# ---------------------------------------------------------------------------- #

# ACTIVITY 1.4g TIMESERIES\_4 SERVER

# ---------------------------------------------------------------------------- #

# Load in libraries

library(shiny)

library(dplyr)

library(ggplot2)

library(RColorBrewer)

library(lubridate)

# Load in the raw data

raw\_data <- read.csv("data/raw\_data.csv", stringsAsFactors=FALSE)

# Create a colour palette

col\_palette <- brewer.pal(name="Dark2", n=8)

# Collect list of years

raw\_data$date <- as.Date(raw\_data$date) # Change the structure of 'date' to a date

yrs <- c(unique(year(raw\_data$date)), 2015) #Extract year and take only the unique years

# Set plot breaks

plot\_breaks = seq(from=0, to=12\*length(yrs)-1, by=12)

# Summarise for 'all' data - this will always be plotted

overall\_summary <- raw\_data %>%

group\_by(month) %>%

summarise(n = length(month)) %>%

mutate(region="All data",

species="All data")

# Summarise for 'all' sexes, divided by region data

region\_allspecies\_summary <- raw\_data %>%

group\_by(month, region, age) %>%

summarise(n = length(month)) %>%

mutate(species="All species")

# Summarise for 'all' region, divided by sex data

species\_allregions\_summary <- raw\_data %>%

group\_by(month, species, age) %>%

summarise(n = length(month)) %>%

mutate(region="All regions")

# Summarise region and sex data

region\_species\_summary <- raw\_data %>%

group\_by(month, region, species, age) %>%

summarise(n = length(month))

# Join summary data together

summary\_data <- bind\_rows(overall\_summary, region\_allspecies\_summary, species\_allregions\_summary, region\_species\_summary)

# Create a dataframe that matches the

start\_df <- summary\_data %>%

filter(region=="All regions" | region=="All data") %>%

group\_by(month, region, species) %>%

summarise(n = sum(n))

# Collect min and max ages for the slider

min\_age <- min(raw\_data$age)

max\_age <- max(raw\_data$age)

#------------------------------------------------------------------------------#

# Begin server section

shinyServer(function(input, output, session) {

# Setup reactiveValues object that can be updated reactively

data\_subset <- reactiveValues(data = start\_df)

# Subset for the chosen region and sexes

observeEvent(input$action\_button, {

# Subset for region

data\_sub = summary\_data %>%

filter(region==input$select\_region | region=="All data")

# Subset for sex

if(length(input$select\_species)>0){

data\_sub = data\_sub %>%

filter(species %in% input$select\_species | species=="All species" | species=="All data")

} else {

data\_sub = data\_sub %>%

filter(species=="All species" | species=="All data")

}

# Subset for age

data\_sub = data\_sub %>%

filter(age >= input$age\_slider[1] & age <= input$age\_slider[2] | is.na(age))

# Summarise data using subsets created above

data\_sub = data\_sub %>%

group\_by(month, region, species) %>%

summarise(n = sum(n))

data\_subset$data = as.data.frame(data\_sub)

})

# Reset the app

observeEvent(input$reset\_button, {

# Use the default data when reset

data\_subset$data = as.data.frame(start\_df)

# Reset the widgets to their default values

updateSelectInput(session, inputId = "select\_region", selected = "All regions")

updateCheckboxGroupInput(session, inputId = "select\_species", selected = c("cat", "dog", "human", "jackal", "lion"))

updateSliderInput(session, inputId = "age\_slider", min = min\_age, max = max\_age, value = c(min\_age, max\_age))

})

# Produce plot

output$tsPlot <- renderPlot({

ggplot() +

geom\_path(data=data\_subset$data, aes(x=month, y=n, color=species), size=1) +

scale\_color\_manual(name="Species", values=col\_palette) +

labs(title=input$select\_region, x="Date (Month)", y="Number of records") +

# Extra plotting code to control appearence

scale\_x\_continuous(breaks=plot\_breaks, labels=yrs, limits=c(min(overall\_summary$month), max(overall\_summary$month))) +

theme\_classic() +

theme(axis.text = element\_text(size=14),

axis.title = element\_text(size=18),

plot.title = element\_text(size=20),

legend.title = element\_text(size=18),

legend.text = element\_text(size=14))

})

})

# ---------------------------------------------------------------------------- #

# ACTIVITY 1.4g TIMESERIES\_4 UI

# ---------------------------------------------------------------------------- #

# Load in libraries

library(shiny)

library(dplyr)

library(ggplot2)

# Load in the raw data

raw\_data <- read.csv("data/raw\_data.csv", stringsAsFactors=FALSE)

# Collect a list of regions for the dropdown menu

options\_list <- c("All regions", sort(unique(raw\_data$region)))

# Collect min and max ages for the slider

min\_age <- min(raw\_data$age)

max\_age <- max(raw\_data$age)

#------------------------------------------------------------------------------#

# Begin ui section

shinyUI(fluidPage(

titlePanel("Day 1 - Timeseries\_4"),

# Add a line break

br(),

# Add text section

h4("This app is identical to the last, with a new widget: actionButton"),

h4("The action button triggers the plot update after the user has set the other widget values"),

# Add a line break

br(),

# Sidebar with a slider input for number of bins

sidebarLayout(

sidebarPanel(

# Add a dropdown menu widget

selectInput("select\_region", label = h3("Select a Region:"),

choices = options\_list,

selected = 1),

br(),

# Add a checkbox widget

checkboxGroupInput("select\_species", label = h3("Select a Species"),

choices = list("Cat" = "cat", "Dog" = "dog", "Human"="human",

"Jackal"="jackal", "Lion"="lion"),

selected = c("cat", "dog", "human", "jackal", "lion")),

br(),

# Add a slider

sliderInput("age\_slider", label = h3("Select a maximum age"),

min = min\_age, max = max\_age, value = c(min\_age, max\_age)),

br(),

# Add an action button

actionButton("action\_button", label = "Update plot"),

# Add another action button

actionButton("reset\_button", label = "Reset plot")

),

# Show a plot of the generated distribution

mainPanel(

plotOutput("tsPlot", height=700)

)

)

))