**ui.R**

#

# This is the user-interface definition of a Shiny web application. You can

# run the application by clicking 'Run App' above.

#

# Find out more about building applications with Shiny here:

#

# http://shiny.rstudio.com/

#

require(ggmap)

require(ggplot2)

require(scales)

require(shiny)

require(DT)

require(rgdal)

require(shapefiles)

require(foreign)

require(spacetime)

require(sp)

require(zoo)

require(xts)

shinyUI(pageWithSidebar(

# Header:

headerPanel("RAKUN SOFTWARE"),

# Input in sidepanel:

sidebarPanel(

conditionalPanel( condition="input.conditionedPanels==1",

helpText("Dataset Panel"),

tags$style(type='text/css', ".well { max-width: 20em; }"),

# Tags:

tags$head(

tags$style(type="text/css", "select[multiple] { width: 100%; height:10em}"),

tags$style(type="text/css", "select { width: 100%}"),

tags$style(type="text/css", "input { width: 19em; max-width:100%}")

),

# Select filetype:

selectInput("readFunction", "Function to read data:", c(

# Base R:

"read.table",

"read.csv",

"read.csv2",

"read.delim",

"read.delim2",

# foreign functions:

"read.spss",

"read.arff",

"read.dbf",

"read.dta",

"read.epiiinfo",

"read.mtp",

"read.octave",

"read.ssd",

"read.systat",

"read.xport",

# Advanced functions:

"scan",

"readLines"

)),

# Argument selecter:

htmlOutput("ArgSelect"),

# Argument field:

htmlOutput("ArgText"),

# Upload data:

fileInput("file", "Upload data-file:"),

fileInput('inputdata', 'Upload shapefile',accept=c('.shp','.dbf','.sbn','.sbx','.shx',".prj"), multiple=TRUE),

uiOutput("choose\_columns")

),

conditionalPanel(condition="input.conditionedPanels==2",

helpText("Plot Variables"),

textInput("filter", "X-axis:", value = "", width = NULL, placeholder = NULL),

textInput("filter2", "Y-axis:", value = NULL, width = NULL, placeholder = NULL),

helpText("Get Map"),

actionButton("action", label = "Help"),

uiOutput("HelpBox"),

textInput("filter3", "LAT1:", value = "", width = NULL, placeholder = NULL),

textInput("filter4", "LON1:", value = NULL, width = NULL, placeholder = NULL),

textInput("filter5", "LAT2:", value = "", width = NULL, placeholder = NULL),

textInput("filter6", "LON2:", value = NULL, width = NULL, placeholder = NULL),

actionButton("action8", label = "Get Map by Location Name"),

textInput("filter20", "Location Name:", value = NULL, width = NULL, placeholder = NULL)

),

conditionalPanel(condition="input.conditionedPanels==3",

helpText("Plot Variables"),

actionButton("action6", label = "Filter Data"),

textInput("filter7", "Search:", value = "", width = NULL, placeholder = NULL),

uiOutput("combobox15"),

uiOutput("combobox6"),

uiOutput("combobox7"),

uiOutput("combobox8"),

uiOutput("choose\_columns4"),

uiOutput("combobox2"),

uiOutput("combobox"),

uiOutput("slider")

) ,

conditionalPanel(condition="input.conditionedPanels==4",

helpText("Plot Variables"),

actionButton("action5", label = "Filter Data"),

textInput("filter11", "Search:", value = "", width = NULL, placeholder = NULL),

uiOutput("combobox14"),

uiOutput("combobox12"),

uiOutput("combobox13"),

uiOutput("choose\_columns3"),

uiOutput("combobox3")

),

conditionalPanel(condition="input.conditionedPanels==5",

helpText("Plot Variables"),

actionButton("action9", label = "Filter Data"),

textInput("filter15", "Search:", value = "", width = NULL, placeholder = NULL),

textInput("filter16", "X-axis:", value = NULL, width = NULL, placeholder = NULL),

textInput("filter24", "Y-axis:", value = "", width = NULL, placeholder = NULL),

uiOutput("combobox4")

),

conditionalPanel(condition="input.conditionedPanels==6",

helpText("Plot Variables"),

actionButton("action7", label = "Filter Data"),

textInput("filter17", "Search:", value = "", width = NULL, placeholder = NULL),

uiOutput("combobox11"),

uiOutput("combobox9"),

uiOutput("combobox10"),

uiOutput("choose\_columns2"),

uiOutput("combobox5"),

uiOutput("slider2")

),

conditionalPanel(condition="input.conditionedPanels==7",

helpText("Plot Variables"),

textInput("filter19", "Search:", value = "", width = NULL, placeholder = NULL),

uiOutput("combobox16"),

uiOutput("choose\_columns5")

)

),

# Main:

mainPanel(

tabsetPanel(

tabPanel("Dataset", DT:: dataTableOutput("data\_table"),value = 1),

tabPanel("Location Plot", plotOutput("data\_point\_plot"),value=2),

tabPanel("Map Plot", plotOutput("map\_by\_year\_selection"),value=3),

tabPanel("St Plot", plotOutput("st\_plot"),value=4),

tabPanel("Plot", plotOutput("plot"),value=5),

tabPanel("Moran Plot", plotOutput("moran\_plot"),value=6),

tabPanel("Correlation Plot", plotOutput("corr\_plot"),value=7),

id="conditionedPanels"

)

)

))

**Server.R**

#

# This is the server logic of a Shiny web application. You can run the

# application by clicking 'Run App' above.

#

# Find out more about building applications with Shiny here:

#

# http://shiny.rstudio.com/

#

options(shiny.maxRequestSize=150\*1024^2)

require(lubridate)

require(tcltk)

require(leaflet)

require(ggthemes)

require(RColorBrewer)

require(spdep)

require(tcltk)

require(shiny)

require(data.table)

require(plotly)

require(scales)

require(DT)

require(dplyr)

require(ggplot2)

require(reshape2)

require(maptools)

require(corrplot)

shinyServer(function(input, output) {

### Argument names:

ArgNames <- reactive({

Names <- names(formals(input$readFunction)[-1])

Names <- Names[Names!="..."]

return(Names)

})

# Argument selector:

output$ArgSelect <- renderUI({

if (length(ArgNames())==0) return(NULL)

selectInput("arg","Argument:",ArgNames())

})

## Arg text field:

output$ArgText <- renderUI({

fun\_\_arg <- paste0(input$readFunction,"\_\_",input$arg)

if (is.null(input$arg)) return(NULL)

Defaults <- formals(input$readFunction)

if (is.null(input[[fun\_\_arg]]))

{

textInput(fun\_\_arg, label = "Enter value:", value = deparse(Defaults[[input$arg]]))

} else {

textInput(fun\_\_arg, label = "Enter value:", value = input[[fun\_\_arg]])

}

})

### Data import:

Dataset <- reactive({

if (is.null(input$file)) {

# User has not uploaded a file yet

return(data.frame())

}

args <- grep(paste0("^",input$readFunction,"\_\_"), names(input), value = TRUE)

argList <- list()

for (i in seq\_along(args))

{

argList[[i]] <- eval(parse(text=input[[args[i]]]))

}

names(argList) <- gsub(paste0("^",input$readFunction,"\_\_"),"",args)

argList <- argList[names(argList) %in% ArgNames()]

Dataset <- as.data.frame(do.call(input$readFunction,c(list(input$file$datapath),argList)))

return(Dataset)

})

output$choose\_columns <- renderUI( {

# Get the data set with the appropriate name

colnames <- names(Dataset())

# Create the checkboxes and select them all by default

checkboxGroupInput("columns", "Choose columns",

choices = colnames,selected=TRUE)

})

output$data\_table <- DT::renderDataTable({

A <- Dataset()

DT::datatable(A[, input$columns, drop = FALSE])

})

output$value <- renderPrint({ input$filter })

output$value <- renderPrint({ input$filter2 })

output$value <- renderPrint({ input$filter3 })

output$value <- renderPrint({ input$filter4 })

output$value <- renderPrint({ input$filter5 })

output$value <- renderPrint({ input$filter6 })

output$value <- renderPrint({ input$filter7 })

output$value <- renderPrint({ input$filter8 })

output$value <- renderPrint({ input$filter9 })

output$value <- renderPrint({ input$filter11 })

output$value <- renderPrint({ input$filter12 })

output$value <- renderPrint({ input$filter13 })

output$value <- renderPrint({ input$filter14 })

output$value <- renderPrint({ input$filter15 })

output$value <- renderPrint({ input$filter16 })

output$value <- renderPrint({ input$filter17 })

output$value <- renderPrint({ input$filter18 })

output$value <- renderPrint({ input$filter20 })

output$value <- renderPrint({ input$filter21 })

output$value <- renderPrint({ input$filter22 })

output$value <- renderPrint({ input$filter23 })

output$value <- renderPrint({ input$filter24 })

output$combobox<-renderUI( {

P<-names(Dataset())

selectizeInput("combo", "Choose Date", choices=P, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox2<-renderUI( {

Q<-names(Dataset())

selectizeInput("combo2", "Search in column", choices=Q, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox3<-renderUI( {

G<-names(Dataset())

selectizeInput("combo3", "Search in column", choices=G, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox4<-renderUI( {

U<-names(Dataset())

selectizeInput("combo4", "Search in column", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox5<-renderUI( {

U<-names(Dataset())

selectizeInput("combo5", "Search in column", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox6<-renderUI( {

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

shape@data$ID<-c(1:length(shape@data[,1]))

A<-as.data.frame(shape)

U<-names(A)

selectizeInput("combo6", "Object Id in Shapefile", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox7<-renderUI( {

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

shape@data$ID<-c(1:length(shape@data[,1]))

A<-as.data.frame(shape)

U<-names(A)

selectizeInput("combo7", "Location Name in Shapefile:", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox8<-renderUI( {

U<-names(Dataset())

selectizeInput("combo8", "Location Name in Dataset", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox9<-renderUI( {

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

A<-as.data.frame(shape)

U<-names(A)

selectizeInput("combo9", "Location Name in Shapefile:", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox10<-renderUI( {

U<-names(Dataset())

selectizeInput("combo10", "Location Name in Dataset", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox11<-renderUI( {

U<-names(Dataset())

selectizeInput("combo11", "Group by Time Component:", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox12<-renderUI( {

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

A<-as.data.frame(shape)

U<-names(A)

selectizeInput("combo12", "Location Name in Shapefile:", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox13<-renderUI( {

U<-names(Dataset())

selectizeInput("combo13", "Location Name in Dataset", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox14<-renderUI( {

U<-names(Dataset())

selectizeInput("combo14", "Group by Time Component:", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox15<-renderUI( {

U<-names(Dataset())

selectizeInput("combo15", "Group by Time Component:", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$combobox16<-renderUI( {

U<-names(Dataset())

selectizeInput("combo16", "Search in column", choices=U, selected = NULL, multiple = FALSE,

options = NULL)

})

output$slider<-renderUI({

B<-Dataset()

W<-B[,input$combo]

M<-max(W,na.rm=TRUE)

Z<-min(W,na.rm=TRUE)

numericInput("date", "Date:",Z,

min = Z, max = M)

})

output$slider2<-renderUI({

B<-Dataset()

W<-B[,input$combo11]

M<-max(W,na.rm=TRUE)

Z<-min(W,na.rm=TRUE)

numericInput("date2", "Date:",Z,

min = Z, max = M)

})

observe({

output$plot<-renderPlot({

if (input$action9%%2) {

B<-Dataset()

Y1<-B[,input$combo4]

B$Date = mdy(paste( B$Month, B$Day,B$Date))

filtered\_data <- B %>% filter(grepl(input$filter15, Y1))

filtered\_data2 <- filtered\_data %>%

group\_by\_(as.symbol(input$filter16)) %>%

summarize(count = n())

ggplot(filtered\_data2, aes\_string(x=input$filter16,y=input$filter24)) +

geom\_line(color = "#F2CA27", size = 0.5) +

geom\_smooth(color = "#1A1A1A") +

theme\_solarized(light=FALSE)

}

else{

B<-Dataset()

Y1<-B[,input$combo4]

filtered\_data <- B %>% filter(grepl(input$filter15, Y1))

ggplot(filtered\_data, aes\_string(x=input$filter16,y=input$filter24)) +

geom\_line(color = "#F2CA27", size = 0.5) +

geom\_smooth(color = "#1A1A1A") +

theme\_solarized(light=FALSE)

}

})

})

observe({

output$data\_point\_plot<-renderPlot({

if (input$action8%%2) {

B<-Dataset()

map1 <- get\_map(location = input$filter20, source = "stamen", maptype = "toner-lite")

ggmap(map1) +

geom\_point(data = B, aes\_string(x=input$filter, y=input$filter2), color = "#27AE60", size = 0.5, alpha = 0.01) +

theme\_solarized()

}

else{

B<-Dataset()

lat1<-as.numeric(input$filter3)

lon1<-as.numeric(input$filter4)

lat2<-as.numeric(input$filter5)

lon2<-as.numeric(input$filter6)

bbox = c(lat1,lon1,lat2,lon2)

map <- get\_map(location = bbox, source = "stamen", maptype = "toner-lite")

ggmap(map) +

geom\_point(data = B, aes\_string(x=input$filter, y=input$filter2), color = "#27AE60", size = 0.5, alpha = 0.01) +

theme\_solarized()

}

})

})

output$choose\_columns3 <- renderUI( {

# Get the data set with the appropriate name

colnames <- names(Dataset())

# Create the checkboxes and select them all by default

checkboxGroupInput("columns3", "Group by:",

choices = colnames,selected=TRUE)

})

observe({

output$st\_plot<-renderPlot({

if (input$action5%%2) {

B<-Dataset()

Y1<-B[,input$combo3]

filtered\_data <- B %>% filter(grepl(input$filter11, Y1))

dots<-input$columns3

filtered\_data2 <- filtered\_data %>%

group\_by\_(.dots=dots) %>%

summarize(count = n())

#read shpfile

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

districts <-shape

P<-districts@data[,input$combo12]

Z<-unlist(P)

row.names(districts) <- as.character(Z)

districts@polygons<-districts@polygons[order(Z)]

D1<-filtered\_data2[,input$combo14]

D2<-unlist(D1)

o <- order(D2)

sf1<-filtered\_data2[o, ]

K<-sf1[,input$combo14]

L<-unlist(K)

dts0 <- paste(as.character(unique(L), 2000), "-01-01", sep = "")

dts2 <- as.POSIXct(dts0, format = "%Y-%m-%d")

sf1<-as.data.frame(sf1)

districts@data <- data.frame(districts@data, sf1[match(districts@data[,input$combo12], sf1[,input$combo13]),])

sf\_st <- STFDF(as(districts,"SpatialPolygons"),dts2,sf1, endTime=delta(dts2))

stplot(sf\_st[, , "count"],mode = "ts", auto.key = list(space = "right",cex = 0.7))

}

else

{

B<-Dataset()

#read shpfile

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

districts <-shape

P<-districts@data[,input$combo12]

Z<-unlist(P)

row.names(districts) <- as.character(Z)

districts@polygons<-districts@polygons[order(Z)]

D1<-B[,input$combo14]

D2<-unlist(D1)

o <- order(D2)

sf1<-B[o, ]

K<-sf1[,input$combo14]

L<-unlist(K)

dts0 <- paste(as.character(unique(L), 1900), "-01-01", sep = "")

dts2 <- as.POSIXct(dts0, format = "%Y-%m-%d")

sf1<-as.data.frame(sf1)

districts@data <- data.frame(districts@data, sf1[match(districts@data[,input$combo12], sf1[,input$combo13]),])

sf\_st <- STFDF(as(districts,"SpatialPolygons"),dts2,sf1, endTime=delta(dts2))

stplot(sf\_st[, , input$filter11],mode = "ts", auto.key = list(space = "right",cex = 0.7))

}

})

})

output$choose\_columns2 <- renderUI( {

# Get the data set with the appropriate name

colnames <- names(Dataset())

# Create the checkboxes and select them all by default

checkboxGroupInput("columns2", "Group by:",

choices = colnames,selected=TRUE)

})

observe({

output$moran\_plot<-renderPlot({

if (input$action7%%2) {

B<-Dataset()

Y1<-B[,input$combo5]

filtered\_data <- B %>% filter(grepl(input$filter17, Y1))

dots<-input$columns2

filtered\_data2 <- filtered\_data %>%

group\_by\_(.dots=dots) %>%

summarize(count = n())

D1<-filtered\_data2[,input$combo11]

D2<-unlist(D1)

o <- order(D2)

sf1<-filtered\_data2[o, ]

sf1<-as.data.frame(sf1)

#read shpfile

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

districts <-shape

P<-districts@data[,input$combo9]

Z<-unlist(P)

districts@polygons<-districts@polygons[order(Z)]

districts@data <- data.frame(districts@data, sf1[match(districts@data[,input$combo9], sf1[,input$combo10]),])

D3<-sf1[,input$combo11]

D4<-unlist(D3)

A<-subset(sf1,D4==as.numeric(input$date2))

districts\_nb<-poly2nb(districts,queen = FALSE)

listw <- nb2listw(districts\_nb)

moran.plot(A$count,listw)

}

else{

B<-Dataset()

D1<-B[,input$combo11]

D2<-unlist(D1)

o <- order(D2)

sf1<-B[o, ]

sf1<-as.data.frame(sf1)

#read shpfile

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

districts <-shape

P<-districts@data[,input$combo9]

Z<-unlist(P)

districts@polygons<-districts@polygons[order(Z)]

districts@data <- data.frame(districts@data, sf1[match(districts@data[,input$combo9], sf1[,input$combo10]),])

D3<-sf1[,input$combo11]

D4<-unlist(D3)

A<-subset(sf1,D4==as.numeric(input$date2))

B<-A[,input$filter17]

districts\_nb<-poly2nb(districts,queen = FALSE)

listw <- nb2listw(districts\_nb)

moran.plot(B,listw)

}

})

})

output$choose\_columns5<- renderUI( {

# Get the data set with the appropriate name

colnames <- names(Dataset())

# Create the checkboxes and select them all by default

checkboxGroupInput("columns5", "Choose columns",

choices = colnames,selected=TRUE)

})

output$corr\_plot<-renderPlot({

B<-Dataset()

Y1<-B[,input$combo16]

filtered\_data <- B %>% filter(grepl(input$filter19, Y1))

dots<-input$columns5

filtered\_data2 <- filtered\_data %>%

group\_by\_(.dots=dots) %>%

summarize()

filtered\_data2<-as.data.frame(filtered\_data2)

M<-cor(filtered\_data2)

corrplot(M,method = "circle")

})

output$choose\_columns4 <- renderUI( {

# Get the data set with the appropriate name

colnames <- names(Dataset())

# Create the checkboxes and select them all by default

checkboxGroupInput("columns4", "Choose columns",

choices = colnames,selected=TRUE)

})

observe({

output$map\_by\_year\_selection<-renderPlot({

if (input$action6%%2) {

B<-Dataset()

Y1<-B[,input$combo2]

filtered\_data <- B %>% filter(grepl(input$filter7, Y1))

dots<-input$columns4

filtered\_data2 <- filtered\_data %>%

group\_by\_(.dots=dots) %>%

summarize(count = n())

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

districts <-shape

P<-districts@data[,input$combo7]

Z<-unlist(P)

districts@polygons<-districts@polygons[order(Z)]

D1<-filtered\_data2[,input$combo15]

D2<-unlist(D1)

o <- order(D2)

sf1<-filtered\_data2[o, ]

K<-sf1[,input$combo15]

L<-unlist(K)

dts0 <- paste(as.character(unique(L), 2000), "-01-01", sep = "")

dts2 <- as.POSIXct(dts0, format = "%Y-%m-%d")

sf1<-as.data.frame(sf1)

districts.df <- fortify(districts)

districts.df <- merge(districts.df, districts@data, by.x="id", by.y=input$combo6, all.x=T, a..ly=F)

districts.df <- merge(districts.df, sf1, by.x=input$combo7, by.y=input$combo8, all.x=T, a..ly=F)

D3<-districts.df[,input$combo15]

D4<-unlist(D3)

A<-subset(districts.df,D4==as.numeric(input$date))

ggp <- ggplot(data=A, aes(x=long, y=lat, group=group)) + geom\_polygon(aes(fill=count)) # draw polygons

ggp <- ggp + geom\_path(color="grey", linemitre=2) # draw boundaries

ggp <- ggp + coord\_equal()

ggp <- ggp + scale\_fill\_gradient(low = "#ffffcc", high = "#ff4444",

space = "Lab", na.value = "grey50",

guide = "colourbar")

print(ggp)

ggp

}

else{

B<-Dataset()

myshape<- input$inputdata

if (is.null(myshape))

return(NULL)

dir<-dirname(myshape[1,4])

for ( i in 1:nrow(myshape)) {

file.rename(myshape[i,4], paste0(dir,"/",myshape[i,1]))}

getshp <- list.files(dir, pattern="\*.shp", full.names=TRUE)

shape<-readShapePoly(getshp)

districts <-shape

districts@data$ID<-c(1:length(districts@data[,1]))

P<-districts@data[,input$combo7]

Z<-unlist(P)

districts@polygons<-districts@polygons[order(Z)]

D1<-B[,input$combo15]

D2<-unlist(D1)

o <- order(D2)

sf1<-B[o, ]

K<-sf1[,input$combo15]

L<-unlist(K)

dts0 <- paste(as.character(unique(L), 1900), "-01-01", sep = "")

dts2 <- as.POSIXct(dts0, format = "%Y-%m-%d")

sf1<-as.data.frame(sf1)

districts.df <- fortify(districts)

districts.df <- merge(districts.df, districts@data, by.x="id", by.y=input$combo6, all.x=T, a..ly=F)

districts.df <- merge(districts.df, sf1, by.x=input$combo7, by.y=input$combo8, all.x=T, a..ly=F)

D3<-districts.df[,input$combo15]

D4<-unlist(D3)

A<-subset(districts.df,D4==as.numeric(input$date))

ggp <- ggplot(data=A, aes(x=long, y=lat, group=group)) + geom\_polygon(aes(fill=count)) # draw polygons

ggp <- ggp + geom\_path(color="grey", linemitre=2) # draw boundaries

ggp <- ggp + coord\_equal()

ggp <- ggp + scale\_fill\_gradient(low = "#ffffcc", high = "#ff4444",

space = "Lab", na.value = "grey50",

guide = "colourbar")

print(ggp)

ggp

}

})

})

output$HelpBox = renderUI({

if (input$action%%2){helpText("Define limits of your map's X and Y coordinates")

}else return()

})

})