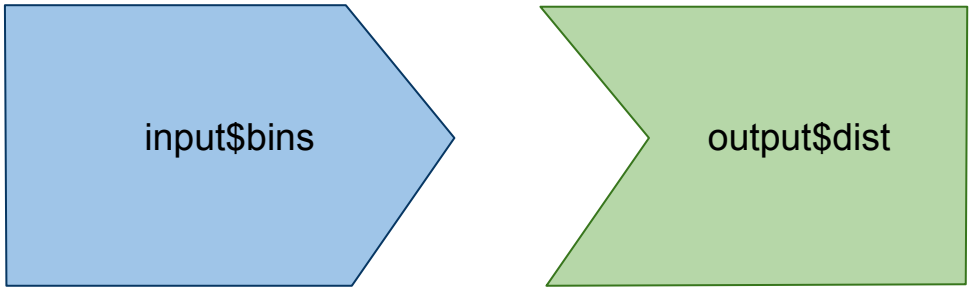
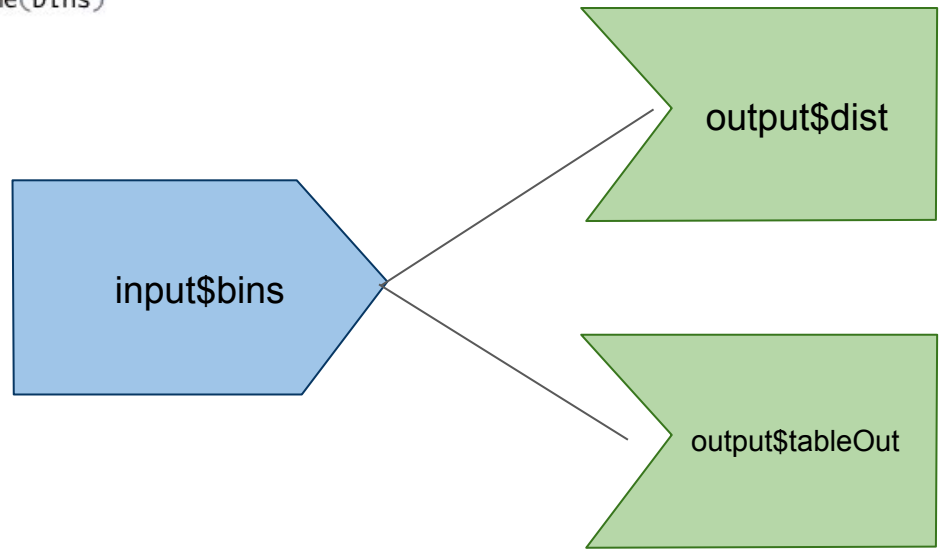
Reactivity

Shiny App

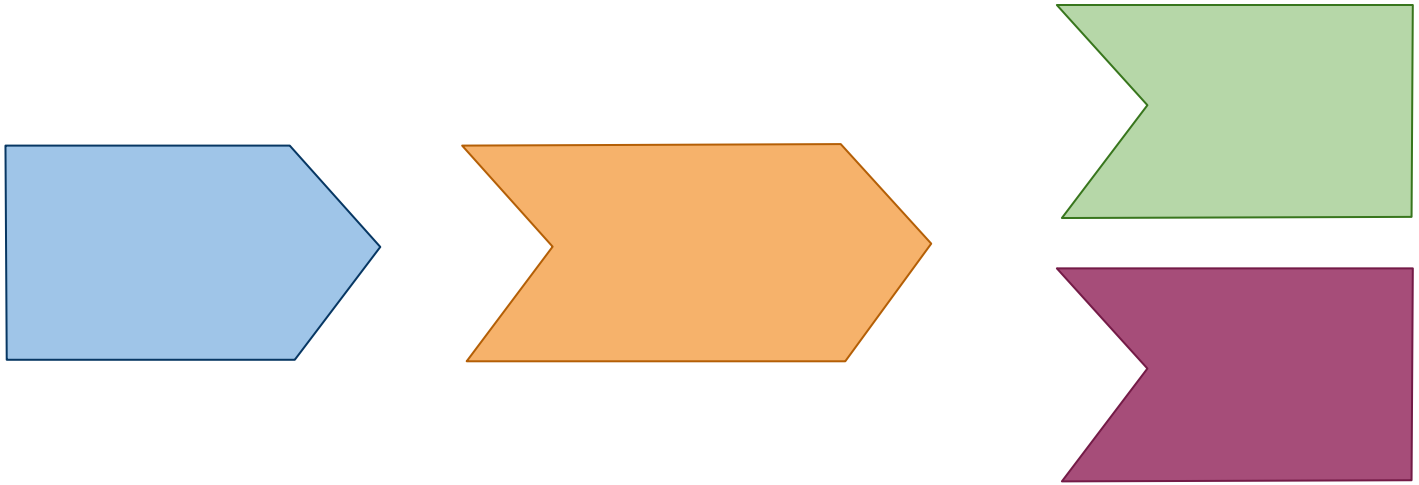
```
1
2  library(shiny)
3
4  shinyServer(function(input, output) {
5
6      output$distPlot <- renderPlot({
7          x    <- faithful[, 2]
8          bins <- seq(min(x), max(x), length.out = input$bins + 1)
9          hist(x, breaks = bins, col = 'darkgray', border = 'white')
10     })
11 })
```



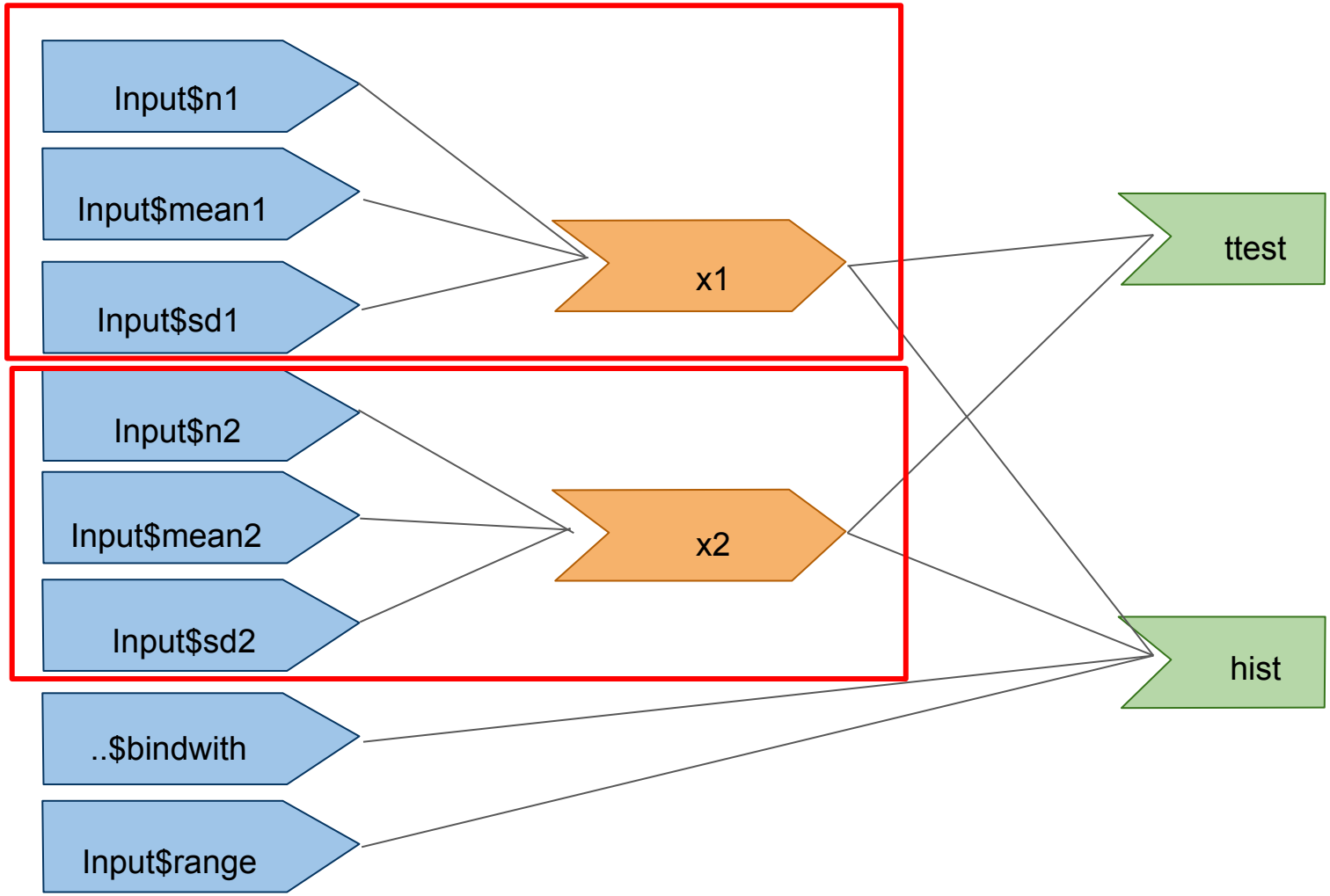
```
1
2  library(shiny)
3
4  shinyServer(function(input, output) {
5
6      output$distPlot <- renderPlot({
7          x    <- faithful[, 2]
8          bins <- seq(min(x), max(x), length.out = input$bins + 1)
9          hist(x, breaks = bins, col = 'darkgray', border = 'white')
10     })
11
12
13     output$tableOut <- renderTable({
14     if(input$bins!=0){
15         as.data.frame(bins)
16     }else{NULL}
17     })
18
19
20 })
```



```r
fib <- function(n) ifelse(n<3, 1, fib(n-1)+fib(n-2))

server <- function(input, output) {
  currentFib          <- reactive({ fib(as.numeric(input$n)) })

  output$nthValue     <- renderText({ currentFib() })
  output$nthValueInv  <- renderText({ 1 / currentFib() })
}
```



```
server <- function(input, output, session) {
  x1 <- reactive(rnorm(input$n1, input$mean1, input$sd1))
  x2 <- reactive(rnorm(input$n2, input$mean2, input$sd2))

  output$hist <- renderPlot({
    histogram(x1(), x2(), binwidth = input$binwidth, xlim = input$range)
  })

  output$ttest <- renderText({
    t_test(x1(), x2())
  })
}
```

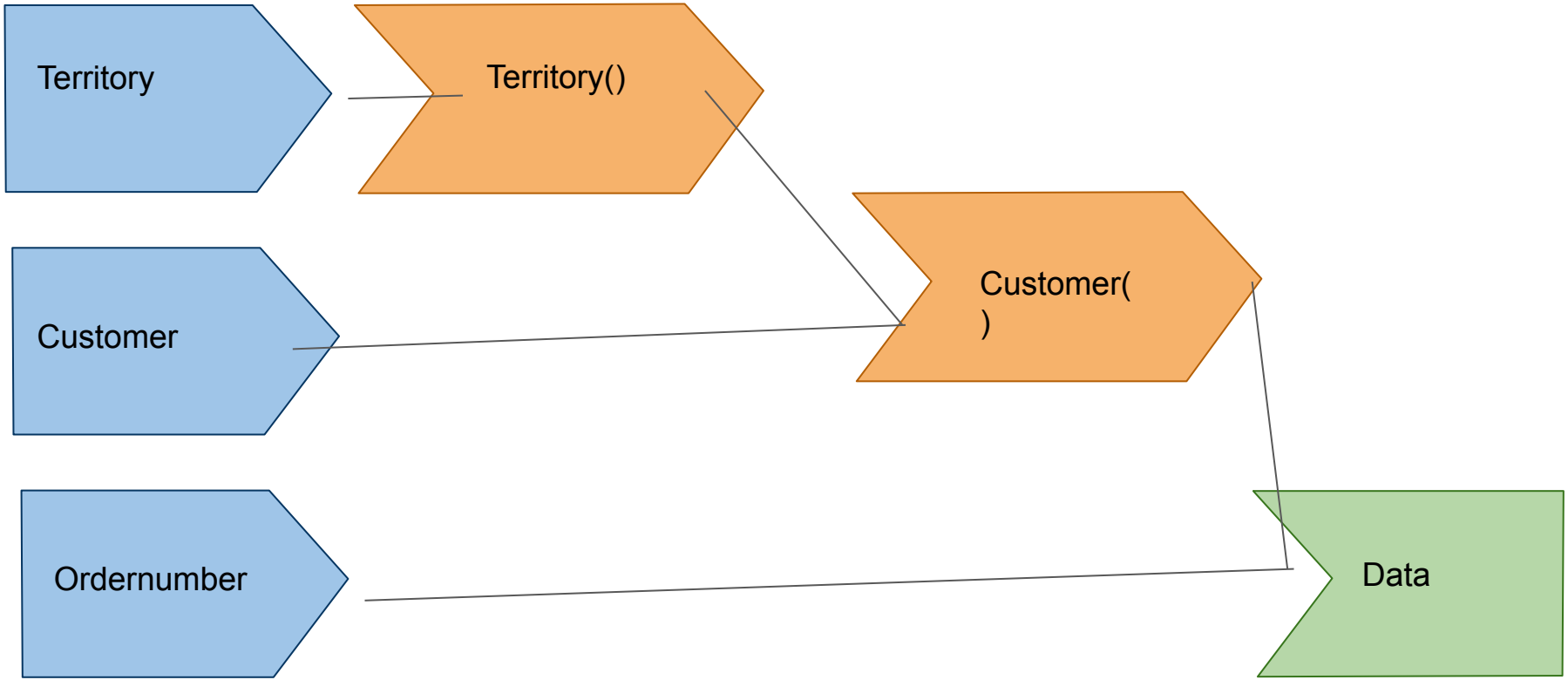

Input$n1
Input$mean1
Input$sd1
x1
Input$n2
Input$mean2
Input$sd2
x2
..$bindwith
Input$range
ttest
hist

Updating UI

```r
ui <- fluidPage(
  sliderInput("x1", "x1", 0, min = -10, max = 10),
  sliderInput("x2", "x2", 0, min = -10, max = 10),
  sliderInput("x3", "x3", 0, min = -10, max = 10),
  actionButton("reset", "Reset")
)

server <- function(input, output, session) {
  observeEvent(input$reset, {
    updateNumericInput(session, "x1", value = 0)
    updateNumericInput(session, "x2", value = 0)
    updateNumericInput(session, "x3", value = 0)
  })
}
```

```r
ui <- fluidPage(
  numericInput("n", "Simulations", 10),
  actionButton("simulate", "Simulate")
)

server <- function(input, output, session) {
  observeEvent(input$n, {
    label <- paste0("Simulate ", input$n, " times")
    updateActionButton(session, "simulate", label = label)
  })
}
```

https://www.kaggle.com/kyanyoga/sample-sales-data


Territory
Territory()
Customer
Customer(
)
Ordernumber
Data