UDMCA – A Shiny App for Single Species Univariate Changepoint Analysis

Udani A. Wijewardhana * uwijewardhana@swin.edu.au

UDMCA is a Shiny web application that allows to visualize changepoints by using Bayesian changepoint techniques implemented in 'changepoint', 'breakpoint', 'cumSeg' and 'bcp'. To carry out these analyses users simply need to click the buttons that create the input files required, execute the software and process the output to generate tables of values and plots with the results. User can use four Bayesian changepoint methods to identify the significant changes of the abundance level using raw data (abundance/occurrence). Still these techniques allow to detect changepoints for a single location (univariate scenario). Therefore, the data entered by users are considered as a single location data when analyse changepoints.

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 "Year", "Count" and the numeric predictors for bcp method (optional). The first two column names "Year" and "Count" are case sensitive.

2) Changepoint analysis page that user can choose a Bayesian changepoint method to carry out the changepoint analysis.

This page gives the option to fit Bayesian changepoints for species count data. This app has developed for univariate changepoint methods and it will consider the count data CSV as a single located dataset. It is easy to identify the changepoint locations if you use count data in ascending order with an ID (e.g. for monthly data starting from January 2002, assign 2 for February 2002). Four Bayesian changepoint packages (changepoint, breakpoint, cumSeg and bcp) could be used to find the significant changes. This page also visualizes the relevant changepoint profile plots. For bcp package user has the option to use predictor variables to analyse. As the algorithms stated in the page, user can change the relevant parameters for each method and find the best scenario.

Changepoint Analysis - Statistical packages used

changepoint package: The changepoint package implements various mainstream and specialised changepoint methods for finding single and multiple changepoints within data which includes many popular non-parametric and frequentist methods (Killick, Haynes and Eckley, 2016).

breakpoint package: The breakpoint package implements variants of the Cross-Entropy (CE) method to estimate both the number and the corresponding locations of break-points in biological sequences of continuous and discrete measurements. The proposed method primarily built to detect multiple break-points in genomic sequences. However, it can be easily extended and applied to other problems (Priyadarshana and Sofronov, 2016).

cumSeg package: The cumSeg package (Muggeo, 2010) estimates the of number and location of change points in mean-shift (piecewise constant) models which is useful to model genomic sequences of continuous measurements. The algorithm first estimates the highest number of

change points using the efficient 'segmented' algorithm of Muggeo (2003) and then select some of them using a generalized BIC criterion by applying the lar's algorithm of Efron et al. (2004) (Muggeo, 2010).

bcp package: The bcp package provides an implementation of the Barry and Hartigan (1993) product partition model for the normal errors change point problem using Markov Chain Monte Carlo. It also extends the methodology to regression models on a connected graph (Wang and Emerson, 2015) and allows estimation of change point models with multivariate responses (Erdman and Emerson, 2007).

Structure of UDMCA App

Information about all the packages used are shown in Table 1.

Table 1. Softwares and R packages used for developing UDMCA

Package	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.
plyr	Wickham, 2011	plyr is an R package that makes it simple to split data apart, do stuff to it, and mash it back together. This is a common data-manipulation step. Importantly, plyr makes it easy to control the input and output data format from a syntactically consistent set of functions.
htmlwidgets	Vaidyanathan *et al.,* 2016	Provides a framework for easily creating R bindings to JavaScript libraries.
shiny	Chang *et al.,* 2016	Web Application Framework for R.
changepoint breakpoint bcp cumSeg	Killick, Haynes and Eckley, 2016 Priyadarshana and Sofronov, 2016 Erdman and Emerson, 2007 Muggeo, 2010	Bayesian changepoint analysis techniques.
DT	Xie, 2016	Create data tables.

We create a sidebar-style user interface. A title panel, a sidebar panel for inputs on the left, and a main panel for outputs on the right make up this layout. The `fluidPage()` function contains the user interface elements, which allows the programme to automatically resize to the size of the browser window. `titlePanel()` is used to add the title of the app. Then we create a sidebar layout with input and output definitions using `sidebarLayout()`. The variables `sidebarPanel()` and `mainPanel()` are sent to `sidebarLayout()`. The `sidebarPanel()` creates a left-hand sidebar panel for inputs. `mainPanel()` generates a main panel on the right for showing outputs. Here we have added texts with the description of the panels while separating the multiple elements in the same panel with commas.

```
ui <- fluidPage(
      titlePanel("title"),
      sidebarLayout(
      sidebarPanel("sidebar panel for inputs"),
      mainPanel("main panel for outputs")
  )
)
```

To import the data, we want to show in the app we have used the `read.csv()` function A sample data can be found in the repository called `Data.csv`. We write this code at the

beginning of `app.R` outside the `server()` function to read the data once which is not unnecessarily run more than once and the performance of the app is not decreased. We use DT package to show the data in using an interactive table. In `ui` we use `DTOutput()`, and in `server()` we use `renderDT()`.

The input values are used to read the CSV and shapefile files. This is accomplished through the use of a reactive expression. An R expression that uses an input value and returns a value is known as a reactive expression. The `reactive()` method, which takes a R expression surrounded by braces (`{}`), is used to produce a reactive expression. When the input value changes, the reactive expression updates. For instance, `read.csv(input$filedata$datapath)` reads the data, where `input$filedata$datapath` is the data path provided in the value of the input that uploads the data. Inside `reactive()`, we put `read.csv(input$filedata$datapath)`. The reactive expression is executed each time `input$filedata$datapath` is modified in this way. The reactive expression's output is assigned to data. Data can be obtained with `data()` in `server()`. Each time the reactive expression that builds is executed, `data()` will be changed.

```
data <- reactive({read.csv(input$filedata$datapath)}) # in server()
```

We start the reactive expression that reads the data with `req(input$filedata)`. If the data has not been uploaded the `input$filedata = ""`. This stops the execution of the reactive expression, then `data()` is not updated, and the output depending on `data()` is not executed.

HTML widgets are created with JavaScript libraries and embedded in Shiny by using the HTML widgets package (Vaidyanathan et al., 2016). These HTML widgets are included into the application by calling an output for the widget in the `ui` and assigning a render call to the output on the `server()`.

We have included the ability for the user to select a specific variable and year to be displayed. To make it easier to choose a variable, we have included a menu input with all of the options. The map and time plot will then be rebuilt when the user picks a specific variable. We need to place an input function `*Input()` in the `ui` object to add an input to a Shiny app. Several arguments are required for each input function. The first two are `inputId`, which is an id that is required to retrieve the input value, and label, which is the text displayed next to the input in the app. We used `selectInput()`function for predictor variables normalization option by selecting the appropriate names from boxes containing the possible choices. Similarly, changepoint analysis page includes `numericInput()` functions which allow to enter a single number or a range to fit the model. As shown below, we create the input with a menu that provides the variable options using `selectInput()`.

```
# in ui
selectInput(
  inputId = "variableselected",
  label = "Select variable",
  choices = c("cases", "population")
)
```

Set up and installation

To build this Shiny app, we need to clone the Zip file from UDMCA 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. The users who do not have R in their computer can use UDMCA to launch the Shiny app. A snapshot of the Shiny app created is shown in Figure 1.

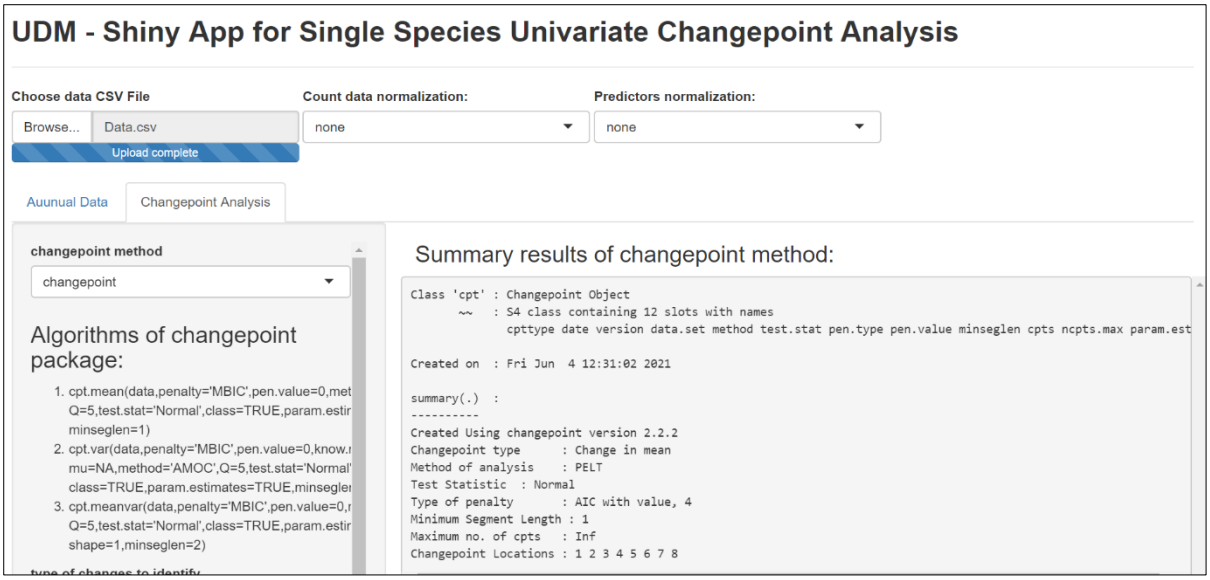

Figure 1 Snapshot of the UDMCA App

References

Shiny.rstudio.com. 2020. Shiny - Reactivity - An Overview. [online] Available at: https://shiny.rstudio.com/articles/reactivity-overview.html [Accessed 29 April 2020].

Killick R, Haynes K and Eckley IA (2016). changepoint: An R package for changepoint analysis. R package version 2.2.2, https://CRAN.R-project.org/package=changepoint.

Priyadarshana, W. J. R. M., and G. Sofronov. (2016). breakpoint: An R Package for Multiple Break-Point Detection via the Cross-Entropy Method. CRAN Repository (2016): 1-25.

Muggeo, V. M. R., and G. Adelfio. (2011). Efficient change pointdetection for genomic sequences of continuous measurements.Bioinformatics 27:161-166. http://dx.doi.org/10.1093/bioinformatics/btq647

Muggeo, V.M.R. (2003), Estimating regression models with unknown break-points. Statist. Med., 22: 3055-3071. doi:10.1002/sim.1545

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Leastangle regression.Annals of Statistics,32, 407– 489.

Barry,D. and Hartigan,J.A. (1993) A Bayesian analysis for change point problems.J. Am.Stat. Assoc.,88, 309– 319.

Chandra Erdman, John W. Emerson (2007), bcp: An R Package for Performing a Bayesian Analysis of Change Point Problems, Journal of Statistical Software, 23(3), 1-13, URL http://www.jstatsoft.org/v23/i03/.

Xiaofei Wang and John W. Emerson (2015), Bayesian Change Point Analysis of Linear Models on General Graphs, Working Paper.

Xie Y (2016). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.2, URL https://CRAN.R-project.org/package=DT.

Vaidyanathan R, Xie Y, Allaire J, Cheng J, Russell K (2016). htmlwidgets: HTML Widgets for R. R package version 0.7, URL https://CRAN.R-project.org/package=htmlwidgets.

R Core Team (2017). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org.

Wickham H, Francois R (2016). dplyr: A Grammar of Data Manipulation. R package version 0.5.0, URL https://CRAN.R-project.org/package=dplyr.

ChangW, Cheng J, Allaire J, Xie Y,McPherson J (2016). shiny: Web Application Framework for R. R package version 0.14.1, URL https://CRAN.R-project.org/package=shiny.

Xie Y (2016). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.2, URL https://CRAN.R-project.org/package=DT.