code following selection of "Set Phases 2-4" button   
 observeEvent(input$set\_rate\_phases, {  
 req(user$data)  
 if(!is.null(input$interactive\_plot\_date\_window)) {  
   
 user$rate\_phases\_boundaries <- c(input$interactive\_plot\_date\_window[[1]],  
 input$interactive\_plot\_date\_window[[2]])  
   
 user$rate\_phases\_data <- user$data %>%   
 filter(Time >= user$rate\_phases\_boundaries[[1]],   
 Time <= user$rate\_phases\_boundaries[[2]]) %>%   
 mutate(time0 = Time - Time[1], .before = Force\_One)  
   
 phase2 <- user$rate\_phases\_data %>%   
 filter(time0 <= time0[16])  
   
 phase2\_linfit <- lm(log10(phase2$Force\_One) ~ phase2$time0)  
   
 phase2$lm <- predict(phase2\_linfit)  
   
 phase2\_model <- nlsLM(Force\_One ~ (a\*exp(-b\*time0)),  
 data = phase2,  
 start = list(a = (10^phase2\_linfit$coefficients[[1]]),  
 b = (-phase2\_linfit$coefficients[[2]])/(log10(exp(1)))),  
 control = nls.control(maxiter = 100))  
   
 phase2\_mdl\_summary <- broom::tidy(phase2\_model)  
  
 user$grd <- list(a = phase2\_mdl\_summary$estimate[[1]],  
 b = phase2\_mdl\_summary$estimate[[2]],  
 c = tail(user$rate\_phases\_data$Force\_One, n=1),  
 d = phase2\_mdl\_summary$estimate[[2]]/2,  
 e = phase2\_mdl\_summary$estimate[[1]],  
 g = phase2\_mdl\_summary$estimate[[2]]/4)  
  
   
 mdl <- nlsLM(Force\_One ~ (a\*exp(-b\*time0))+  
 (c\*(1.0-exp(-d\*time0))) +  
 (e\*exp(-g\*time0)),  
 data = user$rate\_phases\_data,  
 start = user$grd,  
 control = nls.control(maxiter = 100))  
   
 user$rate\_phases\_data$fit <- predict(mdl)  
   
 user$mdl\_tidy <- broom::tidy(mdl)  
  
 user$plot\_rates <- ggplot(data = user$rate\_phases\_data,  
 aes(x = time0, y = Force\_One)) +  
  
 geom\_point() +  
  
 geom\_line(aes(y = fit),   
 size = 0.8,   
 col = 'red') +  
  
 ggtitle("Fit")  
  
 fits\_seperated <- get\_seperate\_phases(user$mdl\_tidy,  
 user$rate\_phases\_data$time0)  
  
 user$plot\_rates\_seperated <- ggplot() +  
 geom\_line(data = fits\_seperated,  
 aes(x = time0,   
 y = Force\_One,   
 color = phase)) +  
 geom\_line(data = user$rate\_phases\_data,  
 aes(x = time0, y = fit),   
 size = 0.8,   
 col = "red") +  
 ggtitle("Fit Seperated")  
   
 user$plot\_rates\_comb <- ggarrange(user$plot\_rates,  
 user$plot\_rates\_seperated,  
 ncol=1)  
  
 user$rate\_parameters <- list(data.frame(user$grd),  
 data.frame(user$rate\_phases\_data),  
 user$mdl\_tidy)  
  
 names(user$rate\_parameters) <- list("Starting Parameters",  
 "Fitted Data",  
 "Model")  
   
   
 }  
 })  
   
 ## Output code  
   
 # Phase 3 Amplitude  
   
 output$phase\_3 <- renderPlot({  
 req(user$plot\_amp)  
 user$plot\_amp  
 })   
   
 output$amp\_datatable <- renderTable({  
 req(user$amp\_parameters)  
 user$amp\_parameters  
 })   
   
 output$download\_amp <- downloadHandler(  
 filename = function() {  
 paste("Woods\_MXXFxxCxx\_P3\_ggplot", '.pdf', sep = '')  
 },  
 content = function(file) {  
 ggsave(filename = file, plot = user$plot\_amp)  
 }  
 )  
  
 output$download\_amp\_values <- downloadHandler(  
 filename = function() {  
 paste("Woods\_MXXFxxCxx\_P3\_Parameters", '.xlsx', sep = '')  
 },  
 content = function(file) {  
 writexl::write\_xlsx(user$amp\_parameters, path = file)  
 }  
 )  
   
   
 # Rate fittings   
   
 output$fit <- renderPlot({  
 req(user$plot\_rates)  
 user$plot\_rates  
 })  
  
 output$fit\_split <- renderPlot({  
 req(user$plot\_rates\_seperated)  
 user$plot\_rates\_seperated  
 })  
   
 output$rates <- renderTable({  
 req(user$mdl\_tidy)  
 user$mdl\_tidy  
 })   
   
 output$download\_rate <- downloadHandler(  
 filename = function() {  
 paste("Woods\_MXXFxxCxx\_Rates\_ggplot", '.pdf', sep ='')  
 },  
 content = function(file) {  
 ggsave(filename = file, plot = user$plot\_rates\_comb)  
 }  
 )  
  
 output$download\_rate\_values <- downloadHandler(  
 filename = function() {  
 paste("Woods\_MXXFxxCxx\_Rates\_Parameters", '.xlsx', sep = '')  
 },  
 content = function(file) {  
 writexl::write\_xlsx(user$rate\_parameters, path = file)  
 }  
 )  
   
 }  
  
shinyApp(ui = ui, server = server)