

# UDMCA – A Shiny App for Single Species Univariate Changepoint Analysis

Udani A. Wijewardhana \* [uwijewardhana@swin.edu.au](mailto: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 annual raw data. Since still developed univariate changepoint methods, app will consider the annual data CSV as a single located dataset. 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 <i>et al.</i> , 2016	Provides a framework for easily creating R bindings to JavaScript libraries.
shiny	Chang <i>et al.</i> , 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.

The complete code of the Shiny app is given below, and a snapshot of the Shiny app created is shown in Figure 1.

```
# Loading libraries
library(shiny)
library(DT)
library(plyr)
library(dplyr)
library(breakpoint)
library(cumSeg)
library(changepoint)
library(bcp)

### Shiny user interface ###

ui <- fluidPage(

# Add title of the app
titlePanel(strong("UDMCA - Shiny App for Single Species Univariate Changepoint
Analysis", style = "color:#3474A7")),

# Read the datafile in .CSV format
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"))),

# Option to normalize count data
div(style="display: inline-block;vertical-align:top; width: 300px;",
selectInput("datanorm", "Count data normalization:", choices=c("rnorm",
"stand", "none"), selected = "none")),

# Option to normalize predictors (for bcp method - optional)
div(style="display: inline-block;vertical-align:top; width: 300px;",
selectInput("prednorm", "Predictors normalization:", choices=c("rnorm",
"stand", "none"), selected = "none")),

# There are two tables; the data output tab and changepoint analysis tab

tabsetPanel(

tabPanel("Auunual Data", fluidRow(column(12, DT::dataTableOutput("contents")))),

tabPanel("Changepoint Analysis",
  sidebarLayout(
    sidebarPanel(div(style='height:900px; overflow: scroll',
      selectInput("changepoint", "changepoint method",
choices=c("changepoint", "breakpoint", "cumSeg", "bcp"), selected =
"changepoint"),

# changepoint method
conditionalPanel(
  condition = "input.changepoint == 'changepoint'",
  tags$h3("Algorithms of changepoint package:"),
  tags$ol(
```

```

tags$li("cpt.mean(data,penalty='MBIC',pen.value=0,method='AMOC',
Q=5,test.stat='Normal',class=TRUE,param.estimates=TRUE, minseglen=1)"),

tags$li("cpt.var(data,penalty='MBIC',pen.value=0,known.mean=FALSE,
mu=NA,method='AMOC',Q=5,test.stat='Normal',
class=TRUE,param.estimates=TRUE,minseglen=2)"),

tags$li("cpt.meanvar(data,penalty='MBIC',pen.value=0,method='AMOC',
Q=5,test.stat='Normal',class=TRUE,param.estimates=TRUE,
shape=1,minseglen=2)")),

selectInput("changes", "type of changes to identify", choices=c("mean",
"variance", "mean and variance"), selected = "mean"),
selectInput("penalty", "penalty", choices=c("BIC", "MBIC", "AIC"), selected =
"AIC"),
selectInput("method", "method", choices=c("AMOC", "PELT", "SegNeigh",
"BinSeg"), selected = "PELT"),
numericInput("Q", "maximum number of changepoints to search", 5, min = 0, max =
NA, step = 1),

helpText("The maximum number of changepoints to search for using the BinSeg
method. The maximum number of segments (number of changepoints + 1) to search
for using the SegNeigh method."),

selectInput("test.stat", "test statistic", choices=c("Normal", "Gamma",
"Exponential", "Poisson"), selected = "Normal"),

helpText("Gamma, Exponential and Poisson distribution only applicable for mean
and variance function."),

selectInput("class", "class", choices=c("TRUE", "FALSE"), selected = "TRUE"),
selectInput("param.estimates", "param.estimates", choices=c("TRUE", "FALSE"),
selected = "TRUE"),
numericInput("shape", "shape parameter for Gamma distribution", 1, min = -100,
max = NA, step = 0.001),
numericInput("minseglen", "minseglen", 1, min = 1, max = NA, step = 1),

helpText("Positive integer giving the minimum segment length (no. of
observations between changes), default is the minimum allowed by theory. For
cpt.var and cpt.meanvar, the minimum minseglen is 2 and for cpt.mean, the
minimum minseglen is 1.")),

# breakpoint method
conditionalPanel(
  condition = "input.changepoint == 'breakpoint'",
  tags$h3("Algorithms of breakpoint package:"),
  tags$ol(

tags$li("CE.NB(data, Nmax = 10, eps = 0.01, rho = 0.05, M = 200, h = 5, a =
0.8, b = 0.8, distyp = 1, penalty = 'BIC', parallel = FALSE)"),

tags$li("CE.NB.Init(data, init.locs, eps = 0.01, rho = 0.05, M = 200, h = 5, a
= 0.8, b = 0.8, distyp = 1, penalty = 'BIC', var.init = 1e+05, parallel =
FALSE)"),

tags$li("CE.Normal.Init.Mean(data, init.locs, eps = 0.01, rho = 0.05, M = 200,
h = 5, a = 0.8, b = 0.8, distyp = 1, penalty = 'BIC', var.init = 1e+05,
parallel = FALSE)"),

```

```

tags$li("CE.Normal.Init.MeanVar(data, init.locs, eps = 0.01, rho = 0.05, M =
200, h = 5, a = 0.8, b = 0.8, distyp = 1, penalty = 'BIC', var.init = 1e+05,
parallel = FALSE)"),

tags$li("CE.Normal.Mean(data, Nmax = 10, eps = 0.01, rho = 0.05, M = 200, h =
5, a = 0.8, b = 0.8, distyp = 1, penalty = 'BIC', parallel = FALSE)"),

tags$li("CE.Normal.MeanVar(data, Nmax = 10, eps = 0.01, rho = 0.05, M = 200, h
= 5, a = 0.8, b = 0.8, distyp = 1, penalty = 'BIC', parallel = FALSE)"),

tags$li("CE.ZINB(data, Nmax = 10, eps = 0.01, rho = 0.05, M = 200, h = 5, a =
0.8, b = 0.8, distyp = 1, penalty = 'BIC', parallel = FALSE)"),
tags$li("CE.ZINB.Init(data, init.locs, eps = 0.01, rho = 0.05, M = 200, h = 5,
a = 0.8, b = 0.8, distyp = 1, penalty = 'BIC', var.init = 1e+05, parallel =
FALSE)")),

selectInput("algorithm", "breakpoint function", choices=c("CE.NB",
"CE.NB.Init", "CE.Normal.Init.Mean", "CE.Normal.Init.MeanVar", "CE.ZINB",
"CE.ZINB.Init"), selected = "CE.NB"),
numericInput("Nmax", "maximum number of breakpoints to search", 10, min = 0,
max = NA, step = 1),
numericInput("eps", "the cut-off value for the stopping criterion in the CE
method", 0.01, min = 0, max = 100, step = 0.00001),
numericInput("rho", "the fraction which is used to obtain the best performing
set of sample solutions", 0.05, min = 0, max = NA, step = 0.00001),
numericInput("M", "sample size to be used in simulating the locations of break-
points", 200, min = 0, max = NA, step = 1),
numericInput("h", "minimum aberration width", 5, min = 0, max = NA, step = 1),
numericInput("a", "a smoothing parameter value", 0.8, min = 0, max = NA, step =
0.00001),

helpText("It is used in the four parameter beta distribution to smooth both
shape parameters. When simulating from the truncated normal distribution, this
value is used to smooth the estimates of the mean values."),

numericInput("b", "a smoothing parameter value", 0.8, min = 0, max = NA, step =
0.00001),

helpText("It is used in the truncated normal distribution to smooth the
estimates of the standard deviation."),

selectInput("distyp", "distribution to simulate break-point locations",
choices=c("1", "2"), selected = "1"),

helpText("Options: 1 = four parameter beta distribution, 2 = truncated normal
distribution"),

selectInput("penalty", "penalty", choices=c("BIC", "AIC"), selected = "BIC"),
numericInput("var.init", "Initial variance value to facilitate the search
process", 100000, min = 0, max = NA, step = 0.00001),
selectInput("parallel", "parallel", choices=c("TRUE", "FALSE"), selected =
"FALSE"),

textInput("init.locs", "Initial break-point locations - enter a vector (comma
delimited) - e.g. '0,1,2'", NULL)),

# cumSeg method
conditionalPanel(
  condition = "input.changepoint == 'cumSeg'",

```

```

tags$h3("Algorithms of cumSeg package:"),

tags$li("fit.control(toll = 0.001, it.max = 5, display = FALSE, last = TRUE,
maxit.glm = 25, h = 1, stop.if.error = FALSE)"),

tags$li("sel.control(display = FALSE, type = c('bic', 'mdl', 'rss'), S = 1, Cn
= log(log(n)), alg = c('stepwise', 'lasso'), edf.psi = TRUE)"),

tags$li("jumpoints(y, x, k = min(30, round(length(y)/10)), output = '2', psi =
NULL, round = TRUE, control = fit.control(), selection = sel.control())"),
      hr(),
numericInput("k", "k = the starting number of changepoints", 2, min = 0, max =
NA, step = 1),

helpText("It should be quite larger than the supposed number of (true)
changepoints. This argument is ignored if starting values of the changepoints
are specified via psi."),

selectInput("output", "output", choices=c(1,2,3), selected = 1),
      textInput("psi", "psi = starting values for the changepoints -
enter a vector (comma delimited) - e.g. '0,1,2'", NULL),

helpText("When psi=NULL (default), k quantiles are assumed."),

selectInput("round", "round", choices=c("TRUE", "FALSE"), selected = "TRUE"),
numericInput("toll", "toll = positive convergence tolerance", 0.001, min = 0,
max = NA, step = 0.0001),
numericInput("it.max", "it.max = integer giving the maximal number of
iterations", 5, min = 0, max = NA, step = 1),
selectInput("display", "display", choices=c("TRUE", "FALSE"), selected =
"FALSE"),
selectInput("last", "last", choices=c("TRUE", "FALSE"), selected = "TRUE"),
numericInput("maxit.glm", "maxit.glm", 25, min = 0, max = NA, step = 1),
numericInput("h", "h", 1, min = 0, max = NA, step = 1),
selectInput("stop.if.error", "stop.if.error", choices=c("TRUE", "FALSE"),
selected = "FALSE"),

helpText("logical indicating if the algorithm should stop when one or more
estimated changepoints do not assume admissible values."),

selectInput("type", "type", choices=c("bic", "mdl", "rss"), selected = "bic"),
      numericInput("S", "S", 1, min = 0, max = NA, step = 1),

helpText("If type = 'rss' the optimal model is selected when the residual sum
of squares decreases by the threshold S."),

numericInput("Cn", "Cn", 1, min = 0, max = NA, step = 0.0000001),

helpText("If type= 'bic' a character string (as a function of 'n') to specify
to generalized BIC. If Cn=1 the standard BIC is used."),

selectInput("alg", "alg", choices=c("stepwise", "lasso"), selected = "stepwise"),
selectInput("edf.psi", "edf.psi", choices=c("TRUE", "FALSE"), selected =
"TRUE")),

# bcp method
conditionalPanel(
  condition = "input.changepoint == 'bcp'",
  tags$h3("Algorithm of bcp package:"),

```



```

tags$li("bcp(y, x = NULL, id = NULL, adj = NULL, w0 = NULL, p0 = 0.2, d = 10,
burnin = 50, mcmc = 500, return.mcmc = FALSE, boundaryType = 'node', p1 = 1,
freqAPP = 20, nreg = -1)"),

selectInput("pred", "predictors", choices=c("Yes", "No"), selected = "No"),

helpText("w0 is applicable when predictors are available."),

textInput("w0", "w0 - enter a vector (comma delimited) - e.g. '0.2,0.5,0.7'",
NULL),
numericInput("p0", "p0", 0.2, min = 0, max = 1, step = 0.01),

helpText("w0 and p0 are optional values which are between 0 and 1 for Barry and
Haritgan's hyperparameters; these default to the value 0.2, which has been
found to work well."),

numericInput("p1", "p1 = The proportion of Active Pixel Passes run that are the
actual Active Pixel Passes", 1, min = 0, max = 1, step = 0.01),
numericInput("d", "d", 10, min = 0, max = NA, step = 1),

helpText("a positive number only used for linear regression change point
models. Lower d means higher chance of fitting the full linear model (instead
of the intercept-only model)."),

numericInput("burnin", "burnin", 50, min = 0, max = NA, step = 1),
numericInput("mcmc", "mcmc", 500, min = 0, max = NA, step = 1),
selectInput("return.mcmc", "return.mcmc", choices=c("TRUE", "FALSE"), selected
= "FALSE"),
numericInput("freqAPP", "freqAPP", 20, min = 0, max = NA, step = 1),
selectInput("boundaryType", "boundaryType", choices=c("node", "edge"), selected
= "node"),
numericInput("nreg", "nreg", -1, min = 2, max = NA, step = 1),

helpText("only applicable for regression; related to parameter d describing the
minimum number of observations needed in a block to allow for fitting a
regression model. Defaults to 2*number of predictors."))))),

# main panel
mainPanel(
  tags$h3("Summary results of changepoint method:"),
  fluidRow(div(style='height:400px; overflow: scroll',
verbatimTextOutput("changepoint"))),
  tags$h3("Changepoint Plot:"),
  fluidRow(plotOutput("changepointPlot")))
)))

### Shiny server ###

server <- function(input, output) {

# Read data csv file

filedata <- reactive({
  inFile <- input$file
  if (is.null(inFile)){return(NULL)}

  x <- data.frame(read.csv(inFile$datapath, fileEncoding="UTF-8-BOM"))

```



```

x$Count0 <- x$Count
x$Count0[is.na(x$Count0)] <- 0

if(ncol(x)>3){
p = subset(x, select = -c(Year,Count,Count0))
}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)}}
}

if(input$datanorm == "rnorm"){
x$Count0 <- round(rnorm(x$Count0), digits = 4)
} else if (input$datanorm == "stand"){
x$Count0 <- round(scale(x$Count0), digits = 4)
} else {x$Count0 <- round(x$Count0, digits = 4)}

z = subset(x, select = c(Year,Count,Count0))

if(!is.null(p)){
  Final = cbind(z, p)
}else {
  Final = x
}
return(Final)
})

# Create predictors dataframe (optional)
predictors <- reactive({
  x <- filedata()
  p <- subset(x, select = -c(Year,Count,Count0))
  if(is.null(p)){pred = NULL}else{pred = p}
  return(p)
})

# Reactive functions for changepoint analysis techniques

fit <- reactive({
  fit <- fit.control(toll = input$toll, it.max = input$it.max, display =
input$display, last = input$last, maxit.glm = input$maxit.glm, h = input$h,
stop.if.error = input$stop.if.error)
return(fit)
})

sel <- reactive({
  sel <- sel.control(display = input$display, type = input$type, S =
input$S, Cn = input$Cn, alg = input$alg, edf.psi = input$edf.psi)
return(sel)
})

psi <- reactive({
  data <- filedata()
  x <- 1:nrow(data)

```

```

    if(input$psi == ""){
      psi = quantile(x, prob= seq(0,1,l=input$k+2)[-c(1,input$k+2)], names=FALSE)
      return(psi)
    } else {
      psi = as.numeric(unlist(strsplit(input$psi,"")))
      return(psi)
    }
  })

w0 <- reactive({

  inFile <- input$file
  if (is.null(inFile)){return(NULL)}

  x <- filedata()
  predictors <- predictors()

  w = as.numeric(unlist(strsplit(input$w0,"")))
  if (any(w > 1) | any(w < 0)){stop("Each element in w0 must be between 0 and
1.")}

  if(input$pred == "No"){return(NULL)}
  if(input$pred == "Yes"){
    if(length(w) == 0){
      w <- NULL
    } else {
      if(length(w) == ncol(predictors)){
        w <- w
      } else if(length(w) > ncol(predictors)){
        stop("Length of w0 is greater than the number of predictors.")
      } else if(length(w) == 1){
        w <- rep(w, ncol(predictors))
        print("Assume you wanted each error-to-signal ratio to be iid from U(0,
w0).")
      } else {
        stop("Length of w0 is less than the number of predictors.")
      }
    }
  }
  return(w)
})

```

### # Changepoint analysis techniques

```

changepoint <- reactive({

  data <- filedata()
  predictors = predictors()
  w0 <- w0()
  #adj <- adjx()

  # changepoint method
  if(input$changepoint == "changepoint"){

    if(input$method == "Gamma"){

      if(input$changes == "mean"){

```

```

    cpt = cpt.mean(data$Count0, penalty=input$penalty,
pen.value=input$pen.value, method=input$method, Q=input$Q,
    test.stat=input$test.stat, class=input$class,
param.estimates=input$param.estimates,
    shape=input$shape, minseglen=input$minseglen)

} else if(input$changes == "variance"){
    cpt = cpt.var(data$Count0, penalty=input$penalty,
pen.value=input$pen.value, method=input$method, Q=input$Q,
    test.stat=input$test.stat, class=input$class,
param.estimates=input$param.estimates,
    shape=input$shape, minseglen=input$minseglen)
} else{
    cpt = cpt.meanvar(data$Count0, penalty=input$penalty,
pen.value=input$pen.value, method=input$method, Q=input$Q,
    test.stat=input$test.stat, class=input$class,
param.estimates=input$param.estimates,
    shape=input$shape, minseglen=input$minseglen)

}} else{

if(input$changes == "mean"){
    cpt = cpt.mean(data$Count0, penalty=input$penalty,
pen.value=input$pen.value, method=input$method, Q=input$Q,
    test.stat=input$test.stat, class=input$class,
param.estimates=input$param.estimates,
    minseglen=input$minseglen)

} else if(input$changes == "variance"){
    cpt = cpt.var(data$Count0, penalty=input$penalty,
pen.value=input$pen.value, method=input$method, Q=input$Q,
    test.stat=input$test.stat, class=input$class,
param.estimates=input$param.estimates,
    minseglen=input$minseglen)

} else{
    cpt = cpt.meanvar(data$Count0, penalty=input$penalty,
pen.value=input$pen.value, method=input$method, Q=input$Q,
    test.stat=input$test.stat, class=input$class,
param.estimates=input$param.estimates,
    minseglen=input$minseglen)
}}
return(cpt)

# breakpoint method
} else if(input$changepoint == "breakpoint"){

    if(input$algorithm == "CE.NB"){
        bp = CE.NB(as.data.frame(data$Count0), Nmax = input$Nmax, eps = input$eps,
rho = input$rho, M = input$M, h = input$h,
a = input$a, b = input$b, distyp = input$distyp, penalty = input$penalty,
parallel = input$parallel)

} else if(input$algorithm == "CE.NB.Init"){
    bp = CE.NB.Init(as.data.frame(data$Count0),
as.numeric(unlist(strsplit(input$init.locs,","))),
eps = input$eps, rho = input$rho, M = input$M, h = input$h,
a = input$a, b = input$b, distyp = input$distyp, penalty = input$penalty,
var.init = input$var.init, parallel = input$parallel)

```

```

} else if(input$algorithm == "CE.NB.Init.Mean"){
  bp = CE.Normal.Init.Mean(as.data.frame(data$Count0),
  as.numeric(unlist(strsplit(input$init.locs,","))),
  eps = input$eps, rho = input$rho, M = input$M, h = input$h,
  a = input$a, b = input$b, distyp = input$distyp, penalty = input$penalty,
  var.init = input$var.init, parallel = input$parallel)

} else if(input$algorithm == "CE.NB.Init.MeanVar"){
  bp = CE.Normal.Init.MeanVar(as.data.frame(data$Count0),
  as.numeric(unlist(strsplit(input$init.locs,","))),
  eps = input$eps, rho = input$rho, M = input$M, h = input$h,
  a = input$a, b = input$b, distyp = input$distyp, penalty = input$penalty,
  var.init = input$var.init, parallel = input$parallel)

} else if(input$algorithm == "CE.Normal.Mean"){
  bp = CE.Normal.Mean(as.data.frame(data$Count0), Nmax = input$Nmax,
  eps = input$eps, rho = input$rho, M = input$M, h = input$h,
  a = input$a, b = input$b, distyp = input$distyp, penalty = input$penalty,
  parallel = input$parallel)

} else if(input$algorithm == "CE.Normal.MeanVar"){
  bp = CE.Normal.MeanVar(as.data.frame(data$Count0), Nmax = input$Nmax,
  eps = input$eps, rho = input$rho, M = input$M, h = input$h,
  a = input$a, b = input$b, distyp = input$distyp, penalty = input$penalty,
  parallel = input$parallel)

} else if(input$algorithm == "CE.ZINB"){
  bp = CE.ZINB(as.data.frame(data$Count0), Nmax = input$Nmax, eps = input$eps,
  rho = input$rho, M = input$M, h = input$h,
  a = input$a, b = input$b, distyp = input$distyp, penalty = input$penalty,
  parallel = input$parallel)

} else {
  bp = CE.ZINB.Init(as.data.frame(data$Count0),
  as.numeric(unlist(strsplit(input$init.locs,","))),
  eps = input$eps, rho = input$rho, M = input$M, h = input$h,
  a = input$a, b = input$b, distyp = input$distyp, penalty = input$penalty,
  var.init = input$var.init, parallel = input$parallel)

}
return(bp)

# cumSeg method
} else if(input$changepoint == "cumSeg"){

  x <- 1:nrow(data)

  cumSeg = jumpoints(as.matrix(data$Count0), x, output = input$output,
  k = input$k, round = input$round, psi = psi(), control = fit(),
  selection = sel())
  return(cumSeg)
} else {

if(input$pred == "Yes" && !is.null(predictors)){pred <- as.matrix(predictors)

# bcp method
} else {pred <- NULL}

```

```

    bcp = bcp::bcp(y = as.matrix(data$Count0), x = pred, id = NULL,
      adj = NULL, w0 = w0(),
      p0 = input$p0, d = input$d, burnin = input$burnin, mcmc = input$mcmc,
      return.mcmc = input$return.mcmc, boundaryType = input$boundaryType,
      p1 = input$p1,
      freqAPP = input$freqApp, nreg = input$nreg)
  return(bcp)
}
})

# Output of data table
output$contents <- DT::renderDataTable({
  if (is.null(input$file)) {return(NULL)}
  df <- DT::datatable(filedata())
  df
})

# Output of changepoint method summary results
output$changepoint <- renderPrint({
  req(input$file)
  changepoint()
})

# Output of changepoint method plot
output$changepointPlot <- renderPlot({
  req(input$file)
  if(input$changepoint == "breakpoint"){
    data <- filedata()
    profilePlot(changepoint(), as.data.frame(data$Count0))
  } else {
    plot(changepoint())
  }
})

```

## UDM - Shiny App for Single Species Univariate Changepoint Analysis

Choose data CSV File

Browse... Data.csv

Upload complete

Count data normalization:

none

Predictors normalization:

none

Annual Data

Changepoint Analysis

changepoint method

changepoint

Algorithms of changepoint package:

1. cpt.mean(data,penalty='MBIC',pen.value=0,met Q=5,test.stat='Normal',class=TRUE,param.estir minseglen=1)
2. cpt.var(data,penalty='MBIC',pen.value=0,known mu=NA,method='AMOC',Q=5,test.stat='Normal' class=TRUE,param.estimates=TRUE,minseglen=1)
3. cpt.meanvar(data,penalty='MBIC',pen.value=0,met Q=5,test.stat='Normal',class=TRUE,param.estir shape=1,minseglen=2)

type of changes to identify

Summary results of changepoint method:

```

Class 'cpt' : Changepoint Object
~ : S4 class containing 12 slots with names
  cpttype date version data.set method test.stat pen.type pen.value minseglen cpts ncpts.max param.est

Created on : Fri Jun 4 12:31:02 2021

summary(.) :
-----
Created Using changepoint version 2.2.2
Changepoint type : Change in mean
Method of analysis : PELT
Test Statistic : Normal
Type of penalty : AIC with value, 4
Minimum Segment Length : 1
Maximum no. of cpts : Inf
Changepoint Locations : 1 2 3 4 5 6 7 8

```

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 point detection 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](https://doi.org/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>.
- Chang W, 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>.