BUILDING WEB APPLICATIONS IN R WITH SHINY

Using reactives

Why use reactives?

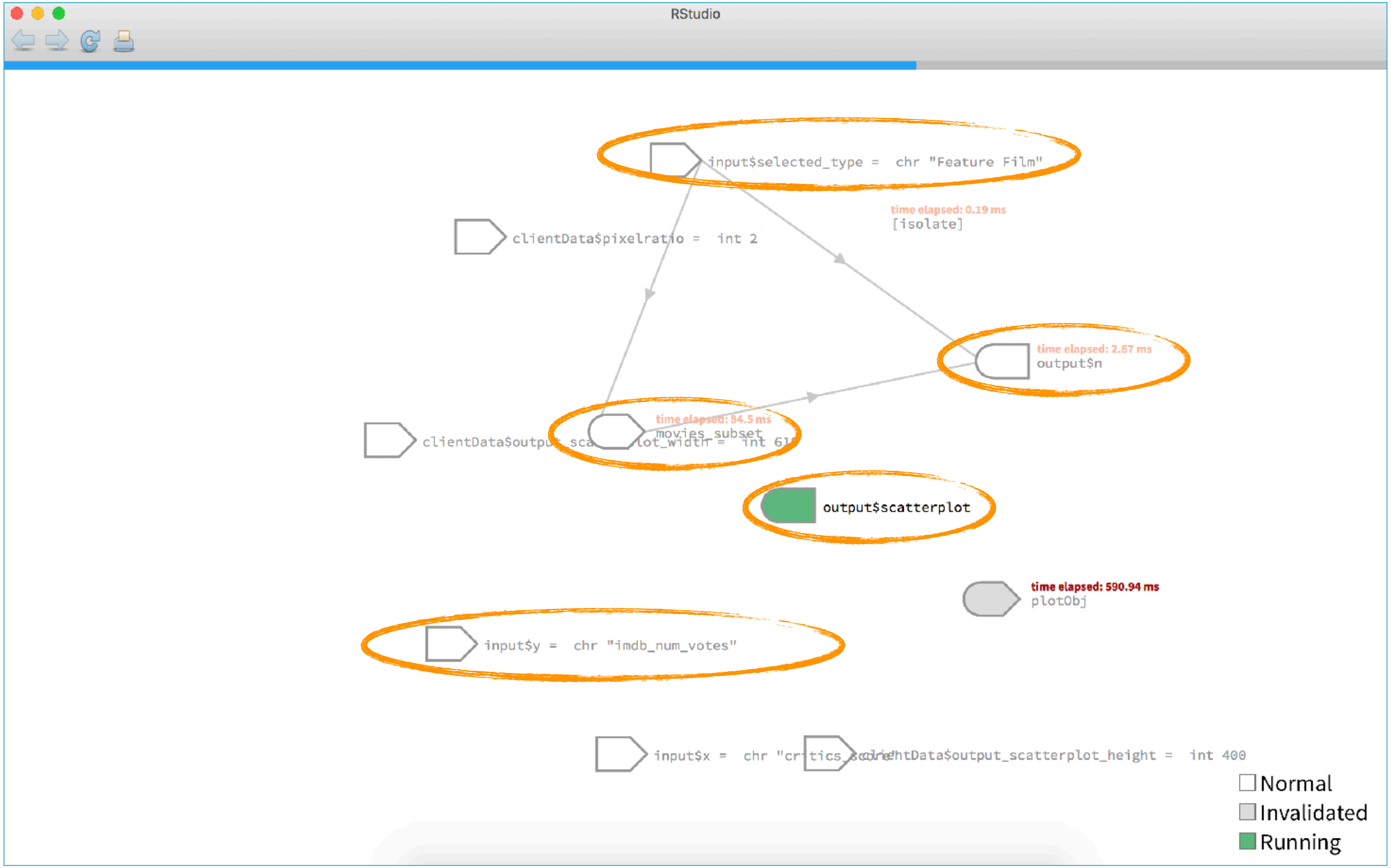
- By using a reactive expression for the subsetted data frame, we were able to get away with subsetting once and then using the result twice
- In general, reactive conductors let you
    - not repeat yourself (i.e. avoid copy-and-paste code)
    - decompose large, complex calculations into smaller pieces to make them more understandable
- Benefits similar to decomposing a large complex R script into a series of small functions that build on each other

Functions vs. reactives

- Each time you call a function, R will evaluate it.
- Reactive expressions are lazy, they only get executed when their input changes.
    - Even if you call a reactive expression multiple times, it only re-executes when its input(s) change.

Reactlog

- Using many reactive expressions in your app can create a complicated dependency structure in your app.
- The **reactlog** is a graphical representation of this dependency structure, and it also gives you you very detailed information about what's happening under the hood as Shiny evaluates your application
- To view:
    - In a fresh R session, run `options(shiny.reactlog = TRUE)`
    - Then, launch your app as you normally would
    - In the app, press Ctrl+F3


RStudio
input$selected_type = chr "Feature Film"
time elapsed: 0.19 ms
[isolate]
clientData$pixelratio = int 2
time elapsed: 2.67 ms
output$n
clientData$output_scatterplot_width = int 61
time elapsed: 84.5 ms
movies_subset
output$scatterplot
time elapsed: 590.94 ms
plotObj
input$y = chr "imdb_num_votes"
input$x = chr "critics_score"
clientData$output_scatterplot_height = int 400
☐ Normal
▨ Invalidated
🟩 Running

BUILDING WEB APPLICATIONS IN R WITH SHINY

Let's practice!