## ****UDMTA - A shiny App for Species Annual Temporal Abundance Models****

Udani Wijewardhana

# Loading libraries  
library(shiny)  
library(DT)

library(plyr)  
library(dplyr)

library(leaflet)  
library(INLA)

## Loading required package: Matrix

## Loading required package: sp

## Loading required package: parallel

## Loading required package: foreach

## This is INLA\_20.04.18 built 2020-04-28 22:41:54 UTC.  
## See www.r-inla.org/contact-us for how to get help.

#############################################################################  
# Shiny App for Annual Species Temporal Abundance Models   
#############################################################################  
  
# First the user needs to upload the data csv file into the application and   
# then select whether normalize the numerical predictors or not.  
# The data file should include only:  
# 1. Species - Different species  
# 2. Year - Detected Year  
# 3. Count - Species count  
# with or without predictor variables (numeric/factor).  
# The above names are case sensitive."),  
# A sample format of the data can be found in

# https://github.com/uwijewardhana/UDMTA.  
  
### Shiny User Interface ###  
  
ui <- fluidPage(  
   
titlePanel(strong("UDMTA - A shiny App for Annual Species Temporal Abundance Models", titleWidth = 350)),  
   
# Loading the data file  
div(style="display: inline-block;vertical-align:top; width: 300px;", fileInput("file", "Choose data CSV File", multiple = FALSE, accept = c("text/csv", "text/comma-separated-values,text/plain", ".csv"))),  
div(style="display: inline-block;vertical-align:top; width: 300px;", selectInput("prednorm", "Numeric predictors normalization:", choices=c("rnorm", "stand", "none"), selected = "none")),  
   
tabsetPanel(  
   
tabPanel("Data",  
 fluidRow(style = "margin-top: 25px;",  
 column(8, p(tags$b('Annual Numeric Data', style = 'font-size: 150%; font-family:Helvetica; color:#4c4c4c; text-align:left;'))),  
 column(4, p(tags$b('Summary of Numeric Predictors', style = "font-size: 150%; font-family:Helvetica; color:#4c4c4c; text-align:left;")))),  
 fluidRow(column(8, DT::dataTableOutput("contents")),  
 column(4, verbatimTextOutput("datasummary")))  
),  
   
tabPanel("Species Distribution Model",  
 sidebarLayout(  
 sidebarPanel(div(style='height:950px; overflow: scroll',  
 selectInput("distribution", "Distribution:", choices=c("Poisson", "Negative Binomial","Zeroinflated Poisson", "Zeroinflated Negative Binomial",  
"Poisson Hurdle", "Negative Binomial Hurdle"), selected = "Poisson"),  
 selectInput("tempeffect", "temporal random effect model:", choices=c("'ar1'", "'iid'", "'rw1'", "'rw2'"), selected = "'ar1'"),  
 selectInput("factor", "Include factor variables in the model:", choices=c("No", "Yes"), selected = "No"),  
 h5('Generate Interaction Variables Here (if applicable)'),  
 uiOutput("independent"),  
 uiOutput("makeInteract1"), uiOutput("makeInteract2"),  
 uiOutput("uiAdded"), actionButton("actionBtnAdd", "Create Interaction Term"),  
 hr(),  
 actionButton("summary", "Summary"))),  
   
mainPanel(fluidRow(column(12, p(tags$b('Summary results of species distribution model:', style = "font-size: 150%; font-family:Helvetica; color:#4c4c4c; text-align:left;")))),  
 fluidRow(column(12, verbatimTextOutput("summary"))))  
  
))))

### Shiny Server ###  
  
server <- function(input, output, session){  
   
# Read Data CSV file  
   
filedata1 <- reactive({  
 inFile <- input$file  
 if (is.null(inFile)){return(NULL)}  
   
 x <- as.data.frame(read.csv(inFile$datapath, fileEncoding="UTF-8-BOM"))  
 x$Count <- as.character(x$Count)  
 x$Count <- as.numeric(x$Count)  
  
 y = dplyr::select\_if(x, is.numeric)  
 z = cbind(Species = x[ , (names(x) %in% c("Species"))], y)  
 Final <- unique(z)  
})  
   
# Subset possible numeric predictor variables  
   
filedata2 <- reactive({  
 req(input$file)  
 x <- filedata1()  
   
 y = dplyr::select\_if(x, is.numeric)  
 if(ncol(y)>2){  
 p = subset(y, select = -c(Count))  
 p <- unique(p)  
 p = subset(p, select = -c(Year))  
 }else {p = NULL}  
   
 if(!is.null(p)){  
 for(i in 1:ncol(p)){  
 if(input$prednorm == "rnorm"){p[,i] <- round(rnorm(p[,i]), digits = 4)  
 } else if(input$prednorm == "stand"){p[,i] <- round(scale(p[,i]), digits = 4)  
 } else {p[,i] <- round(p[,i], digits = 4)}}  
 }   
 return(p)  
})  
   
# Output of the data table  
   
output$contents <- DT::renderDataTable({  
req(input$file)  
df <- filedata1()  
return(DT::datatable(df, options = list(scrollX = TRUE)))  
})  
   
# Output of the numeric predictors summary table  
   
output$datasummary <- renderPrint({  
 req(input$file)  
 df <- filedata2()  
 if (is.null(df)){return(NULL)}  
 return(summary(df))  
})  
   
# Rendering the list to the ui  
   
output$uiAdded <- renderUI({checkboxGroupInput('added', 'List of combinations', choices = names(interacts))})  
   
# The main named list that will be used in other tasks  
interacts <- reactiveValues()  
makeReactiveBinding("interacts")  
   
observe({  
 input$actionBtnAdd # Trigger Add actions  
 isolate({  
 a <- c(input$makeInteract1,input$makeInteract2)  
 b <- a %>% paste(collapse = "\*")  
 if(b != "")  
 interacts[[b]] <- a  
})})

# Create dataframe for regression with only numeric variables  
num <- reactive({  
 inFile <- input$file  
 if (is.null(inFile)){return(NULL)}  
 x <- as.data.frame(read.csv(inFile$datapath, fileEncoding="UTF-8-BOM"))  
  
 if(input$distribution == "Poisson Hurdle" | input$distribution == "Negative Binomial Hurdle"){  
 x$Count[x$Count == 0] <- NA  
 } else {  
 x$Count = x$Count  
 }  
  
 y = dplyr::select\_if(x, is.numeric)  
   
 if(ncol(y)>2){  
 p = subset(y, select = -c(Count))  
 p <- unique(p)  
 p = subset(p, select = -c(Year))  
 }else {p = NULL}  
  
 if(!is.null(p)){  
 for(i in 1:ncol(p)){  
 if(input$prednorm == "rnorm"){p[,i] <- round(rnorm(p[,i]), digits = 4)  
 }else if(input$prednorm == "stand") {p[,i] <- round(scale(p[,i]), digits = 4)  
 }else {p[,i] <- p[,i]}}}  
   
 d1 = cbind(Year = unique(x$Year), p, effect = unique(x$Year))  
 d2 <- aggregate(Count ~ Species + Year, x, FUN = sum)  
 d2$ID <- paste(d2$Species, d2$Year, sep = "-", collapse = NULL)  
 d3 <- d1[rep(seq\_len(nrow(d1)), length(unique(x$Species))), ]  
 d3$Species <- rep(unique(x$Species), each = length(unique(x$Year)))  
 d3$ID <- paste(d3$Species, d3$Year, sep = "-", collapse = NULL)  
 d4 <- join(d3, d2, by = "ID", type = "left", match = "all")  
 d4 <- d4[order(d4$Species, d4$Year),]  
 d3 <- d3[ , !(names(d3) %in% c("ID"))]  
 Count = d4$Count  
 Final <- cbind(d3, Count)  
   
 return(Final)  
})

# Create dataframe for regression with categorical variables  
fac <- reactive({  
 inFile <- input$file  
 if (is.null(inFile)){return(NULL)}  
 x <- as.data.frame(read.csv(inFile$datapath, fileEncoding="UTF-8-BOM"))  
   
 fac = data.frame(x %>% select\_if(~ !((is.integer(.x)) | (is.numeric(.x)))))  
 for(i in 1:ncol(fac)){fac[,i] = as.factor(fac[,i])}   
 y = dplyr::select\_if(x, is.numeric)  
 x <- cbind(y,fac)  
   
 if(input$distribution == "Poisson Hurdle" | input$distribution == "Negative Binomial Hurdle"){  
 x$Count[x$Count == 0] <- NA  
 } else {  
 x$Count = x$Count  
 }  
  
 if(ncol(y)>2){  
 p = subset(y, select = -c(Count))  
 p <- unique(p)  
 p = subset(p, select = -c(Year))  
 }else {p = NULL}  
   
 if(!is.null(p)){  
 for(i in 1:ncol(p)){  
 if(input$prednorm == "rnorm"){p[,i] <- round(rnorm(p[,i]), digits = 4)  
 }else if(input$prednorm == "stand") {p[,i] <- round(scale(p[,i]), digits = 4)  
 }else {p[,i] <- p[,i]}}}  
   
 xx = cbind(Year = unique(x$Year), p, effect = unique(x$Year))  
   
 if(is.null(p)){  
 Final = x  
 }else {  
 z = dplyr::select\_if(x, is.factor)  
 Count <- x[ , (names(x) %in% c("Count"))]  
 n = nrow(x)/length(unique(x$Year))  
 p <- xx[rep(seq\_len(nrow(xx)), n), ]  
   
 Final = cbind(p, Count, z)  
 }  
 return(Final)  
})  
  
# Checkbox list of all numeric variables to use   
independent <- reactive({  
 if(!is.null(input$file)){  
 inFile <- input$file  
  
 x <- as.data.frame(read.csv(inFile$datapath, fileEncoding="UTF-8-BOM"))  
 df = x[ , !(names(x) %in% c("Count", "Species"))]  
 return(names(df))  
 }  
})  
   
output$independent <- renderUI({checkboxGroupInput("independent", "Independent (Predictor) Variables:", independent())})  
   
# Variables to Add to the List of Combinations   
makeInteract <- reactive({  
 if(!is.null(input$file)){  
 inFile <- input$file  
  
 x <- as.data.frame(read.csv(inFile$datapath, fileEncoding="UTF-8-BOM"))  
 df = x[ , !(names(x) %in% c("Count", "Species"))]   
 return(names(df))  
 }  
})  
   
output$makeInteract1 <- renderUI({selectInput("makeInteract1", "Variable1 For Interaction:", makeInteract())})  
output$makeInteract2 <- renderUI({selectInput("makeInteract2", "Variable2 For Interaction:", makeInteract())})  
  
# distribution  
distribution <- reactive({  
 if(input$distribution == "Poisson"){distribution = "poisson"  
 } else if(input$distribution == "Negative Binomial"){distribution = "nbinomial"  
 } else if(input$distribution == "Zeroinflated Poisson") {distribution = "zeroinflatedpoisson1"  
 } else if(input$distribution == "Zeroinflated Negative Binomial") {distribution = "zeroinflatednbinomial1"  
 } else if(input$distribution == "Poisson Hurdle") {distribution = "zeroinflatedpoisson0"  
 } else {distribution = "zeroinflatednbinomial0"}  
 return(distribution)  
})  
  
# formula  
formula <- reactive({  
 if(!is.null(input$added)){  
 formula = paste("Count ~ 1 +", paste(input$independent, collapse = "+"),   
 paste("+", paste(input$added, collapse = "+")),   
 paste("+", "f(effect, model = ", input$tempeffect, ")"))  
 }else {  
 formula = paste("Count ~ 1 + ", paste(input$independent, collapse = "+"),   
 paste("+", "f(effect, model = ", input$tempeffect, ")"))  
 }  
 return(formula)  
})  
  
# Fit SDM using R-INLA  
   
fitsummary <- reactive({  
   
 df1 <- as.data.frame(fac())  
 df2 <- as.data.frame(num())  
   
 model <- list()  
 results <- list()  
 lst1 <- split(df1, df1$Species)   
 lst2 <- split(df2, df2$Species)   
   
 if(input$factor == "Yes"){  
   
 model <- lapply(seq\_along(1:length(unique(df1$Species))), function(x)  
 inla(as.formula(formula()), data = lst1[[x]], family = distribution(),control.family = list(link = "log"),  
 control.compute = list(dic = TRUE, cpo = TRUE, config = TRUE)))  
 results <- lapply(seq\_along(1:length(unique(df1$Species))), function(x) model[[x]]$summary.fixed[,c(1:3,5)])  
   
 }else {  
   
 model <- lapply(seq\_along(1:length(unique(df2$Species))), function(x)  
 inla(as.formula(formula()), data = lst2[[x]], family = distribution(), control.family = list(link = "log"),  
 control.compute = list(dic = TRUE, cpo = TRUE)))  
 results <- lapply(seq\_along(1:length(unique(df2$Species))), function(x) model[[x]]$summary.fixed[,c(1:3,5)])  
   
 }  
 return(results)  
})  
   
# Summary output of SDM  
   
fitsum <- eventReactive(input$summary, {fitsummary()})  
output$summary <- renderPrint({return(fitsum())})  
  
url <- a("Definition", href="https://rdrr.io/github/andrewzm/INLA/man/inla.mesh.2d.html")  
output$tab <- renderUI({tagList("URL link:", url)})  
   
}  
  
shinyApp(ui, server)