UDMTA - A shiny App for Species Annual Temporal Abundance Models

Udani Wijewardhana

```
# Loading libraries
library(shiny)
library(DT)

## Attaching package: 'DT'
## The following objects are masked from 'package:shiny':
##     dataTableOutput, renderDataTable

library(plyr); library(dplyr)
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##     filter, lag

## The following objects are masked from 'package:base':
##     intersect, setdiff, setequal, union

library(tidyr); library(tidyverse)

## -- Attaching packages --------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()   masks plyr::arrange()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::mutate()    masks plyr::mutate()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()

library(INLA)
```

```
## Loading required package: Matrix
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##     expand, pack, unpack

## Loading required package: foreach
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##     accumulate, when

## Loading required package: parallel

## Loading required package: sp

## This is INLA_21.02.23 built 2021-09-11 13:14:12 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
##  - Save 379.7Mb of storage running 'inla.prune()'

################################################################################
########### Shiny App for Species Annual 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:
#         Year - Detected Year
#         Count - Species count
#         with or without predictor variables (numeric/factor).
# The above names are case sensitive.
# User can include any number of factor or numeric variables.
# Predictor variables should include as character or factor variables.
# A sample format of the data can be found in https://github.com/uwijewardhan
a/UDMTA.
# Data should be ordered according to factor levels and Year as in sample "Da
ta.csv".

### 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;", fileInpu
t("file", "Choose data CSV File", multiple = FALSE, accept = c("text/csv", "t
ext/comma-separated-values,text/plain", ".csv"))),
div(style="display: inline-block;vertical-align:top; width: 300px;", selectIn
put("prednorm", "Numeric predictors normalization:", choices=c("rnorm", "stan
```

```
d", "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-si
ze: 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", "N
egative 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:", choi
ces=c("No", "Yes"), selected = "No"),
         h5('Generate Interaction Variables Here (if applicable)'),
         uiOutput("independent"),
         uiOutput("makeInteract1"), uiOutput("makeInteract2"),
         uiOutput("uiAdded"), actionButton("actionBtnAdd", "Create Interactio
n Term"),
         hr(),

         actionButton("summary", "Summary"),
         actionButton("predict", "Predict"))),

mainPanel(fluidRow(column(12, p(tags$b('Summary results of species distributi
on model:', style = "font-size: 150%; font-family:Helvetica; color:#4c4c4c; t
ext-align:left;")))),
          fluidRow(column(12, verbatimTextOutput("summary"))),
          fluidRow(column(12, verbatimTextOutput("predict"))))

))))

### 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)
    return(x)
})

# 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
})})

# 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"))]
    return(names(df))
  }
})

output$independent <- renderUI({checkboxGroupInput("independent", "Independen
t (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"))]
    return(names(df))
  }
})

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

# Create dataframe for regression with predictor variables
Final <- reactive({

    x <- filedata1()
    if (is.null(x)){return(NULL)}
    p <- filedata2()
```

```
    if(input$distribution == "Poisson Hurdle" | input$distribution == "Negati
ve Binomial Hurdle"){
      x$Count[x$Count == 0] <- NA
    } else {
      x$Count = x$Count
    }

    fac = x %>% select_if(negate(is.numeric))
    for(i in 1:ncol(fac)){fac[,i] = as.factor(fac[,i])}
    y = dplyr::select_if(x, is.numeric)
    x <- cbind(y,fac)

    n = length(c(input$independent))

    if(n>1){
      M = data.frame(x[ , (names(x) %in% c(input$independent))])
    }else if(n == 0){
      M = NULL
    }else{
      M = data.frame(x[ , (names(x) %in% c(input$independent))])
      names(M)[1] <- c(input$independent)[1]
    }

    if(input$factor == "No"){

      Count <- aggregate(Count ~ Year, x, FUN = sum)
      Final = cbind(unique(cbind(M, t = x$Year)), Count = Count$Count, effect
= unique(x$Year))
      Final = subset(Final, select = -c(t))

    }else {

      z = dplyr::select_if(M, is.factor)
      z = cbind(z, Year = x$Year)
      z = unite(z, ID, sep = "-", remove = FALSE, na.rm = FALSE)
      Final = unique(z)

      z$Count = x$Count
      zz <- aggregate(Count ~ ID, z, FUN = sum)
      Final <- left_join(Final, zz, by = "ID")
      Final$effect = Final$Year

      c = dplyr::select_if(M, is.numeric)
      if("Year" %in% colnames(c)){c = subset(c, select = -c(Year))}
      c$ID = z$ID
      c = unique(c)

      Final = merge(x = Final, y = c, by = "ID", all.x = TRUE)
```

```
      Final = subset(Final, select = -c(ID))
    }
    return(Final)
})

# distribution
distribution <- reactive({
  if(input$distribution == "Poisson"){distribution = "poisson"
  } else if(input$distribution == "Negative Binomial"){distribution = "nbinom
ial"
  } else if(input$distribution == "Zeroinflated Poisson") {distribution = "ze
roinflatedpoisson1"
  } else if(input$distribution == "Zeroinflated Negative Binomial") {distribu
tion = "zeroinflatednbinomial1"
  } else if(input$distribution == "Poisson Hurdle") {distribution = "zeroinfl
atedpoisson0"
  } 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({

    df <- as.data.frame(Final())

    if(input$factor == "Yes"){

    model <- inla(as.formula(formula()), data = df, family = distribution(),
              control.family = list(link = "log"),
              control.compute = list(dic = TRUE, cpo = TRUE, config = TRUE), v
erbose = T)

    }else {

    model <- inla(as.formula(formula()), data = df, family = distribution(),
              control.family = list(link = "log"),
```

```
            control.compute = list(dic = TRUE, cpo = TRUE), verbose = T)
    }
    return(model)
})

# Summary output of SDM

fitres <- reactive({round(fitsummary()$summary.fixed[c(1:3,5)], digits = 3)}

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

# Predictions of SDM
fitpredict <- reactive({round(fitsummary()$summary.linear.predictor[c(1:3,5)]
, digits = 3)})

fitpred <- eventReactive(input$predict, {fitpredict()})
output$predict <- renderPrint({return(fitpred())})


}

shinyApp(ui, server)
```