1920 lines (1885 sloc) 77.3 KB

|  |  |
| --- | --- |
|  | library(shiny) |
|  | library(dplyr) |
|  | library(lubridate) |
|  | library(shinyAce) |
|  | library(ggplot2) |
|  | library(googlesheets) |
|  | library(DT) |
|  | library(shinydashboard) |
|  | bowlsgs <- gs\_title("Bowls") |
|  | winner = data.frame(Game = c("Celebration Bowl", "New Orleans Bowl", "Cure Bowl", "Las Vegas Bowl","New Mexico Bowl","Camellia Bowl", "Cheribundi Tart Cherry Boca Raton Bowl", "DXL Frisco Bowl", "Bad Boy Mowers Gasparilla Bowl", "Bahamas Bowl", "Famous Idaho Potato Bowl", "Birmingham Bowl", "Lockheed Martin Armed Forces Bowl", "Dollar General Bowl", "Hawai'i Bowl", "Zaxby's Heart of Dallas Bowl", "Quick Lane Bowl", "Cactus Bowl", "Walk-On's Independence Bowl", "New Era Pinstripe Bowl", "Foster Farm's Bowl", "Academy Sports + Outdoors Texas Bowl", "Military Bowl", "Camping World Bowl", "Alamo Bowl", "Holiday Bowl", "Belk Bowl", "Sun Bowl", "Music City Bowl","Arizona Bowl", "Cotton Bowl","TaxSlayer Bowl", "Liberty Bowl", "Fiesta Bowl", "Orange Bowl", "Outback Bowl", "Peach Bowl", "Citris Bowl","Rose Bowl", "Sugar Bowl", "National Championship"), |
|  | Winner = c("North Carolina A&T", "Troy", "Georgia State", "Boise State", "Marshall", "Middle Tennessee", "Florida Atlantic","Louisiana Tech","Temple", "Ohio", "Wyoming", "Southern Florida", "Army", "Appalachain State", "Fresno State", "Utah", "Duke", "Kansas State", "Florida State", "Iowa", "Purdue", "Texas","Navy","Oklahoma State", "TCU", "Michigan State", "Wake Forrest", "NC State", "Northwestern", "Utah State", "Ohio State", "Mississippi State", "Iowa State", "Penn State", "Wisconsin", "South Carolina", "UCF", "Notre Dame","Georgia","Alabama","Alabama"), |
|  | Away = c("Grambling", "Troy", "Western Kentucky", "Boise State", "Marshall", "Middle Tennessee", "Akron", "Louisiana Tech", "Temple", "UAB", "Central Michigan", "Texas Tech", "San Diego State", "Appalachian State", "Fresno State", "Utah","Duke", "Kansas State","Southern Miss", "Iowa", "Arizona", "Texas", "Virginia", "Virginia Tech", "Stanford", "Washington State", "Wake Forest", "NC State", "Kentucky", "Utah State", "USC", "Louisville", "Iowa State", "Washington","Miami", "Michigan", "UCF", "Notre Dame", "Georgia", "Alabama", "Alabama"), |
|  | Home = c("North Carolina A&T", "North Texas", "Georgia State", "Oregon", "Colorado State", "Arkansas State", "Florida Atlantic", "SMU", "Florida International", "Ohio", "Wyoming", "South Florida", "Army","Toledo", "Houston","West Virginia", "Northern Illinios", "UCLA", "Florida State", "Boston College", "Purdue", "Missouri", "Navy", "Oklahoma State", "TCU", "Michigan State","Texas A&M", "Arizona State", "Northwestern", "New Mexico State", "Ohio State", "Mississippi State", "Memphis", "Penn State", "Wisconsin", "South Carolina", "Auburn", "LSU","Oklahoma","Clemson","Georgia") |
|  | ) |
|  |  |
|  | # http://fontawesome.io/icons/ |
|  | # Define UI for slider demo app ---- |
|  | ui <- dashboardPage(skin = "yellow", |
|  |  |
|  | # App title ---- |
|  | dashboardHeader(title = "D Math Pick'em"), |
|  |  |
|  | # Sidebar layout with input and output definitions ---- |
|  | # Sidebar to demonstrate various slider options ---- |
|  | dashboardSidebar( |
|  | sidebarMenu( |
|  | menuItem("Instructions", tabName = "instructions", icon = icon("newspaper-o")), |
|  | menuItem("Information", tabName = "information", icon = icon("dashboard"), startExpanded = TRUE, |
|  | textInput("first", label = "First Name", value = "dusty"), |
|  | textInput("last", label = "Last Name", value = "turner")), |
|  | menuItem("Selections", tabName = "selections", icon = icon("edit")), |
|  | menuItem("Picks", tabName = "picks", icon = icon("database")), |
|  | menuItem("Standings", tabName = "standings", icon = icon("sort-numeric-asc")) |
|  | ) |
|  | ), |
|  |  |
|  | dashboardBody( |
|  | tabItems( |
|  | tabItem(tabName = "selections", |
|  | # actionButton("uploader", "Submit Picks"), |
|  | tags$hr(), |
|  | box(title = "Champianship: 8 January 2018", status = "success", solidHeader = TRUE, collapsible = TRUE, width = 4, |
|  | box(title = winner$Game[41], status = "danger", solidHeader = TRUE, collapsible = TRUE, width = 12, |
|  | uiOutput("addsliderbar41") |
|  | ) |
|  | ), |
|  | box(title = "16 December 2017", status = "success", solidHeader = TRUE, collapsible = TRUE, width = 12, |
|  | box(title = winner$Game[1], status = "danger", solidHeader = TRUE, collapsible = TRUE, width = 4, |
|  | uiOutput("addsliderbar1") |
|  | ), |
|  | ), |
|  | tabItem(tabName = "picks", |
|  | tabPanel("All Selections", dataTableOutput("fromgoogledatatable"))), |
|  | tabItem(tabName = "instructions", tabPanel("Instructions", htmlOutput("instructions"), plotOutput("translation"))), |
|  | tabItem(tabName = "standings", tabPanel("Standings", dataTableOutput("standings"), plotOutput("outlook"))) |
|  | # tabPanel("All Selections", tableOutput("fromgoogle")) |
|  | ) |
|  | ) |
|  | ) |
|  |  |
|  |  |
|  |  |
|  | server <- function(input, output) { |
|  |  |
|  | # Reactive expression to create data frame of all input values ---- |
|  | sliderValues1 <- reactive({ |
|  | gamedf = data.frame( |
|  | Game = winner$Game[1], |
|  | Away = winner$Away[1], |
|  | Home = winner$Home[1], |
|  | Pick = c(input$game1pick), |
|  | Confidence = c(input$game1confidence), |
|  | stringsAsFactors = FALSE) |
|  | gamedf %>% |
|  | mutate(PointsIfCorrect = 2\*log(2\*(.005\*Confidence+.5))) %>% |
|  | mutate(PointsIfWrong = .5\*log(2\*(-.005\*Confidence+.5))) %>% |
|  | # confidencetransformedwrong = (-.005\*confidencewrong+.5) |
|  | # pointsdeducedwrong = .5\*log(2\*confidencetransformedwrong) |
|  | # mutate(PointsIfCorrect = Confidence/4) %>% |
|  | # mutate(PointsIfWrong = Confidence\*-.5) %>% |
|  | mutate(Time = Sys.time()) %>% |
|  | mutate(Time = as\_datetime(Time)) %>% |
|  | mutate(Name = paste0(input$first, ".",input$last)) |
|  | }) |
|  |  |
|  | sliderValues41 <- reactive({ |
|  | gamedf = data.frame( |
|  | Game = winner$Game[41], |
|  | Away = winner$Away[41], |
|  | Home = winner$Home[41], |
|  | Pick = c(input$game41pick), |
|  | Confidence = c(input$game41confidence), |
|  | stringsAsFactors = FALSE) |
|  | gamedf %>% |
|  | mutate(PointsIfCorrect = 2\*log(2\*(.005\*Confidence+.5))) %>% |
|  | mutate(PointsIfWrong = .5\*log(2\*(-.005\*Confidence+.5))) %>% |
|  | mutate(Time = Sys.time()) %>% |
|  | mutate(Time = as\_datetime(Time)) %>% |
|  | mutate(Name = paste0(input$first, ".",input$last)) |
|  | }) |
|  |  |
|  |  |
|  | if(Sys.time()>as.POSIXct("2017-12-16 13:30:00 EST")){ |
|  | observeEvent(input$uploader1, { |
|  | bowlsgs = bowlsgs %>% |
|  | gs\_add\_row(input = sliderValues1()) |
|  | })} |
|  | observeEvent(input$uploader41, { |
|  | bowlsgs = bowlsgs %>% |
|  | gs\_add\_row(input = sliderValues41()) |
|  | }) |
|  |  |
|  | google <- eventReactive(input$uploader1==FALSE | input$uploader41==TRUE,{ |
|  | helper = gs\_read(bowlsgs) |
|  | helper = |
|  | helper %>% select(Game, Away, Home, Pick, PointsIfCorrect, PointsIfWrong, Name, Time, Confidence) %>% |
|  | mutate(Time = as\_datetime(Time)) %>% |
|  | filter(Pick != 3) %>% |
|  | group\_by(Name, Game) %>% |
|  | filter(Time == max(Time)) %>% |
|  | arrange(desc(Time)) %>% |
|  | left\_join(winner) %>% |
|  | select(Game, Away, Home, Pick, Winner, PointsIfCorrect, PointsIfWrong, Name, Time, Confidence) %>% |
|  | mutate(PointsAwarded = ifelse(Winner!=" ",ifelse(Winner==Pick, PointsIfCorrect, ifelse(Winner!=Pick, PointsIfWrong, 0)),0)) %>% |
|  | ungroup() |
|  |  |
|  | }) |
|  |  |
|  | standingdf <- reactive({ |
|  | google() %>% |
|  | # group\_by(Name) %>% |
|  | group\_by(Name, Game) %>% |
|  | filter(Time == max(Time)) %>% |
|  | ungroup() %>% |
|  | group\_by(Name) %>% |
|  | summarise(TotalPoints = round(sum(PointsAwarded), digits = 4)) %>% |
|  | arrange(desc(TotalPoints)) |
|  | }) |
|  |  |
|  | sliderhelp1 <- reactive({ |
|  | google() %>% |
|  | filter(Name == paste0(input$first,".",input$last)) %>% |
|  | filter(Game == winner$Game[1]) |
|  | }) |
|  | sliderhelp41 <- reactive({ |
|  | google() %>% |
|  | filter(Name == paste0(input$first,".",input$last)) %>% |
|  | filter(Game == winner$Game[41]) |
|  | }) |
|  |  |
|  |  |
|  | ############ Outputs |
|  |  |
|  | output$addsliderbar1 <- renderUI({ |
|  | tagList( |
|  | radioButtons("game1pick", label = "Selection", |
|  | choices = list("Grambling" = winner$Away[1], "North Carolina A&T" = winner$Home[1]), selected = sliderhelp1()[,4]), |
|  | sliderInput("game1confidence", |
|  | label = "Confidence", |
|  | min = 0, max = 99, |
|  | value = as.numeric(sliderhelp1()[,10])), |
|  | if(Sys.time()<as.POSIXct("2017-12-16 13:30:00 EST")){ |
|  | actionButton("uploader1", "Submit Pick")} |
|  | else{ |
|  | helpText("Pick Locked") |
|  | } |
|  | # helpText(paste0("Points if Correct: ", as.numeric(sliderhelp1()[,1]))) |
|  | # helpText(paste0("Points if Correct: ", input$game1confidence)) |
|  | )}) |
|  | output$addsliderbar41 <- renderUI({ |
|  | tagList( |
|  | radioButtons("game41pick", label = "Selection", |
|  | choices = list("Alabama" = winner$Away[41], "Georgia" = winner$Home[41]), selected = sliderhelp41()[,4]), |
|  | sliderInput("game41confidence", |
|  | label = "Confidence", |
|  | min = 0, max = 99, |
|  | value = as.numeric(sliderhelp41()[,10])), |
|  | if(Sys.time()<as.POSIXct("2018-1-8 20:01:00 EST")){ |
|  | actionButton("uploader41", "Submit Pick")} |
|  | else{ |
|  | helpText("Pick Locked") |
|  | } |
|  | )}) |
|  |  |
|  |  |
|  |  |
|  |  |
|  | output$fromgoogledatatable = DT::renderDataTable( |
|  | google(), options = list(pageLength = 1000, search = list(regex = TRUE, caseInsensitive = FALSE)) |
|  | ) |
|  |  |
|  | output$standings = DT::renderDataTable( |
|  | standingdf(), options = list(pageLength = 1000, search = list(regex = TRUE, caseInsensitive = FALSE)) |
|  | ) |
|  |  |
|  | output$instructions <- renderUI({ |
|  | str1 = h3("Step1") |
|  | str1.1 = "Type in your first and last name and submission number.<br/>" |
|  | str1.2 = "- The program will only score your last submission.<br/>" |
|  | str1.3 = "- The Program will remember your last submission if you leave and return later.<br/>" |
|  | str1.4 = "- Yes, it is possible to change other people's submissions, I ask that until I figure out how to prevent this, that you only change Jim's picks...<br/>" |
|  | str2 = h3("Step2") |
|  | str2.1 = "Make your selections for each bowl game. The slider bar represents your confidence in your selection" |
|  | str2.2 = "<br/>Be sure to click submit after every selection" |
|  | str2.3 = "For example:<br/>" |
|  | str2.4 = "- A selection of 100 means you have 100% confidence in the team you selected to win.<br/>" |
|  | str2.5 = "- A selection of 0 means no selection even if you have already clicked a radio button. You will not gain or lose any points. <br/>" |
|  | str3 = h3("Step3") |
|  | str3.1 = "If you wish to make no selection (after previously making a selection) set your confidence to 0.<br/>" |
|  | str3.2 = "Click on the 'Picks' menu to see everyone's selections" |
|  | str4 = h3("\*\*\*Scoring CHANGE \*\*\*") |
|  | str4.1 = "- Credit to COL Horton for a creative scoring mechanism I have implemented. My desire from the start was to reward 'confidence' and harshly penalize 'overconfidence'. <br/>" |
|  | str4.2 = "- Instead of the linear reward (or penalty) of points from 0 to 100% confidence, you will be rewarded (or penalized) at an increasing rate based on your confidence level. <br/>" |
|  | str4.25 = "- I have essentially translated your score into a probability of victory where a 2 point confidence level is a 51% probability of vicotry. A 50 point confidence level is a 75% confidence level and so on... <br/>" |
|  | str4.3 = "- Because the new scoring formula takes the natural log of your confidence - and being wrong on a 100 point confidence translates into the natural log of 0 (-infinity) - I am caping confidence at 99. Try coming back from -Infinity... good luck. <br/>" |
|  | str4.4 = "<br/>The confidence to points translation looks like this: <br/>" |
|  | # str4.5 = - "If correct: 2\*log(2\*(.005\*confidencecorrect+.5))" |
|  | # str4.6 = - "If incorrect: .5\*log(2\*(-.005\*confidencewrong+.5))" |
|  |  |
|  | HTML(paste(str1, str1.1, str1.2, str1.3, str1.4, str2, str2.1, str2.2, str2.3, str2.4, str2.5,str3,str3.1,str3.2, str4, str4.1,str4.2,str4.25, str4.3, str4.4, step = '<br/>')) |
|  | }) |
|  |  |
|  | output$translation = renderPlot({ |
|  | ## if correct |
|  | confidencecorrect = seq(0,100,1) |
|  | confidencetransformedcorrect = (.005\*confidencecorrect+.5) |
|  | pointsearnedcorrect = 2\*log(2\*confidencetransformedcorrect) |
|  | correctawarded = data.frame(confidencecorrect=confidencecorrect, confidencetransformedcorrect = confidencetransformedcorrect, pointsearnedcorrect = pointsearnedcorrect) |
|  |  |
|  | ## if wrong |
|  | confidencewrong = seq(0,100,1) |
|  | confidencetransformedwrong = (-.005\*confidencewrong+.5) |
|  | pointsdeducedwrong = .5\*log(2\*confidencetransformedwrong) |
|  | incorrectdeduced = data.frame(confidencewrong=confidencewrong, confidencetransformedwrong = confidencetransformedwrong, pointsdeducedwrong = pointsdeducedwrong) |
|  |  |
|  | #together |
|  | pointsawarded = data.frame(confidencecorrect=c(confidencecorrect,confidencecorrect), |
|  | pointsawarded = c(pointsearnedcorrect, pointsdeducedwrong), |
|  | Result = c(rep("Correct",101),rep("Incorrect",101)) |
|  | ) |
|  | plot = ggplot()+ |
|  | geom\_line(data = pointsawarded, aes(x=confidencecorrect, y=pointsawarded, color = Result)) + |
|  | xlab("Confidence Level") + ylab("Points Awarded") + ggtitle("Points Scaling") |
|  | return(plot) |
|  | }) |
|  |  |
|  |  |
|  |  |
|  | # output$outlook = renderPlot({ |
|  | # google2 = google() %>% |
|  | # group\_by(Name) %>% |
|  | # # filter(Winner==" ") %>% |
|  | # mutate(Totalpoints = sum(PointsAwarded)) %>% |
|  | # mutate(PointsAwarded = ifelse(Winner!=" ",maxpoints = sum(PointsIfCorrect), minpoints = sum(PointsIfWrong), interval = maxpoints-minpoints,0)) %>% |
|  | # arrange(desc(Totalpoints)) |
|  | # |
|  | # # factor(Name, levels = Name[order(Totalpoints)]) |
|  | # plot2 = ggplot(data=google2, aes(y = Totalpoints, x = Name, color = Name)) + |
|  | # geom\_point() + |
|  | # geom\_errorbar(aes(ymin = minpoints, ymax = maxpoints)) + |
|  | # ggtitle("Standings along with Max Possible and Min Possible Points based on current selections") |
|  | # |
|  | # # x$name <- factor(x$name, levels = x$name[order(x$val)]) |
|  | # |
|  | # return(plot2) |
|  | # }) |
|  |  |
|  | output$outlook = renderPlot({ |
|  | google2 = google() %>% |
|  | group\_by(Name) %>% |
|  | # filter(Winner==" ") %>% |
|  | summarise(Totalpoints = sum(PointsAwarded)) %>% |
|  | # mutate(maxpoints = sum(PointsIfCorrect), minpoints = sum(PointsIfWrong), interval = maxpoints-minpoints, Totalpoints = sum(PointsAwarded)) %>% |
|  | arrange(desc(Totalpoints)) %>% |
|  | select(Name, Totalpoints) |
|  | google3 = google() %>% |
|  | filter(Winner==" ") %>% |
|  | group\_by(Name) %>% |
|  | summarise(min=sum(PointsIfWrong), max = sum(PointsIfCorrect)) |
|  | google4 = google2 %>% |
|  | left\_join(google3) %>% |
|  | mutate(minpoints=Totalpoints+min, maxpoints=Totalpoints+max) |
|  | plot2 = ggplot(data=google4, aes(y = Totalpoints, x = Name, color = Name)) + |
|  | geom\_point() + |
|  | geom\_errorbar(aes(ymin = minpoints, ymax = maxpoints)) + |
|  | ggtitle("Standings along with Max Possible and Min Possible Points based on current selections") + |
|  | theme(axis.text.x = element\_text(angle = 45, hjust = 1)) |
|  | return(plot2) |
|  | }) |
|  |  |
|  | } |
|  |  |
|  | shinyApp(ui, server) |