PharmaSUG China 2021 - Paper

BIP: A Powerful Biomarker Interactive Profiler based on R Shiny APP

Miao Yu, Chun Yi, BeiGene Co., Ltd.

ABSTRACT

Currently, pharmaceutical and biotech industry is experiencing transformation. The requirement of Statistical Programmer is also changing along with the environment because the emerging needs of automation based on different programming language. Although SAS remains to be the most critical programming tool in industry, R, an open-source programming language tailored for statistical analysis and data visualization, is rising as a competitive alternative. In this paper, we would like to introduce an R shiny APP called Biomarker Interactive Profiler (BIP). Biomarker data analysis is essential for our drug development. However, before the launching of BIP, the collaboration between Biomarker Scientist, Statistician and Statistical Programmer is time-consuming and lack of efficiency. This APP enables Biomarker Scientists to do exploration directly. Also, it provides a solution to data access control, visualization, statistical analysis at the same time within same platform. This paper introduces the background and usage of BIP and also the programming techniques to build up a complex R Shiny APP like BIP. In addition, the limitation and potential improvements are discussed.

INTRODUCTION

In pharmaceutical and biotech industry, SAS, well documented and validated, has played a critical role especially in regulatory submission. As a statistical programmer, it's essential for us to utilize SAS to support all kinds of data analysis needs from other functions, covering from pre-clinical to post-market. Among those requests, some are quite standard so that standard programs/outputs can be preparable and reusable, while others are not and needs extra effort to meet the demands. Especially for pre-clinical analysis, they're often for exploratory purpose and its analysis varies under different circumstances.

From past experience of supporting all kinds of requests, more and more people started to search for more efficient ways of fulfilling the needs, by trying different programming languages, delivery methods. R started to gain popularity gradually in our industry since it's a common language not only among biostatisticians but also among some other functions like Bioinformatics, Clinical pharmacokinetics. In addition, R is very good at data visualization with well-established packages like ggplot2.In addition, there is also a package called R shiny, a package to build interactive web apps straight from R. It provides us a very good method to build interactive platforms, to better serve the data interpreters. Based on R's strength in visualization and R shiny's capability of building web apps, we can surely provide more and more better solutions to our daily work.

Biomarker Interactive Profiler (BIP) can be a good attempt and practice to improve efficiency by using R shiny. In this paper, we'll do a brief introduction on BIP.

BACKGROUND

Biomarker scientists usually do a lot of research and exploration to detect correlation between different biomarkers and efficacy endpoints, which may require statistical programmers to provide supporting Table, Listing and Figures (TLFs). Given the huge amount of candidate biomarkers, over 1000 gene expressions, and various exploration statistical methods, it's hard for us to identify the best way to support and meanwhile minimize our workloads. Before BIP's launching, we didn't have a formal delivery process regarding this kind of request. Therefore, in some studies, programmers had generated more than 100 TLFs to facilitate biomarker exploration. Among all the TLFs, most are same plot, like KM plot, under different subgroups of patients. In order to save programming resource and meanwhile provide biomarker scientist strong support.

Gained inspiration from GEPIA developed by Peking University, we proposed that we can find a new way rather than TLFs to fulfill the gap and R shiny APPs can be a very good attempt. After BIP's launching and earning some compliments, we started to develop other R shiny APPs serving different purpose. They turned out to be very useful and efficient in different delivery activities.

BIP, as an interactive web app, it has below advantages,

- Applies data access control of confidential efficacy data for biomarker scientists
- Allows biomarker scientists to interact with data and plots freely
- Provides different data visualization choices, including box plot, ROC plot, KM plot and forest plot

In following sections, all the functionalities of BIP will be introduced in detail.

R-

APP STRUCTURE

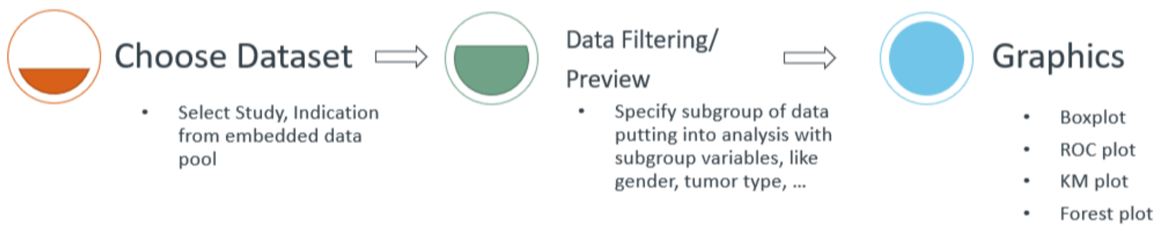
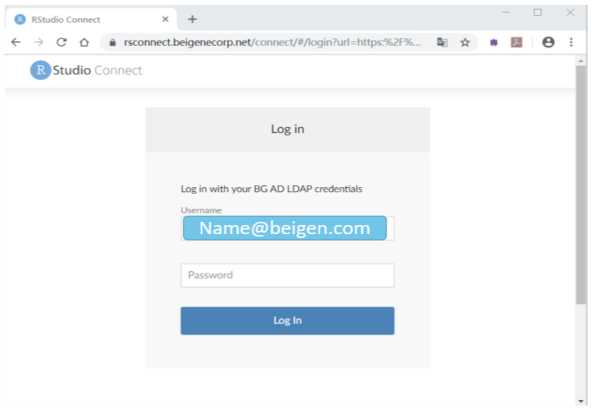

Choose Dataset
Select Study, Indication from embedded data pool
Data Filtering/ Preview
Specify subgroup of data putting into analysis with subgroup variables, like gender, tumor type, ...
Graphics
Boxplot
ROC plot
KM plot
Forest plot

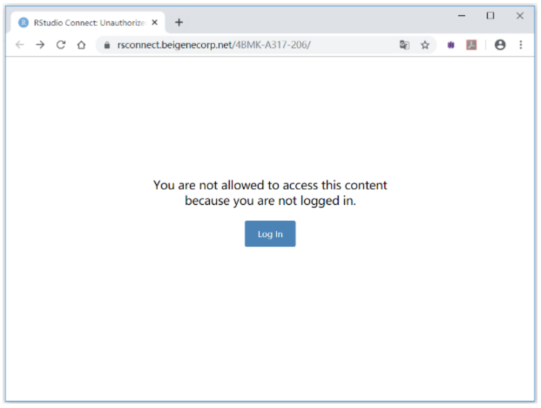
Figure 1. The APP structures

As shown in Figure 1, BIP can be divided into 3 main parts and the data go with this flow. Also, users should follow this order to do explore data. The detailed usage will be introduced in next section.

USAGE

STEP 1: LOGIN


RStudio Connect
rsconnect.beigenecorp.net/connect/#/login?url=https%2F%...
RStudio Connect
Log in
Log in with your BG AD LDAP credentials
Username
Name@beigen.com
Password
Log In

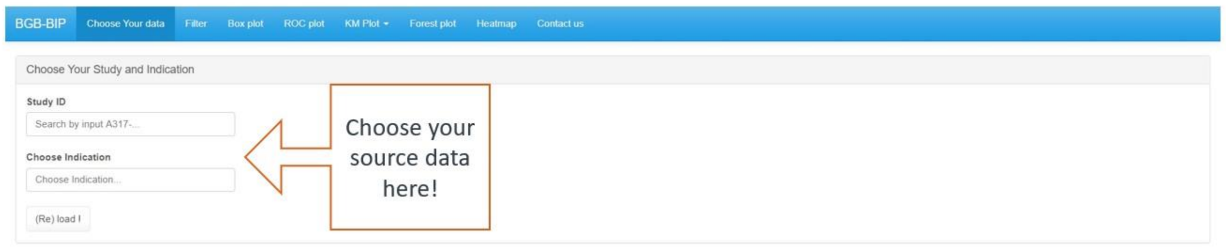

RStudio Connect: Unauthoriz
rsconnect.beigenecorp.net/4BMK-A317-206/
Log In

You are not allowed to access this content because you are not logged in.

Display 1. Login Page

Users who have granted access to this APP can login via their company credentials, otherwise, you'll not be able to access any data embedded.

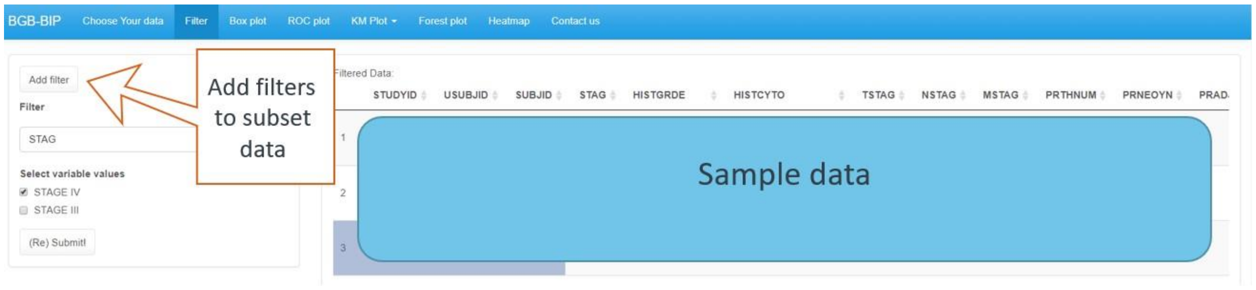
STEP 2: CHOOSE STUDY DATA



Display 2. Choose Study Data Panel

Clinical and biomarker data of all available study has been embedded in this APP. However, users can not see all the study since we have applied access control to individual study. Users will have to apply access to individual study first then to see the study data on the APP. To extract the interested study data from database, users can type the Study ID to choose the studies. After choosing studies, they can further choose interested indication. Then by clicking "(Re) Submit!" button, the data will be ready in a minute with a pop-up notification.

STEP 3: DATA FILTERING AND DISPLAY



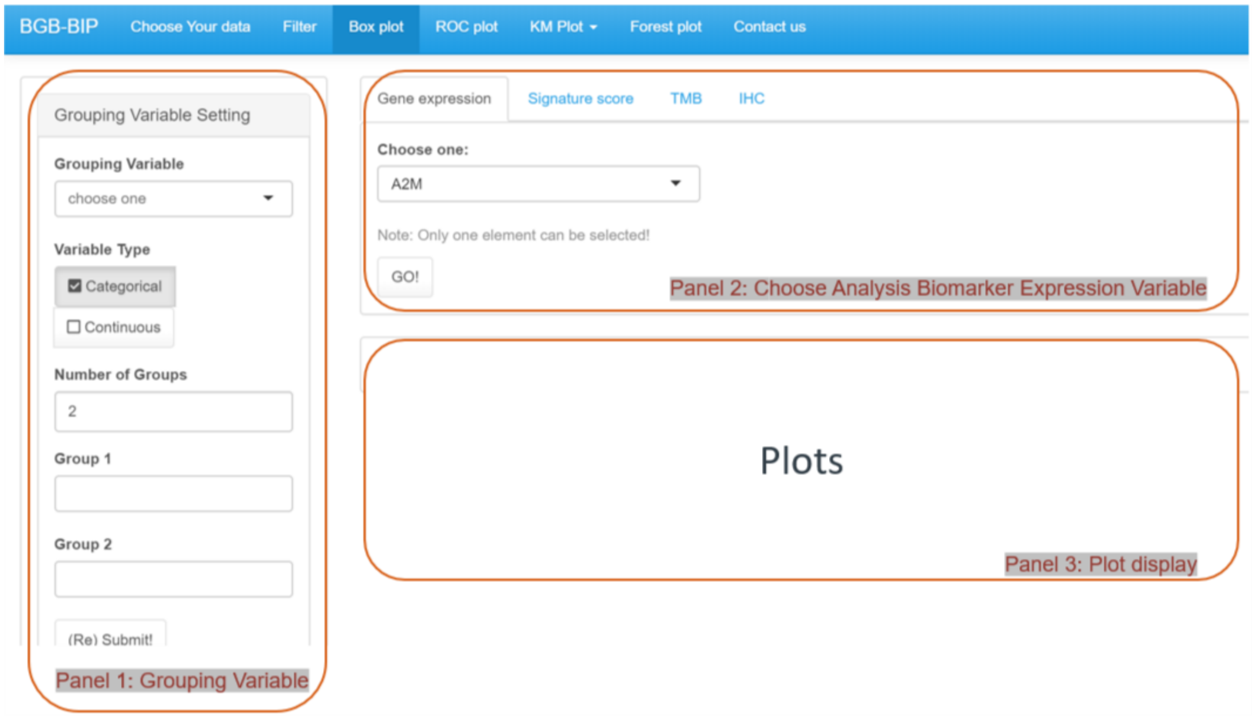
Display 3. Data Filtering Panel

As shown in Display 3, all the baseline sub-group variables identified by biomarker scientist can be added by clicking "Add filter" Button, like age, sex, baseline ECOG and so on. It allows user to further identify subgroup of patients they're interested. For example, if user just want to explore the biomarker expressions in female, stage IV patients with prior medical history. User can add 3 filters "SEX", "STAG" and "MH" then check the corresponding checkbox. Then the data will be further filtered and shown on then right for user to review. After reviewing, data will be passed on to plotting panels.

STEP 4: PLOTTING

In Display 4, it shows the page of box plot. In Panel 1, you will need to identify the grouping variable. it gives users the flexibility to choose Categorical or Continuous variable and also choose how many groups you want to display in plot.

In Panel 2, user will choose the biomarker they want to explore as response variable of boxplot. Different types of Biomarkers are embedded in this APP, Gene Expression, Signature Score, TMB and IHC. After correctly identifying the interested biomarkers, you can click 'GO!' in Panel 2 and then the boxplot will be shown in Panel 3.



Display 4. Plotting Panel

CODING TECHNIQUE HIGHLIGHT

R SHINY MODULES

Why Using R shiny Modules

Many people have already been familiar with R shiny and have experience developing complex shiny APPs. However, if you're not familiar with Shiny Modules, you may find it difficult to manage and standardize your APPs when it goes more and more complicated. Therefore, I'd like to introduce this solution in this paper. A Shiny Module, to put it simple, is a wrapped package of a UI and SERVER functions. A complex R shiny APP can be divided into many small modules. Also, a bunch of small modules can build up to a large APP. All the modules developed can be saved into a library and reused in future. In conclusion, here are some of the advantages of applying R shiny Modules in complex shiny APPs.

- if APPs are becoming more and more complex, it's easy to organize the code by using modules.
- Panels with same functionality can share the same modules.
- Flexibility for future use
- Easy to debug by pieces. You can only maintain the modules, instead of the whole app.

Modules are especially helpful to BIP development. There is a very straightforward reason. When planning for the APP structure, we found that all the pages for plotting (including ROC, boxplot, KM plot, forest plot) had similar layout of user interface as shown in Display 4. Therefore, using modules can help to avoid duplicate coding. Also, all the modules behind the 3 panels can be used cross different plotting pages.

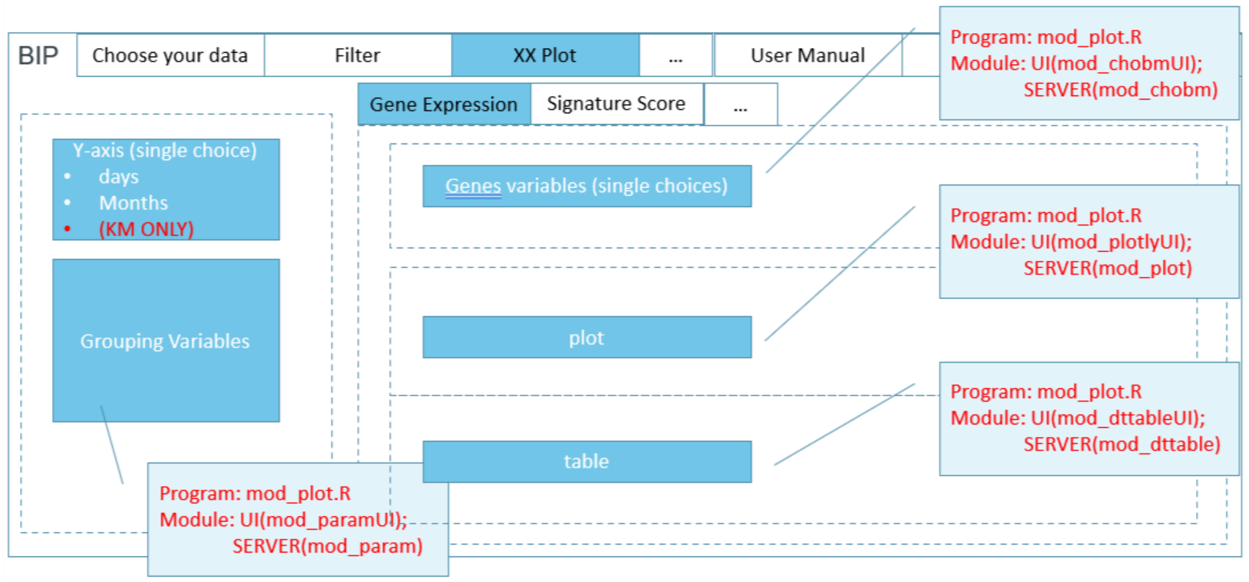


Figure 2. R shiny Modules of plotting pages

In Figure 2, we can see that how we plan the layout and modules behind all the panels.

Application of R shiny Modules in BIP

As shown in below Figure 3, we showed all the modules we developed to support BIP. It's clear to see that the APP can be well organized by modularizing all functional parts. The huge app.R R scripts can be simplified.

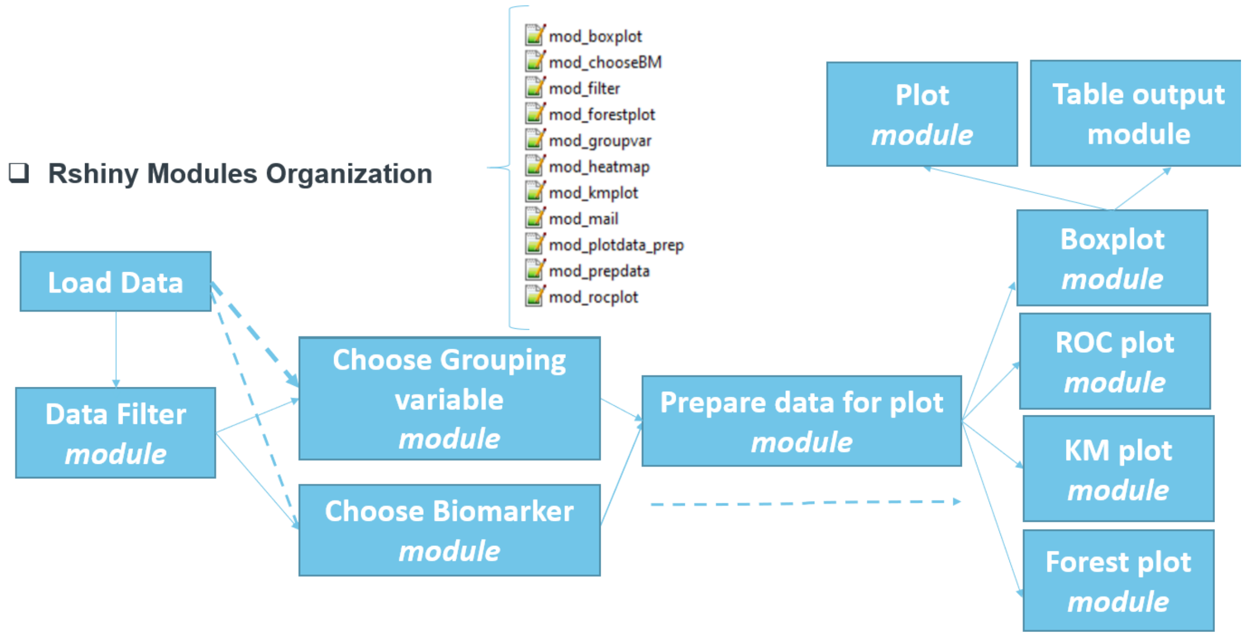
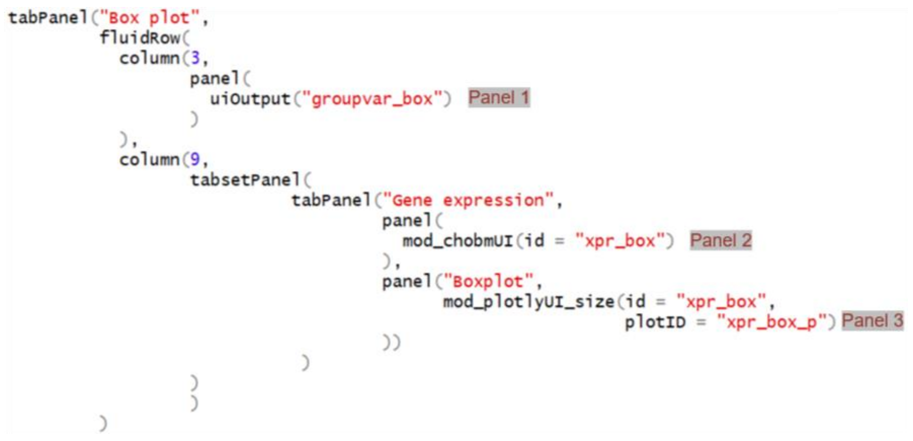


Figure 3. Rshiny Modules of BIP

R shiny Module Examples

As shown in Figure 2, these 3 panels can be developed into independent shiny modules. Below is the corresponding part of UI function of this BIP:

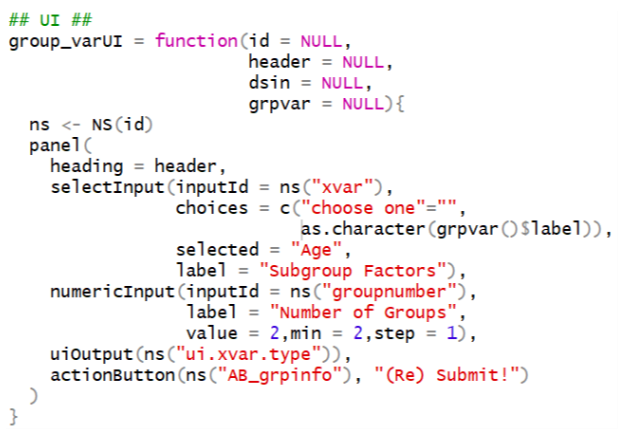


```
tabPanel("Box plot",
        fluidRow(
          column(3,
                 panel(
                   uiOutput("groupvar_box")  Panel 1
                 )
          ),
          column(9,
                 tabsetPanel(
                           tabPanel("Gene expression",
                                    panel(
                                      mod_chobmUI(id = "xpr_box")  Panel 2
                                    ),
                                    panel("Boxplot",
                                          mod_plotlyUI_size(id = "xpr_box",
                                                            plotID = "xpr_box_p") Panel 3
                                    ))
                           )
                 )
                 )
        )
```

As we can see from the R codes, UI function is greatly simplified by utilizing R shiny Module. In addition, by specifying different ID in these modules, they can be utilized cross webpages and even cross APPs.

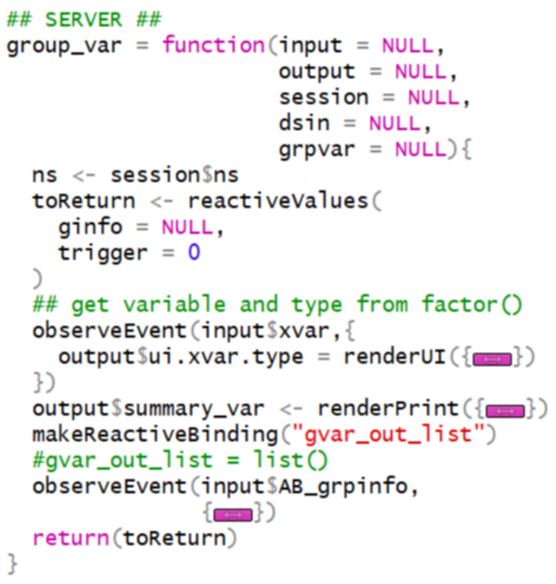
Here is an example of an Rshiny Module, “Choose Subgroup Variable” Panel.

UI function:



```
## UI ##
group_varUI = function(id = NULL,
                       header = NULL,
                       dsin = NULL,
                       grpvar = NULL){
  ns <- NS(id)
  panel(
    heading = header,
    selectInput(inputId = ns("xvar"),
                choices = c("choose one"="",
                            as.character(grpvar()$label)),
                selected = "Age",
                label = "Subgroup Factors"),
    numericInput(inputId = ns("groupnumber"),
                 label = "Number of Groups",
                 value = 2,min = 2,step = 1),
    uiOutput(ns("ui.xvar.type")),
    actionButton(ns("AB_grpinfo"), "(Re) Submit!")
  )
}
```

SERVER function:



```
## SERVER ##
group_var = function(input = NULL,
                     output = NULL,
                     session = NULL,
                     dsin = NULL,
                     grpvar = NULL){
  ns <- session$ns
  toReturn <- reactiveValues(
    ginfo = NULL,
    trigger = 0
  )
  ## get variable and type from factor()
  observeEvent(input$xvar,{
    output$ui.xvar.type = renderUI({▬})
  })
  output$summary_var <- renderPrint({▬})
  makeReactiveBinding("gvar_out_list")
  #gvar_out_list = list()
  observeEvent(input$AB_grpinfo,
               {▬})
  return(toReturn)
}
```

PLOTLY

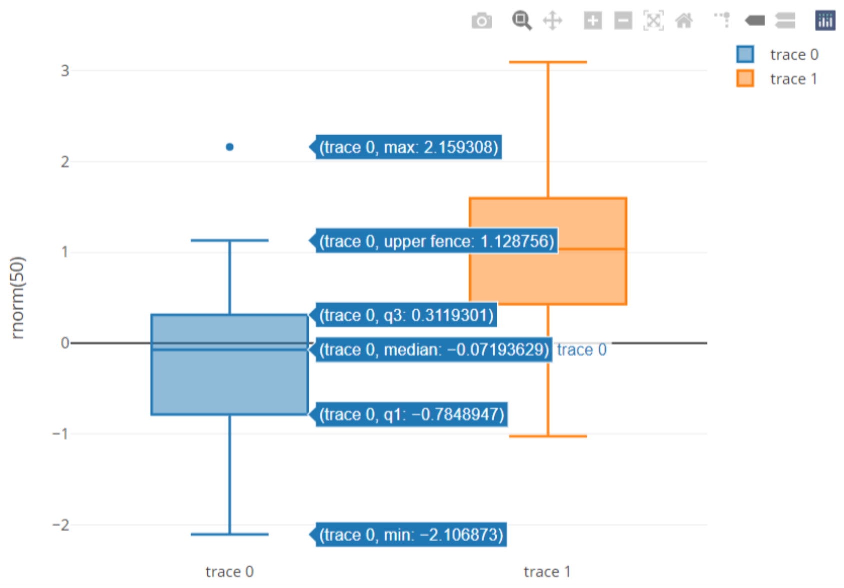


Figure 4. Box plot using plotly package

Using plotly can add more flexibility to our R shiny APPs since it allows us to interact with plot. As shown in below Figure 4, the key information will appear when moving mouse cursor over interested data points. Also, by clicking the buttons on the right upper of plot, we can download, zoom in/out, and manipulate the plot to fulfill users' exploratory needs.

DISCUSSION

LIMITATION

First limitation is about the input datasets. Although BIP can provide flexibility to data exploration, it still runs on the basis of well-formatted datasets, which requires extra efforts to prepare. Another limitation is that the default analysis method, choices of plot is limited. Third limitation is that it often takes months to do some updates on app because we need to do intensive tests and QC before launching to users.

FUTURE IMPROVEMENT

In future, we'd like to add more analysis methods, like correlation analysis. In addition, we will further tailor our plots to make it ready for publish. Therefore, once the data is decided to be shown in publications, the plots are ready at any time.

CONCLUSION

Programming, as an important supporting function in R&D, we have been making various deliveries in all kinds of formats. BIP is one of good attempts among all the innovative way of supporting other functions. It has been intensively used by Bioinformatics department and gains many compliments. Starting from BIP, we will seek for more opportunities to improve efficiency and better ways to collaborate in future.

ACKNOWLEDGMENTS

Sincerely thanks to Amanda and our BIP working group, including Xiaoyuan Huang, Shuo Zhang, Kailu Song for their help and support.

RECOMMENDED READING

- *http://gepia.cancer-pku.cn/*
- *https://shiny.rstudio.com/*
- *https://shiny.rstudio.com/articles/modules.html*
- *https://plotly-r.com/preface.html#getting-help-and-learning-more*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

<Miao Yu>
<BeiGene Inc.>
<miao.yu@beigene.com >

<Chun Yi>
<BeiGene Inc.>
<chun.yi@beigene.com >