

Welcome to Shiny Apps

- What is Shiny?
 - - A web application framework for R.
 - - Enables building interactive web apps using R code.
- Why Learn Shiny?
 - - Create data dashboards, visualizations, and tools.
 - - Share data insights interactively with non-programmers.

Components of a Shiny App

- **UI (User Interface):** Defines the layout and appearance.
- **Server:** Contains the logic to process inputs and generate outputs.
- **App:** Combines the UI and Server.

```
library(shiny)

ui <- fluidPage(
  # UI elements
)

server <- function(input, output) {
  # Server logic
}

shinyApp(ui = ui, server = server)
```

HTS
QC
QHTS I
QHTS II
Results I
Results II
Results III
Target DRC

Heatmap data type

AUC

Custom Samples:

Search sample(s)

Custom Drugs:

Search drug(s)

Subset drugs by

- ☒ None
- ☐ Pathway
- ☐ Targets
- ☐ Mechanism of Action
- ☐ Clinical Phase
- ☐ Information

AUC

1.5

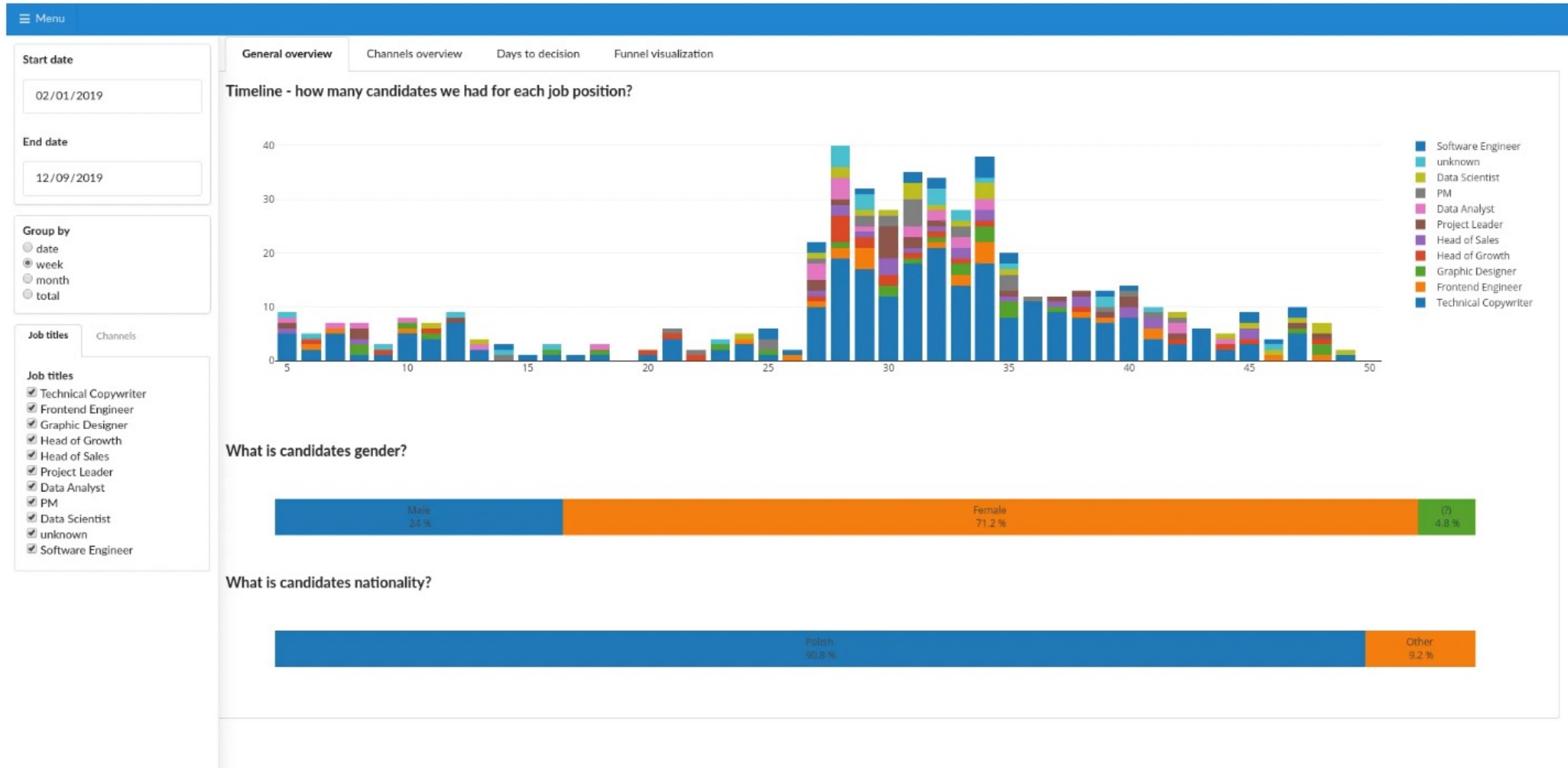
1

0.5

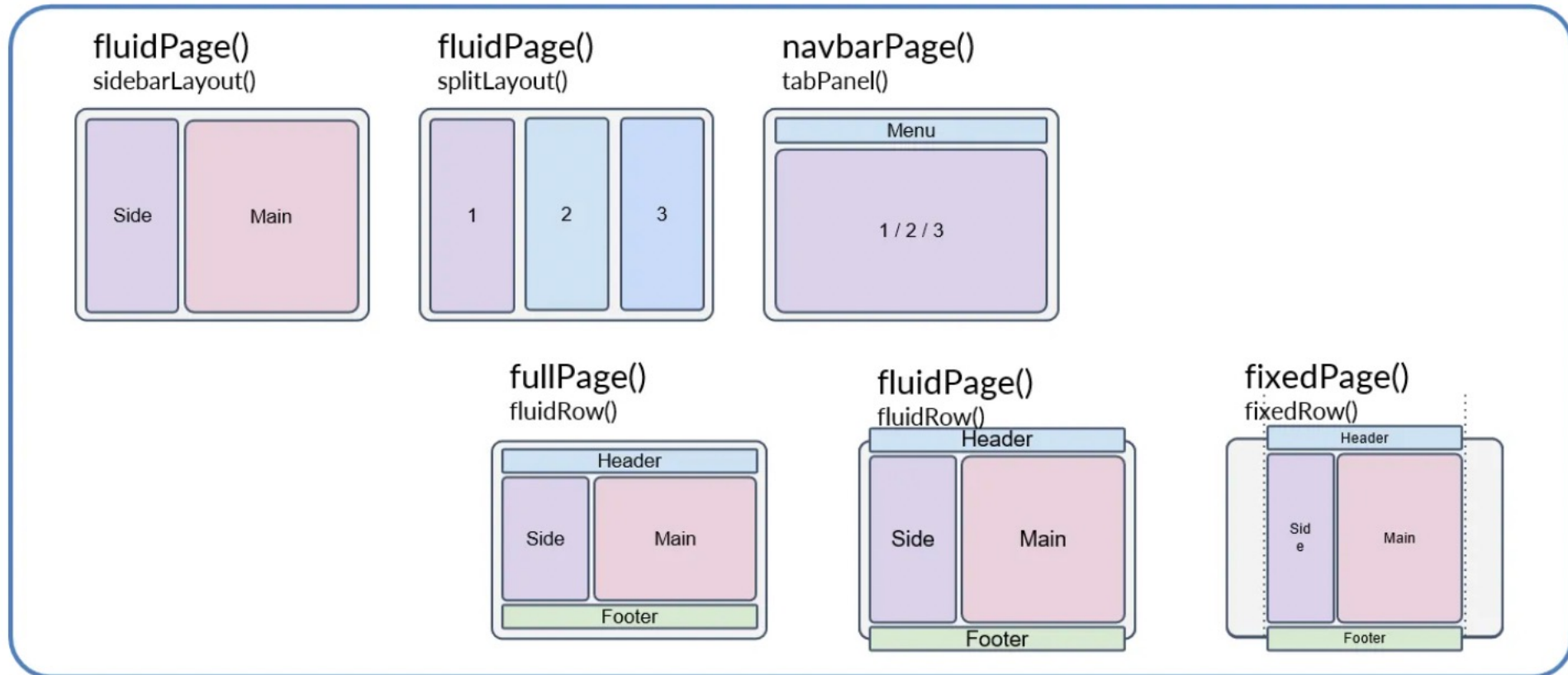
0

BAY-67-2243	NVP-BG-1398	R-428	amorphine	sapiratinib	mycophenolic acid	picosulfide	batimastat	quercetin	ponatinib	axitinib	quizartinib	erlotinib	AMG-330	anagrelide	dasatinib	napabucic	P-3-24781	roscovit	adefovir-dipivoxil	Quinacrine 2HCl	mycophenolate-mofetil	saracatinib	Pacitinib (SB1518)	uprosertib	pellocicb	indirubin	semaxanib	selumetinib	brutinib	MEK162	afatinib	LCs inhibitor	insulin	CEP-37440	brigatinib	giletritinib	foretinib	AXL1717	MK-1775	centinib	belinostat	midostaurin	amscine	irinotecan	JQ1-(+)	lvartinib	cladribine	clotarfamine	cytarabine	baradoxione	lestaurtinib	JIB04	dasatinib	pevonedistat	abemaciclib	barasertib-HOPA	daunorubicin	alisertib	taseleisib	cogantib	BAY-1895344	BAY-1895344 HCl	Halofuginone, HT-100, NSC-7132	defactinib	selinexor	codonatinib	daconitinib	netatinib	poziotinib	everolimus	NVP-BE2235	NVP-AU1922	ganefesib	SB-225002	coaticic	Mivebresib	topotecan	gemcitabine	fanestipmycin	volasertib	panobinostat	idarubicin	anamorelin	MLN9708	lebelopone	vincristine	vinblastine	flanesib	apothione-b	docetaxel	vinorelbine	dinacitib	Triptolide	dactinomycin	Chaetocin, Chetocin	toridepsin	torfezomib	carfilzomib
-------------	-------------	-------	-----------	-------------	-------------------	-------------	------------	-----------	-----------	----------	-------------	-----------	---------	------------	-----------	-----------	-----------	----------	--------------------	-----------------	-----------------------	-------------	--------------------	------------	-----------	-----------	-----------	-------------	----------	--------	----------	---------------	---------	-----------	------------	--------------	-----------	---------	---------	----------	------------	-------------	---------	------------	---------	-----------	------------	--------------	------------	-------------	--------------	-------	-----------	--------------	-------------	-----------------	--------------	-----------	------------	----------	-------------	-----------------	--------------------------------	------------	-----------	-------------	-------------	-----------	------------	------------	------------	------------	-----------	-----------	----------	------------	-----------	-------------	---------------	------------	--------------	------------	------------	---------	------------	-------------	-------------	----------	-------------	-----------	-------------	-----------	------------	--------------	---------------------	------------	------------	-------------

Shiny App Layout - UI



Shiny App Layout - UI



UI Components

- **Layout Functions:**

- **Create a flexible layout**
fluidPage()

- **Define structured layouts**
 - sidebarLayout()
 - mainPanel()
 - sidebarPanel()

- **Input Functions:**

- sliderInput(), textInput(), selectInput(), etc.

- **Output Functions:**

- textOutput(), plotOutput(), tableOutput(), etc.

```
ui <- fluidPage(  
  
  titlePanel("Simple App"),  
  
  sidebarLayout(  
  
    sidebarPanel( selectInput("var", "Choose variable:",  
                             choices = c("A", "B", "C"))  
  ),  
  
  mainPanel( plotOutput("plot")  
  )  
)  
)
```

Server Logic

- How Server Works:
 - Reacts to user inputs.
 - Generates outputs dynamically.
- Key Functions:
 - `renderText()`, `renderPlot()`,
`renderTable()`

Example Server Code:

```
server <- function(input, output) {  
  output$plot <- renderPlot({  
    plot(1:10, main = input$var)  
  })  
}
```

UI – Server connection

UI

```
ui <- fluidPage(  
  titlePanel("Simple App"),  
  sidebarLayout(  
    sidebarPanel( selectInput("var", "Choose  
variable:",  
                           choices = c("A", "B", "C"))  
  ),  
  mainPanel( plotOutput("plot")  
)  
)  
)
```

server

```
server <- function(input, output) {  
  output$plot <- renderPlot({  
    plot(1:10, main = input$var)  
  })  
}
```


Circular format

```
selectInput(  
  inputId = "colorVariable",  
  label = "Please select variable to color point plot Fig1",  
  choices = colnames(bCancer),  
  selected = colnames(bCancer)[3],  
  multiple = F  
  
  plotOutput("pointPlot"))
```

UI

Circular format

UI

```
selectInput(  
  inputId = "colorVariable",  
  label = "Please select variable to color point plot Fig1",  
  choices = colnames(bCancer),  
  selected = colnames(bCancer)[3],  
  multiple = F  
)  
  
plotOutput("pointPlot")
```

server

```
output$pointPlot <- renderPlot({  
  xVariable = "concav" # Create these as constants right now  
  yVariable = "area"  
  
  bCancer %>%  
    ggplot(aes_string(x = xVariable, y = yVariable, color = input$colorVariable)) + # Change color to  
the input value  
    geom_point() +  
    geom_smooth(method = "lm") +  
    ggtitle("Fig1: Point plot") # Add title to identify which input it matches to  
})
```

Circular format

UI

```
selectInput(  
  inputId = "colorVariable",  
  label = "Please select variable to color point plot Fig1",  
  choices = colnames(bCancer),  
  selected = colnames(bCancer)[3],  
  multiple = F
```

```
plotOutput("pointPlot"))
```

server

```
output$pointPlot <- renderPlot({  
  xVariable = "concav" # Create these as constants right now  
  yVariable = "area"  
  
  bCancer %>%  
    ggplot(aes_string(x = xVariable, y = yVariable, color = input$colorVariable)) + # Change color to  
    the input value  
    geom_point() +  
    geom_smooth(method = "lm") +  
    ggtitle("Fig1: Point plot") # Add title to identify which input it matches to  
})
```

Circular format

UI

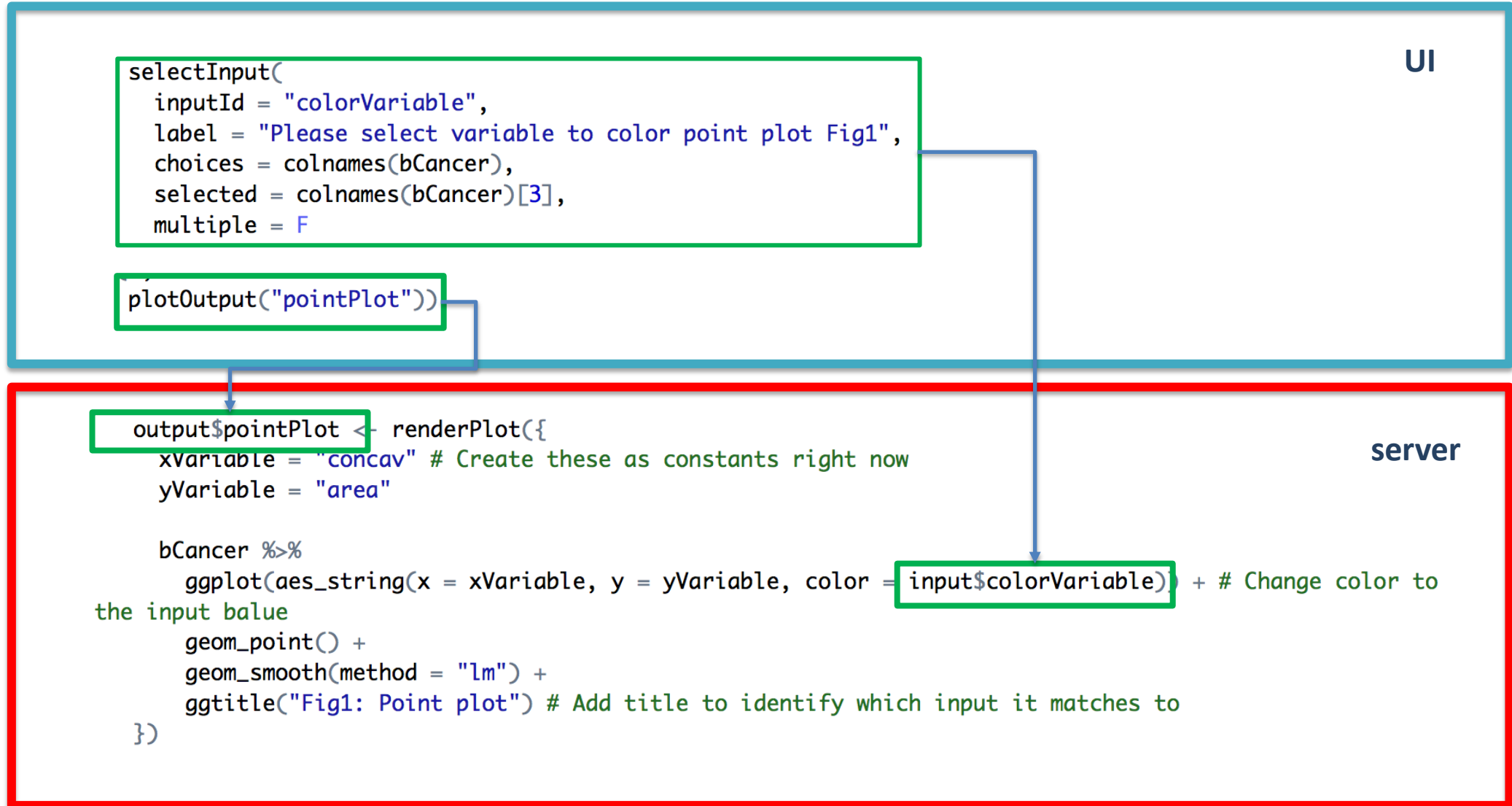
```
selectInput(  
  inputId = "colorVariable",  
  label = "Please select variable to color point plot Fig1",  
  choices = colnames(bCancer),  
  selected = colnames(bCancer)[3],  
  multiple = F  
)
```

```
plotOutput("pointPlot"))
```

server

```
output$pointPlot <- renderPlot({  
  xVariable = "concav" # Create these as constants right now  
  yVariable = "area"  
  
  bCancer %>%  
    ggplot(aes_string(x = xVariable, y = yVariable, color = input$colorVariable)) + # Change color to  
the input value  
    geom_point() +  
    geom_smooth(method = "lm") +  
    ggtitle("Fig1: Point plot") # Add title to identify which input it matches to  
})
```

Circular format



Adding Interactivity

Reactive Expressions:

- Use `reactive()` for reusable calculations.

```
reactiveVal <- reactive({  
  input$num * 2  
})
```

Observe for Side Effects:

- Use `observe()` to perform actions without generating outputs.

```
observe({  
  print(input$num)  
})
```

When to Use Reactivity

- You should use reactivity whenever:
 - 1. Dynamic Output Is Needed:** The content of your app (e.g., plots, tables, text) needs to change based on user inputs.
 - 2. Dependent Computations Exist:** You want to compute something once and reuse the result across multiple outputs.
 - 3. Efficient Updates Are Required:** You want to avoid recalculating unchanged components unnecessarily.
- Examples:
 - Displaying a plot that updates when a user changes a dropdown value.
 - Filtering a table based on user-selected criteria.
 - Dynamically displaying text based on slider input.

LET'S MAKE A SHINY APP!