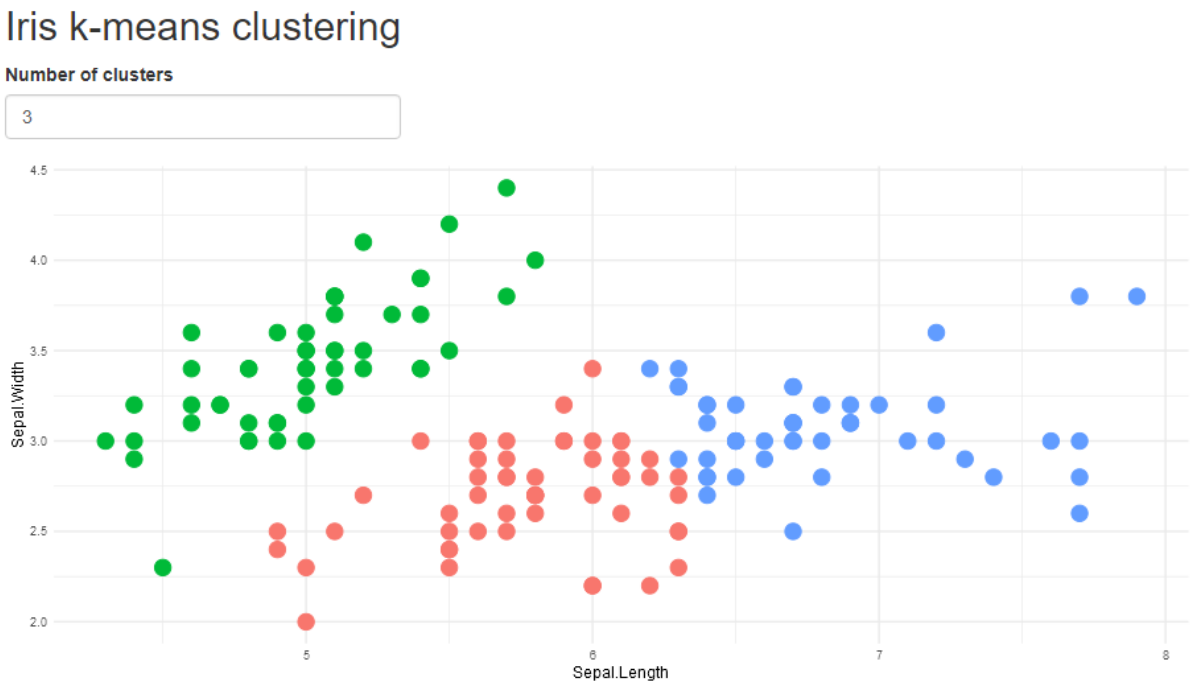
Intro to Shiny tutorial – next level

Start from the final basic app built in the intro tutorial - *shinyIntroFinal.R*

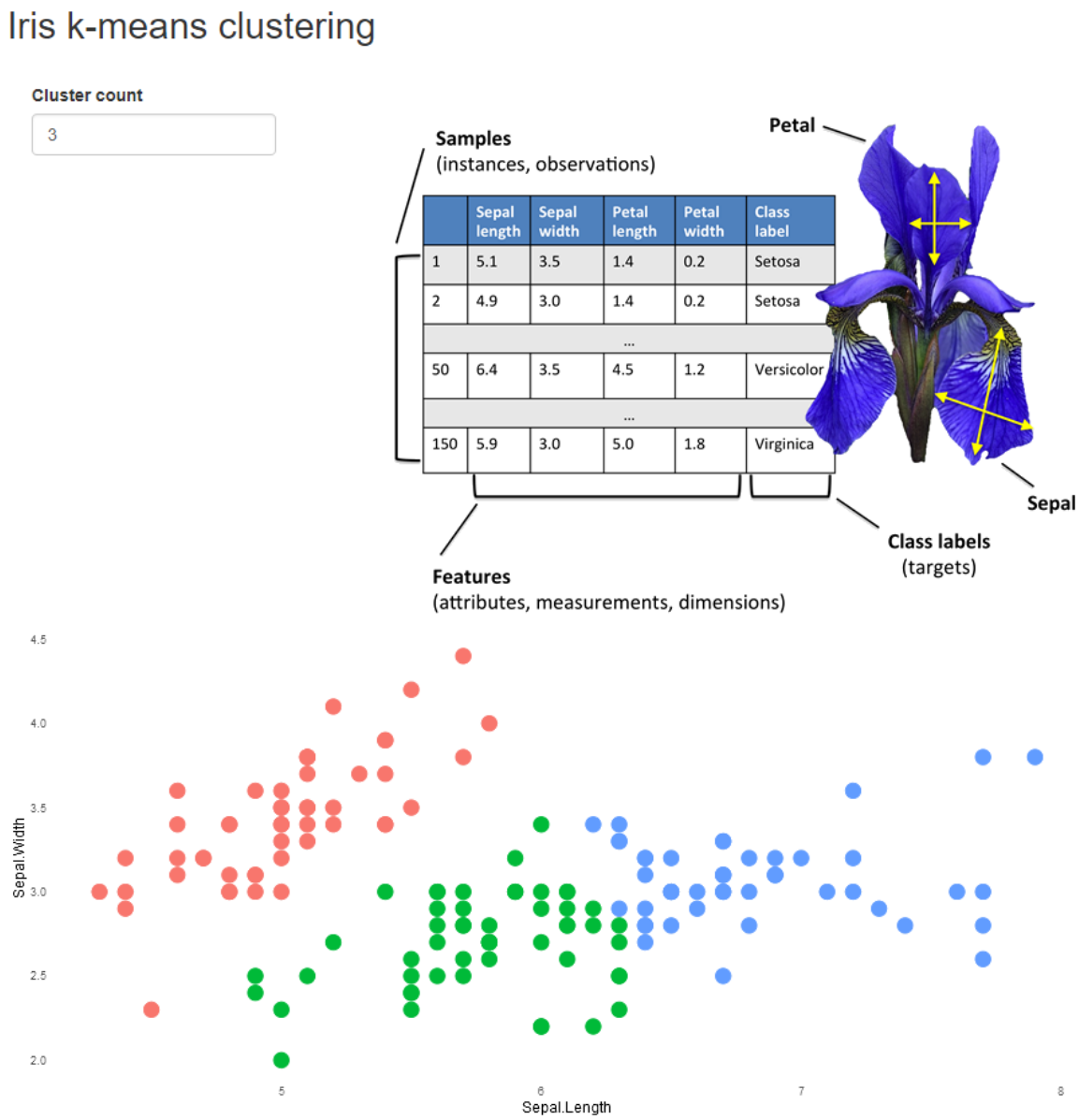


This app clusters the iris dataset sepal length and width according to the k-means clustering algorithm where the user can provide the number of clusters to be created (between 1 - 10)

ONCE YOU FINISHED ALL CHALLENGES (OR GET STUCK) YOU CAN SEE THE FINAL APP AND THE COMBINED CODE IN THE FILE - *ShinyIntroExtra.R*

Challenge 1 – Add more page structure and images

Add the image of the iris dataset (irisData.png) to the page and structure the page in such a way that the number of clusters to select and the image are next to each other, with the plot underneath. The result should look like this

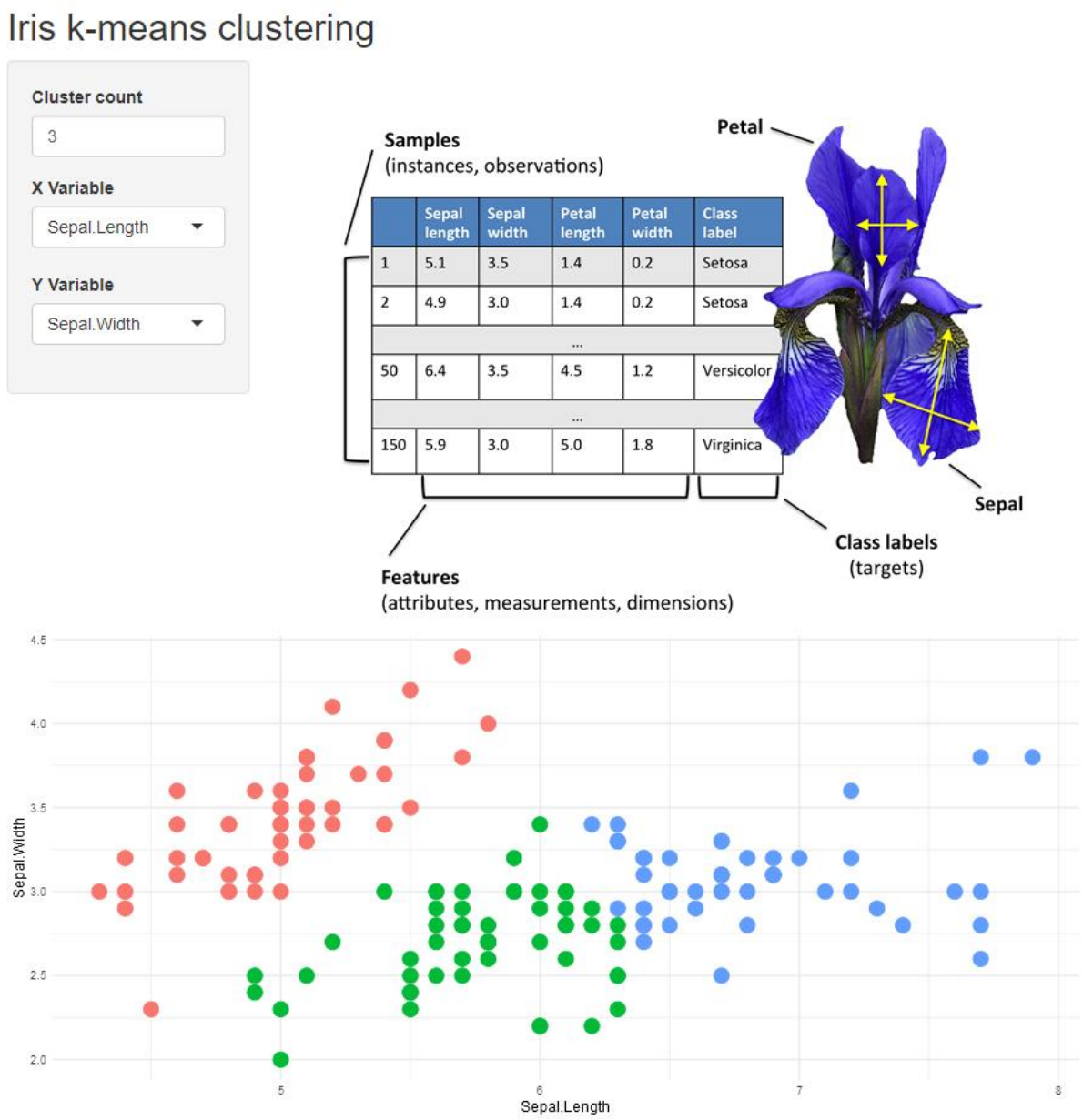


TIPS

- Use Shiny layout rows and columns
- Use Shiny tags to add custom HTML output elements
- Images need to reside in a specific location on order to be shown in the app…

Challenge 2 – Change the data selection

Create two extra selection boxes where the user can decide which of the columns of the iris dataset he/she wishes to use for clustering and make sure the plot updates accordingly. Not that the inputs are nicely grouped together visually in a darker box. The result should something like this:



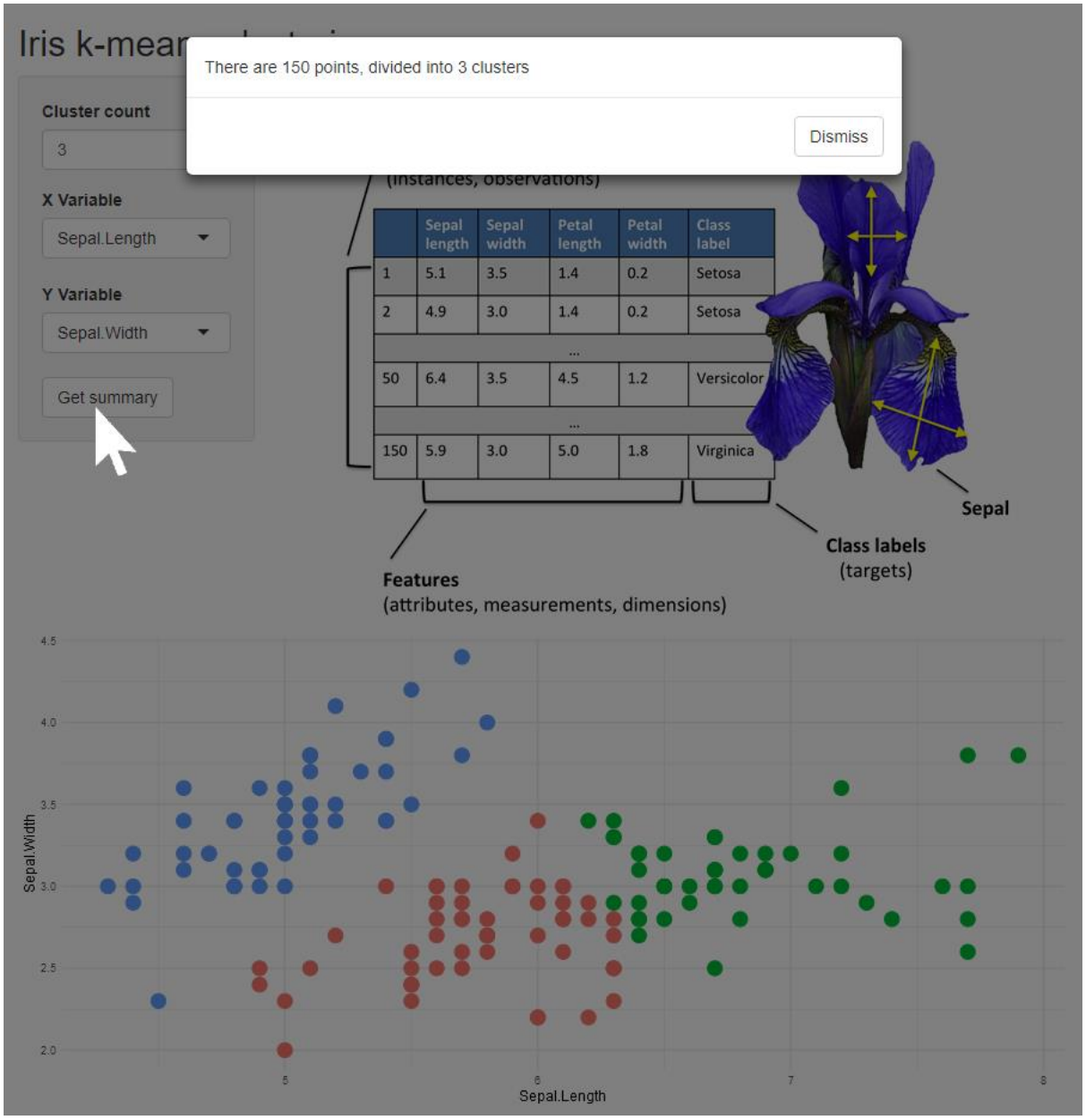
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

TIPS

- Look at the selectInput() function for layout
- Look at the wellPanel() for grouping UI features
- Building the ggplot will trickier as you'll need some tidyverse knowledge. The aes_string() function might come in handy…

Challenge 3 – Build a summary modal

Add a button to the page that will open a modal window (pop-up) displaying some very basic data statistics (number of points and number of clusters). The result should look something like this when the button is clicked:



TIPS

- Buttons often trigger code that need to have a separate reactive environment, take a look at the observeEvent() function
- Modal dialogs create new UI on the server that is then sent to the client and shown in a separate window