UDMTA: Shiny Web Application for Single Species Temporal Abundance Models

Species distribution models can be applied to identify the extinction risks of threatened species which in turn helps to begin conservation planning activities to protect these species before they are lost. Obtaining accurate estimates of trends and other population changes in species observation data is difficult unless some baseline has previously been set, especially when considering threatened species. Spatial or temporal scales can confound inference about changes in species observation data when used to draw conclusions about potential impacts on a different scale. It is important to consider temporal resolution when applying species distribution models. It is important to understand the factors that influence species distributions within the application of environmental niche models (Fryxel et al., 2014). Studies of temporal fluctuation in richness within a local community can help explain geographic patterns of richness. While richness dynamics have long been a subject of palaeontology (Sepkoski 1978;), ecological time scale richness patterns have gotten far less attention than their spatial scale (White 2007).

The application consists of 2 pages with main window:

1) Main window allows the user to upload the input files (data file) and also gives the option to normalize or standardize the counts or predictors data.

This page give the output table of data the user inputs to the app. Used should input a .CSV file with the columns "Species", "Year", "Count" and the predictors if available (numeric/categorical). The first three column names "Species", "Year" and "Count" are case sensitive. The summary of numeric predictors also displayed as a table output in this page. User has the ability to normalize the predictors here.

2) Second tab allow users to fit species temporal models using R-INLA**.**

Structure of UDMTA App

UDMTA is a Shiny web application that allows to fit single species or multispecies temporal models and estimate significant factors and the trend. It is addressed to ecologists interested in analysing species abundance/occurrence data but lacking the appropriate theoretical knowledge to use this statistical software is required. This app developed based on R-INLA (Rue et al. 2018) which provides a number of options to model data collected in space and time. Information about all the packages used are shown in Table 1.

Table 1. Softwares and R packages used for developing PPMCA

Name		Description
dplyr	(Wickham and Francois, 2016)	A fast, consistent tool for working with data frame like objects, both in memory and out of memory.
htmlwidgets	(Vaidyanathan *et al.,* 2016)	Provides a framework for easily creating R bindings to JavaScript libraries.
R-INLA	(Lindgren and Rue, 2015)	Performs full Bayesian analysis on generalised additive mixed models using Integrated Nested Laplace Approximations.
Shiny	(Chang *et al.,* 2016)	Web Application Framework for R.
DT	(Xie, 2016)	Create data tables.

plyr	(Wickham, 2020)	A set of tools that solves a common set of problems: you need to break a big problem down into manageable pieces, operate on each piece and then put all the pieces back together. For example, you might want to fit a model to each spatial location or time point in your study, summarise data by panels or collapse high-dimensional arrays to simpler summary statistics.

The number of recorded species in large numbers of geographic areas are now increasing because of citizen science data. However, citizen science count data often includes excess zeros for bird species especially for rare bird species in specific areas or time periods which means dataset is characterized by an irregular distribution for counts such as; few observations of large flocks, occasional observations of small flocks or single birds, and many zeros. In that reason overdispersion is seemed to be a main problem. Negative Binomial distribution is developed as an alternative to handle the overdispersion. Zero Inflated models and Hurdle models are developed to handle this type of zero-inflated count data. UDMTA allows users to fit Poisson or Negative Binomial distribution which are the most common species distributions with the Hurdle models or Zero Inflated models which are the most common zero inflated data models. We used `selectInput()` to add these choices and call the distribution while fitting the model.

```
# Input the distribution in ui

selectInput("distribution", "Distribution:", choices=c("Poisson", "Negative
Binomial", "Zeroinflated Poisson", "Zeroinflated Negative Binomial",
"Poisson Hurdle", "Negative Binomial Hurdle"), selected = "Poisson")

# Assign the relevant term in server{}

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"}
```

The analysis of temporal data over a continuous domain will be done with smoothing methods described such as '*iid*','*ar*1','*rw*1' or '*rw*2' to take the short term fluctuations into account. We used `selectInput()` to add these choices and call the temporal effect inside the formula.

```
selectInput("tempeffect", "temporal random effect model:", choices=c("'ar1'",
"'iid'", "'rw1'", "'rw2'"), selected = "'ar1'")
```

When an independent variable's effect on a dependent variable change based on the value(s) of one or more other independent variables, this is known as an interaction effect. To know if the rate at which abundance is changing over time differs according to a relevant predictor variable, we have included the facility to add interaction terms between any two predictor variables in our regression models. Variables can be numeric or categorical, user can add any number of interaction terms (variable combinations) to fit the temporal model. The interaction with a categorical variable tells us what the difference in slope is and whether this difference is significant. The interaction term list has created as below.

```
# Create a list of dependent variables in ui
```

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

# Variables to add to the List of interaction combinations
makeInteract <- reactive({
    if(!is.null(input$file)){
    df = all()[ , !(names(all()) %in% c("Count"))]
    return(names(df))}
})

# Select variable 1 for interaction
output$makeInteract1 <- renderUI({selectInput("makeInteract1", "Variable1 For
Interaction:", makeInteract())})

# Select variable 2 for interaction
output$makeInteract2 <- renderUI({selectInput("makeInteract2", "Variable2 For
Interaction:", makeInteract())})
# Rendering the list to the ui

# Create interaction name list
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
})})

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

The concept of a "group" is provided by INLA, which allows for more complex dependency structures. There is also can have a group correlation between groups. To take this effect into account, we allow users to fit multispecies models using most common group effects "*exchangeable*" or *"iid"* effect.

Set up and installation

To build this Shiny app, we need to clone the GitHub repository from UDMTA and save it in our computer. This folder contains a sample Data.CSV file, the vignette and `app.R` file. Then, we can launch the app by clicking the Run App button at the top of the RStudio editor or by executing `runApp("appdir_path")`where `appdir_path` is the path of the directory that contains the `app.R` file. For this we need to install R and RStudio in our computer. User can download and install package R-INLA by R-INLA. The complete code of the Shiny app is given below, and a snapshot of the Shiny app created is shown in Figure 1.

```
### Shiny User Interface ###

ui <- fluidPage(

titlePanel(strong("UDM - 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")),

# Tabset
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("type", "Model type (Multi-species or Single species",
         choices=c("Multi Species", "Single Species"), selected = "Multi
Species"),
         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("controlgroup", "control group model (multispecies):",
choices=c("'iid'", "'exchangeable'"), selected = "'exchangeable'"),

         h5('Generate Interaction Variables Here (if applicable)'),
         uiOutput("independent"),
         uiOutput("makeInteract1"), uiOutput("makeInteract2"),
         uiOutput("uiAdded"), actionButton("actionBtnAdd", "Create Interaction
Term"),
         hr(),
         actionButton("summary", "Summary"))),

# main panel
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"))

    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 predictor 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
})})

all <- reactive({
  inFile <- input$file
  if (is.null(inFile)){return(NULL)}
  x <- as.data.frame(read.csv(inFile$datapath, fileEncoding="UTF-8-BOM"))

  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]}}}

  xx = cbind(Year = unique(x$Year), p)

  if(is.null(p)){
    Final = x
  }else {
    z = dplyr::select_if(x, is.factor)
    q <- x[ , (names(x) %in% c("Species", "Year", "Count"))]
    n = nrow(x)/length(unique(x$Year))
    p <- xx[rep(seq_len(nrow(xx)), n), ]

    Final = cbind(q, subset(p, select = -c(Year)), subset(z, select = -
c(Species)))
  }
  return(Final)
})

# Checkbox list of all numeric variables to use
independent <- reactive({
    if(!is.null(input$file)){
    df = all()[ , !(names(all()) %in% c("Count"))]
    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)){
    df = all()[ , !(names(all()) %in% c("Count"))]
```

```
    return(names(df))}
})

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

# Fit SDM using R-INLA

fitsummary <- reactive({

# multispicies case
    if(input$type == "Multi Species"){

    if(!is.null(input$independent)){
    df <- unique(all()[ , (names(all()) %in% c(as.vector(paste(input$independent,
sep = ",")), "Species", "Count", "Year"))])
    }

# "Count" for the specific distribution
    count = df$Count

    if(input$distribution == "Poisson Hurdle" | input$distribution == "Negative
Binomial Hurdle"){
    count[count == 0] <- NA
    } else {count = count}

    df$Count <- count
    df$effect <- df$Year
    df$group <- rep(c(1:length(unique(df$Species))), each =
nrow(df)/length(unique(df$Species)))

# distribution
    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"}

    if(!is.null(input$added)){
    formula = as.formula(paste("Count ~ 1 +", paste(input$independent, collapse
= "+"),
    paste("+", paste(input$added, collapse = "+")),
    paste("+", "f(effect, model = ", input$tempeffect, ", group = group",
    ", control.group = list(model = ", input$controlgroup,"))")))
    }else{
    formula = as.formula(paste("Count ~ 1 +", paste(input$independent, collapse
= "+"),
    paste("+", "f(effect, model = ", input$tempeffect, ", group = group",
    ", control.group = list(model = ", input$controlgroup,"))")))
    }

    model <- INLA::inla(formula, data = df, family = distribution,
```

```
                        control.family = list(link = "log"),
                        control.compute = list(dic = TRUE, cpo = TRUE, po =
                        TRUE))
    return(model$summary.fixed[,c(1:3,5)])

# single species case
    }else {

    if(!is.null(input$independent)){
    df <- unique(all()[ , (names(all()) %in%
c(as.vector(paste(input$independent, sep = ",")), "Species", "Count",
"Year"))])
    }

# "Count" for the specific distribution
    count = df$Count

    if(input$distribution == "Poisson Hurdle" | input$distribution == "Negative
Binomial Hurdle"){
    count[count == 0] <- NA
    } else {count = count}

    df$Count <- count
    df$effect <- df$Year

# distribution
    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"}

    lst <- split(df, df$Species)

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

    formula = as.formula(paste("Count ~ 1 +", paste(input$independent, collapse
= "+"),
    paste("+", paste(input$added, collapse = "+")),
    paste("+", "f(effect, model = ", input$tempeffect, ")")))
    }else {
    formula = as.formula(paste("Count ~ 1 + ", paste(input$independent,
collapse = "+"),
    paste("+", "f(effect, model = ", input$tempeffect, ")")))
    }

    model <- list()
    results <- list()

    for (i in 1:length(unique(df$Species))){

    model[[i]] <- INLA::inla(formula, data = lst[[i]], family = distribution,
                  control.family = list(link = "log"),
                  control.compute = list(dic = TRUE,cpo = TRUE, po = TRUE))
```

```
    }
    return(lapply(seq_along(1:unique(length(df$Species))), function(x)
model[[x]]$summary.fixed[,c(1:3,5)]))
    results[[i]] <- model[[i]]$summary.fixed[,c(1:3,5)]
    }
    return(results)
    }
})

# Summary output of SDM

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

# INLA mesh definition
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)
```

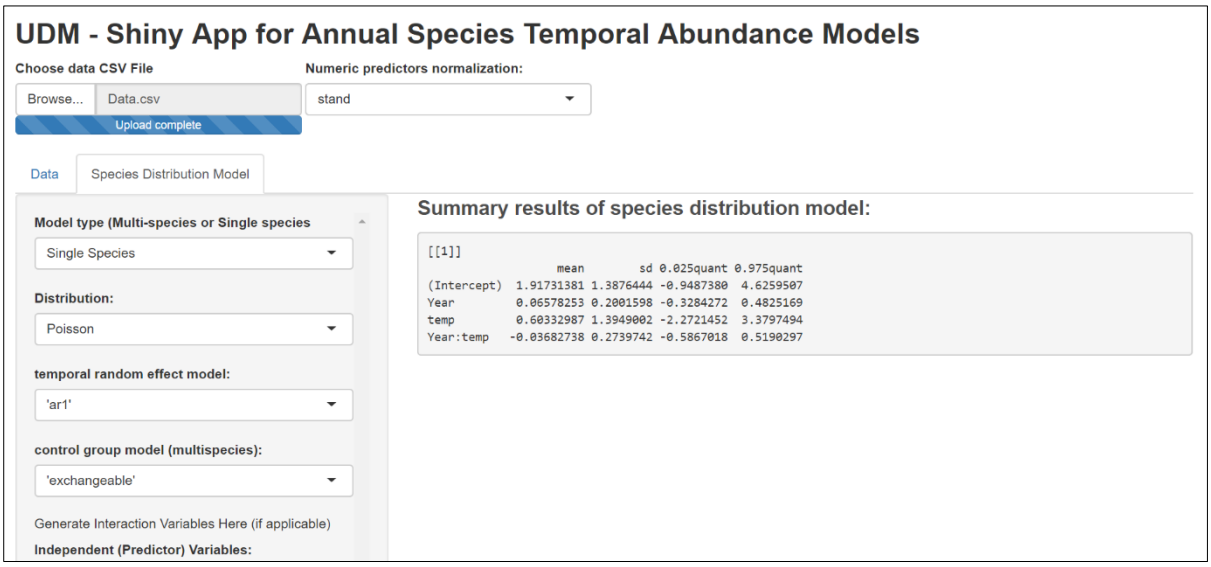

Figure 1 Snapshot of the UDMCA App

References

Sepkoski J., Jr1978A kinetic model of Phanerozoic taxonomic diversity I. Analysis of marine orders. Paleobiology 4, 223–251.

White E. P.2007Spatiotemporal scaling of species richness: patterns, processes and implications. In Scaling biodiversity (eds Storch D., Marquet P. A., Brown J. H.), pp. 325–346 Cambridge, UK: Cambridge University Press.

Fryxell, J, Sinclair A. & A. Cughley, G. (2014) Wildlife ecology, conservation, and management, 3rd edn. Wiley-Blackwell.

Rue, Havard, Finn Lindgren, Daniel Simpson, Sara Martino, Elias Teixeira Krainski, Haakon Bakka, Andrea Riebler, and Geir-Arne Fuglstad. 2018. *INLA: Full Bayesian Analysis of Latent Gaussian Models Using Integrated Nested Laplace Approximations*.