Shiny App for Single species joint models for geospatial annual data

Udani Wijewardhana

```
library(shiny)
library(DT)
library(plyr)
library(dplyr)
library(lattice)
library(gridExtra)
library(stringi)
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 Single species joint models for geospatial annual data
##############################################################################

# 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. Year – Annual/Seasonal/Daily
#         2. Locality - Location
#         3. Latitude
#         4. Longitude
#         5. count - Dependent variable for count model
#         6. zero - Dependent variable for count model
#         with or without predictor variables (numeric).
#         The above names are case sensitive."),
# A sample format of the data can be found in
# https://github.com/uwijewardhana/UDMSD.

### Shiny user interface ###

ui <- fluidPage(

titlePanel(h3("UDMSD - Single species joint models for geospatial annual
data", titleWidth = 350)),
hr(),

div(style="display: inline-block;vertical-align:top; width: 200px;",
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: 200px;",
selectInput("prednorm", "Predictors normalization:", choices=c("rnorm",
"stand", "none"), selected = "none")),

tabsetPanel(

tabPanel("Auunual Data",
         fluidRow(style = "margin-top: 25px;",
                  column(7, p(tags$b('Annual Count Data', style = "font-size:
150%; font-family:Helvetica; color:#4c4c4c; text-align:left;"))),
                  column(5, p(tags$b('Annual Predictor Data', style = "font-
size: 150%; font-family:Helvetica; color:#4c4c4c; text-align:left;")))),
         fluidRow(column(7, DT::dataTableOutput("contents")),
                  column(5, verbatimTextOutput("predictors")))
),

tabPanel("Single-species Joint Model",
         sidebarLayout(
         sidebarPanel(div(style='height:1100px; overflow: scroll',
                sliderInput(inputId = "offset", label = "offsets for automatic
boundaries", value = c(0.1, 0.3), min = 0.05, max = 2),
                sliderInput(inputId = "max.edge", label = "max.edge", value =
c(0.05, 0.5), min = 0.01, max = 2),
                sliderInput(inputId = "cutoff", label = "cutoff", value =
0.01, min = 0, max = 0.2),
                sliderInput(inputId = "min.angle", label = "min.angle", value
= c(21, 30), min = 1, max = 35),
                numericInput("convex", "convex for boundary", 0.5, min = NA,
max = NA, step = 0.0001),
                actionButton("mesh", "mesh"),
                selectInput("distribution", "Distribution of count model:",
                        choices=c("Negative Binomial", "Zeroinflated
Negative Binomial", "Negative Binomial Hurdle"), selected = "Negative
Binomial Hurdle"),
                selectInput(inputId = "pred", label = "Regresstion with
predictors?", choices=c("yes", "no"), selected = "yes"),
                uiOutput("independent"),
                h5('Generate Interaction Variables Here (if applicable)'),
                uiOutput("makeInteract1"), uiOutput("makeInteract2"),
                uiOutput("uiAdded"), actionButton("actionBtnAdd", "Create
Interaction Term"),
                selectInput("tempeffect", "temporal random effect model:",
choices=c("'ar1'", "'iid'", "'rw1'", "'rw2'"), selected = "'ar1'"),
                helpText("Posterior plots are only applicable for spatial or
```

```r
spatio-temporal models."),
               selectInput("speffect", "spatial random effect model:",
choices=c("spde", "'iid'"), selected = "'iid'"),
               actionButton("summary", "Summary")
               )),

mainPanel(fluidRow(column(4, tags$h3("Mesh:"), uiOutput("tab")),
                   column(8, tags$h3("Posterior Plots:"))),
          fluidRow(column(4, div(style='height:360px', plotOutput("mesh",
"mesh", width = "80%", height = "360px"))),
                   column(4, div(style='height:360px',
plotOutput("posteriormPlot"))),
                   column(4, div(style='height:360px',
plotOutput("posteriorsdPlot")))),
          fluidRow(column(12, tags$h3("Summary results of species joint
model:"), div(style='height:590px; overflow: scroll',
verbatimTextOutput("summary")))))
))))

### Shiny server ###

server <- function(input, output) {

# 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)
    return(x)
})

# Extract numeric predictor variables

filedata2 <- reactive({
    req(input$file)
    x <- filedata1()

    y = dplyr::select_if(x, is.numeric)
    if(ncol(y)>6){
    p = subset(y, select = -c(count, zero, Latitude, Longitude))
    p <- unique(p)
    p = subset(p, select = -c(Year))
    }else {p = NULL}

    if(!is.null(p)){
```

```r
    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$predictors <- renderPrint({
    req(input$file)
    df <- filedata2()
    if (is.null(df)){return(NULL)
    }else {
    return(summary(df))}
})

# Create INLA mesh

mesh <- reactive({
    m <- filedata1()
    coords <- cbind(m$Latitude, m$Longitude)
    bnd = inla.nonconvex.hull(coords, convex = input$convex)

    out <- INLA::inla.mesh.2d(
    loc = coords,
    boundary = bnd,
    max.edge = input$max.edge,
    min.angle = rev(input$min.angle),
    cutoff = input$cutoff,
    offset = input$offset)
    return(out)
})

mm <- eventReactive(input$mesh, {
    m <- mesh()
    df <- filedata1()
    coords <- cbind(df$Latitude, df$Longitude)
    plot(m)
    points(coords, col = "red")
})

# Output of mesh

output$mesh <- renderPlot({return(mm())})

# Create numeric predictor variables for joint model

filedata3 <- reactive({
    if (is.null(input$file)){return(NULL)}
    df <- filedata1()
    pred <- filedata2()
    pred <- pred[rep(seq_len(nrow(pred)), length(unique(df$Locality))), ]

    pred.z = matrix(NA, ncol = 10, nrow = nrow(pred)*2)
    for(i in 1:ncol(pred)){
    for(j in 1:nrow(pred)){pred.z[j,i] = pred[j,i]}}
    pred.z = data.frame(pred.z)
    colnames(pred.z) = gsub(" ", "", paste("p", 1:10, ".z"))

    pred.y = matrix(NA, ncol = 10, nrow = nrow(pred)*2)
    for(i in 1:ncol(pred)){
    for(j in 1:nrow(pred)){pred.y[j+nrow(pred),i] = pred[j,i]}}
    pred.y = data.frame(pred.y)
    colnames(pred.y) = gsub(" ", "", paste("p", 1:10, ".y"))

    return(cbind(pred.z, pred.y))
})

# Create dependent variable vectors for joint model
filedata4 <- reactive({

    if (is.null(filedata1())){return(NULL)}
    df <- filedata1()
    z = df$zero
    y = df$count

    if(input$distribution == "Negative Binomial Hurdle"){
    y[y == 0] <- NA
    }else {y = df$count}
    Y = cbind(z=c(z, rep(NA, length(z))), y=c(rep(NA, length(y)), y))
    return(Y)
})

# Rendering the list to the ui
output$uiAdded <- renderUI({checkboxGroupInput('added', 'List of
combinations', choices = names(interacts))})
```

```r
# 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
})})

# Checkbox list of all numeric variables to use
independent <- reactive({
    if(!is.null(input$file)){
    df <- filedata1()
    if(!is.null(filedata2())){
    return(names(df[ , !(names(df) %in% c("count", "zero", "Latitude",
"Longitude", "Locality"))]))}}
})

output$independent <- renderUI({checkboxGroupInput("independent",
"Independent (Predictor) Variables:", independent())})

# Variables to Add to the List of Combinations
makeInteract <- reactive({
    if(!is.null(input$file)){
    df <- filedata1()
    if(!is.null(filedata2())){
    return(names(df[ , !(names(df) %in% c("count", "zero", "Latitude",
"Longitude", "Locality"))]))}}
})

output$makeInteract1 <- renderUI({selectInput("makeInteract1", "Variable1 For
Interaction:", makeInteract())})
output$makeInteract2 <- renderUI({selectInput("makeInteract2", "Variable2 For
Interaction:", makeInteract())})

# distribution
distribution <- reactive({
  if(input$distribution == "Negative Binomial"){distribution = "nbinomial"
  } else if(input$distribution == "Zeroinflated Negative Binomial")
{distribution = "zeroinflatednbinomial1"
  } else {distribution = "zeroinflatednbinomial0"}
  return(distribution)
})
```

```r
# Create joint zero inflation model
fitsummary <- reactive({
  req(input$file)

  df1 <- filedata1()
  df2 <- filedata3()
  Y <- filedata4()

  coords <- cbind(df1$Latitude, df1$Longitude)
  spde = INLA::inla.spde2.matern(mesh(), constr = TRUE)
  coords$hhid = mesh()$idx$loc

  mu.z = c(rep(1,nrow(df1)), rep(NA, nrow(df1))); mu.y = c(rep(NA,
nrow(df1)), rep(1,nrow(df1)))
  effect = rep(df1$Year, 2); S = rep(coords$hhid, 2)

  if (input$pred == "no"){

  data = list(S = S, Y = Y, mu.z = mu.z, mu.y = mu.y, effect = effect)

  formula = paste(paste("Y ~ 0 + mu.z + mu.y +"),
            paste("+ f(effect, model = ", input$tempeffect,")"),
            paste("+ f(S, model = ", input$speffect,")"))
  } else {

  Year.z = c(df1$Year, rep(NA, nrow(df1))); Year.y = c(rep(NA, nrow(df1)),
df1$Year)
  p1.z = df2$p1.z; p2.z = df2$p2.z; p3.z = df2$p3.z; p4.z = df2$p4.z; p5.z =
df2$p5.z
  p6.z = df2$p6.z; p7.z = df2$p7.z; p8.z = df2$p8.z; p9.z = df2$p9.z; p10.z =
df2$p10.z
  p1.y = df2$p1.y; p2.y = df2$p2.y; p3.y = df2$p3.y; p4.y = df2$p4.y; p5.y =
df2$p5.y
  p6.y = df2$p6.y; p7.y = df2$p7.y; p8.y = df2$p8.y; p9.y = df2$p9.y; p10.y =
df2$p10.y

  data = list(S = S, Y = Y, mu.z = mu.z, mu.y = mu.y, effect = effect,
              Year.z = Year.z, Year.y = Year.y,
              p1.z = p1.z, p2.z = p2.z, p3.z = p3.z, p4.z = p4.z,
              p5.z = p5.z, p6.z = p6.z, p7.z = p7.z, p8.z = p8.z,
              p9.z = p9.z, p10.z = p10.z, p1.y = p1.y, p2.y = p2.y,
              p3.y = p3.y, p4.y = p4.y, p5.y = p5.y,
              p6.y = p6.y, p7.y = p7.y, p8.y = p8.y, p9.y = p9.y,
              p10.y = p10.y)

  if(!is.null(input$added)){

  y = as.list(input$added)
  p = lapply(seq_along(y), function(x) gsub("*", " ", y[[x]], fixed = TRUE))
  p = lapply(seq_along(y), function(x) strsplit(p[[x]], " "))
  zz <- lapply(seq_along(y), function(x) gsub(" ", "",  paste(p[[x]][[1]],
".z")))
  zz = lapply(seq_along(y), function(x) stri_c(zz[[x]], collapse = "*"))
  yy <- lapply(seq_along(y), function(x) gsub(" ", "",  paste(p[[x]][[1]],
".y")))
  yy = lapply(seq_along(y), function(x) stri_c(yy[[x]], collapse = "*"))

  t = as.list(input$independent)
  pp <- lapply(seq_along(t), function(x) gsub(" ", "",  paste(t[[x]], ".z")))
  qq <- lapply(seq_along(t), function(x) gsub(" ", "",  paste(t[[x]], ".y")))

  formula = paste(paste("Y ~ 0 + mu.z + mu.y +"), paste(pp,collapse="+"),
            paste("+", qq, collapse = "+"),
            paste("+", zz, collapse = "+"), paste("+", yy, collapse = "+"),
            paste("+ f(effect, model = ", input$tempeffect,")"),
            paste("+ f(S, model = ", input$speffect,")"))
  }else {

  t = as.list(input$independent)
  pp <- lapply(seq_along(t), function(x) gsub(" ", "",  paste(t[[x]], ".z")))
  qq <- lapply(seq_along(t), function(x) gsub(" ", "",  paste(t[[x]], ".y")))

  formula = paste(paste("Y ~ 0 + mu.z + mu.y +"), paste(pp,collapse="+"),
            paste("+", qq, collapse = "+"),
            paste("+ f(effect, model = ", input$tempeffect,")"),
            paste("+ f(S, model = ", input$speffect,")"))
  }}
  model <- inla(as.formula(formula), data = data, family = c("binomial",
distribution()),
                control.family = list(list(link = "logit"), list(link =
"log")),
                control.compute = list(dic = TRUE, cpo = TRUE))
  return(model)
})

proj <- reactive({
    if (is.null(fitsummary())){(NULL)}
    proj <- INLA::inla.mesh.projector(mesh(), projection = "longlat", dims =
c(150, 100))
    return(proj)
})

# Summary output of joint zero inflation model

fitsum <- eventReactive(input$summary, {summary(fitsummary())})
output$summary <- renderPrint({return(fitsum())})
```

```r
# Create posterior mean plot

pmPlot <- reactive({
    if(input$speffect == "'iid'"){return(NULL)
    } else {
    mp <- levelplot(row.values=proj()$x, column.values=proj()$y,
                    inla.mesh.project(proj(),
fitsummary()$summary.random$S$mean),
                    xla='Latitude', yla='Longitude',
                    main='posterior mean plot', contour=TRUE,
                    xlim=range(proj()$x), ylim=range(proj()$y))
    return(mp)}
})

# Create posterior standard deviation plot

psdPlot <- reactive({
    if(input$speffect == "'iid'"){return(NULL)
    } else {
    sdp <- levelplot(row.values=proj()$x, column.values=proj()$y,
                     inla.mesh.project(proj(),
fitsummary()$summary.random$S$sd),
                     xla='Latitude', yla='Longitude',
                     main='posterior standard deviation plot', contour=TRUE,
                     xlim=range(proj()$x), ylim=range(proj()$y))
    return(sdp)}
})

# Output of posterior standard deviation plot
output$posteriormPlot <- renderPlot({pmPlot()})

# Output of posterior standard deviation plot
output$posteriorsdPlot <- renderPlot({psdPlot()})

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)
```