STARTING APP

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

# ACTIVITY 1.4d TIMESERIES\_1 SERVER

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

# Load in libraries

library(shiny)

library(dplyr)

library(ggplot2)

library(lubridate)

library(RColorBrewer)

# 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

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

# Summarise for 'all' data

overall\_summary <- raw\_data %>%

group\_by(month) %>%

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

mutate(species="All Species")

# Summarise data by region

species\_summary <- raw\_data %>%

group\_by(month, species) %>%

summarise(n = length(month))

# Join summary data together

summary\_data <- bind\_rows(overall\_summary, species\_summary)

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

# Begin server section

shinyServer(function(input, output) {

# Subset for the chosen region

data\_subset <- reactive({

data\_sub = summary\_data %>%

filter(species==input$select\_species | species=="All Species")

as.data.frame(data\_sub)

})

# Produce plot

output$tsPlot <- renderPlot({

ggplot() +

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

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

labs(title=input$select\_species, 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))

})

})

STARTING APP

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

# ACTIVITY 1.4d TIMESERIES\_1 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

species\_list <- c("All Species", sort(unique(raw\_data$species)))

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

# Begin UI section

shinyUI(fluidPage(

# Application title

titlePanel("Day 1 - Timeseries\_1"),

# Add a line break

br(),

# Add text section

h4("This app is a little more complicated than the first one."),

h4("There is still only 1 dropdown menu, but this time we are changing the species we want to view on the plot."),

# Add a line break

br(),

# Sidebar with a slider input for number of bins

sidebarLayout(

sidebarPanel(

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

choices = species\_list,

selected = 1)

),

# Show plot

mainPanel(

plotOutput("tsPlot", height=700)

)

)

))

COMPLETED APP

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

# ACTIVITY 1.4d TIMESERIES\_1 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

# Plot Breaks

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

# Summarise for 'all' data

overall\_summary <- raw\_data %>%

group\_by(month) %>%

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

mutate(region = "All Regions")

# Summarise data by region

region\_summary <- raw\_data %>%

group\_by(month, region) %>%

summarise(n = length(month))

# Join summary data together

summary\_data <- bind\_rows(overall\_summary, region\_summary)

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

# Begin server section

shinyServer(function(input, output) {

# Subset for the chosen region

data\_subset <- reactive({

data\_sub = summary\_data %>%

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

as.data.frame(data\_sub)

})

# Produce plot

output$tsPlot <- renderPlot({

ggplot() +

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

scale\_color\_manual(name="Region", 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), legend.text = element\_text(size=14),

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

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

})

})

COMPLETED APP

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

# ACTIVITY 1.4d TIMESERIES\_1 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

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

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

# Begin ui section

shinyUI(fluidPage(

# Application title

titlePanel("Exploratory plots: Timeseries\_1 (Master)"),

# Add a line break

br(),

# Add text section

h4("This app is a little more complicated than the first one."),

h4("There is still only 1 widget (selectInput), but this time we are changing the region we want to view on the plot."),

# Add a line break

br(),

# Sidebar with a slider input for number of bins

sidebarLayout(

sidebarPanel(

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

choices = region\_list,

selected = 1)

),

# Show a plot of the generated distribution

mainPanel(

plotOutput("tsPlot", height=700)

)

)

))