Shiny based medulloblastoma subgroup classifier

MATTHEW BASHTON

10TH OF APRIL 2017

Aims

Classifier designed by Reza, Amir and Ed allows for the subgroup of medulloblastoma (WNT, SHH, Group3, Group4) to be determined using 17 CpG beta values from Agena MassARRAY data

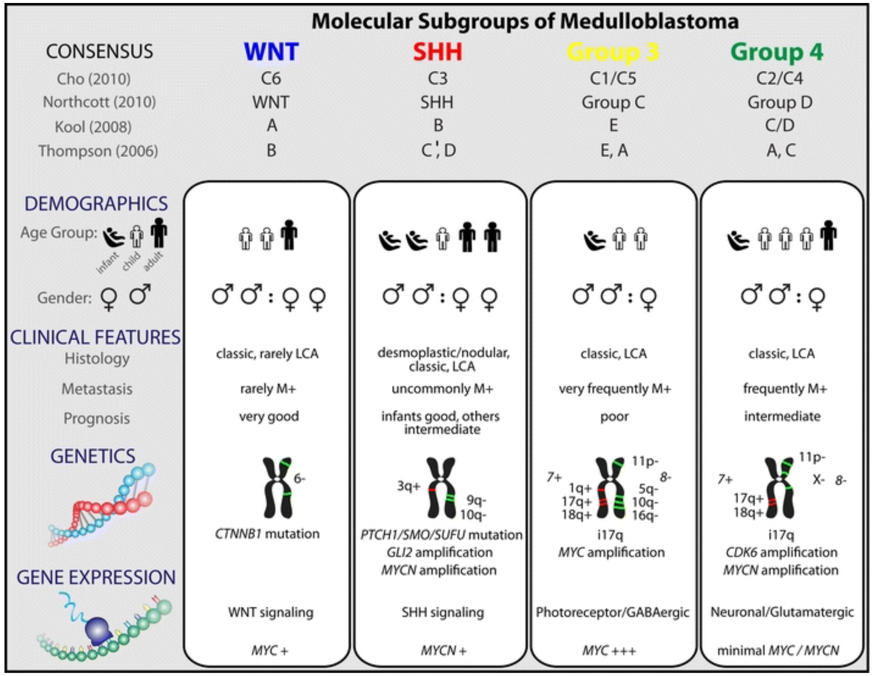
Aim was to enable easy classification of subgroups via end user in a molecular diagnostic setting

Subgroup classification essential for determining patient prognosis and treatment plan

For this we need an easy to use web application

Classifier is already based in R, so a Shiny based solution is naturally the choice

Medulloblastoma subgroups

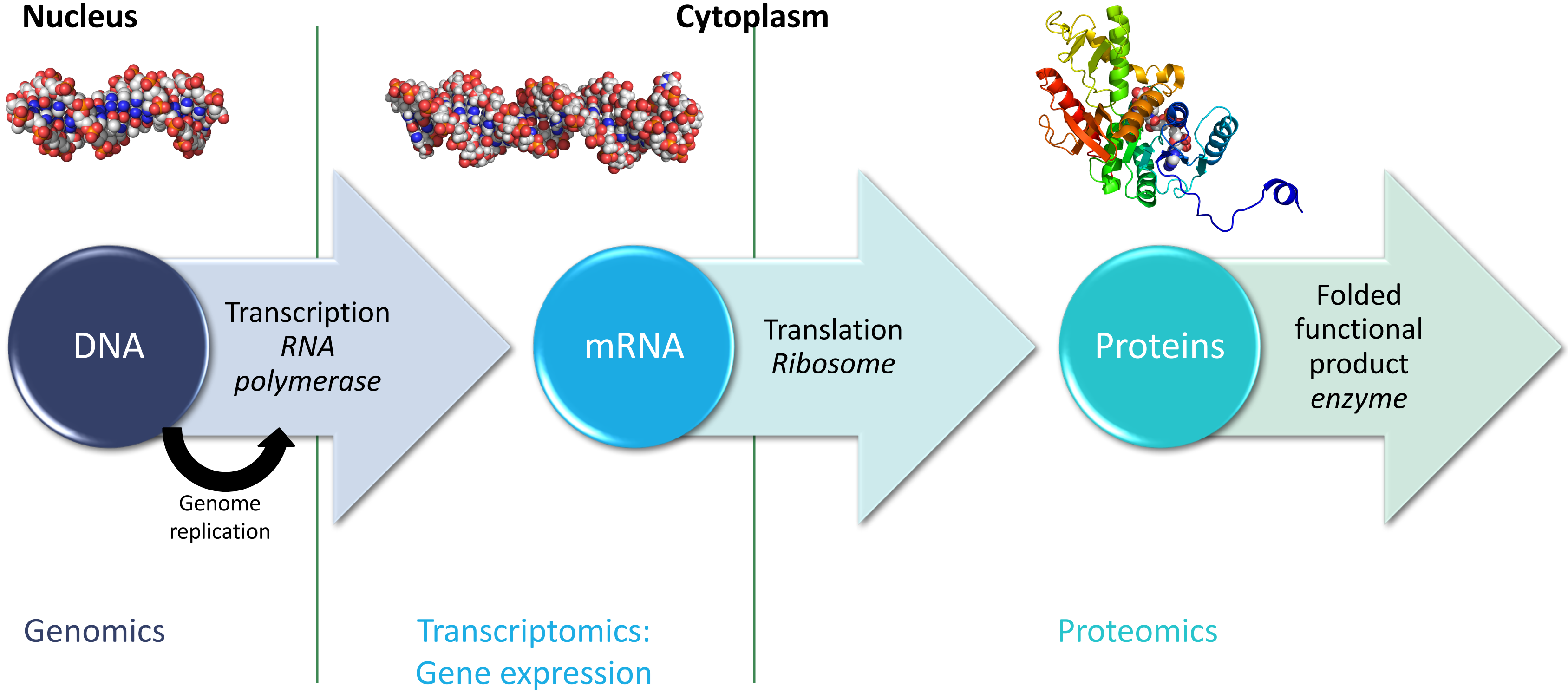


Molecular Subgroups of Medulloblastoma

CONSENSUS	WNT	SHH	Group 3	Group 4
Cho (2010)	C6	C3	C1/C5	C2/C4
Northcott (2010)	WNT	SHH	Group C	Group D
Kool (2008)	A	B	E	C/D
Thompson (2006)	B	C, D	E, A	A, C
DEMOGRAPHICS				
Age Group: infant child adult				
Gender: ♀ ♂	♂♂ : ♀♀	♂♂ : ♀♀	♂♂ : ♀	♂♂ : ♀
CLINICAL FEATURES				
Histology	classic, rarely LCA	desmoplastic/nodular, classic, LCA	classic, LCA	classic, LCA
Metastasis	rarely M+	uncommonly M+	very frequently M+	frequently M+
Prognosis	very good	infants good, others intermediate	poor	intermediate
GENETICS	6- *CTNNB1* mutation	3q+ 9q- 10q- *PTCH1/SMO/SUFU* mutation *GLI2* amplification *MYCN* amplification	7+ 1q+ 17q+ 18q+ 11p- 8- 5q- 10q- 16q- i17q *MYC* amplification	7+ 17q+ 18q+ 11p- X- 8- i17q *CDK6* amplification *MYCN* amplification
GENE EXPRESSION	WNT signaling *MYC* +	SHH signaling *MYCN* +	Photoreceptor/GABAergic *MYC* +++	Neuronal/Glutamatergic minimal *MYC / MYCN*

Take from: Taylor *et al*. *Acta Neuropathol*. 2012 Apr; **123**(4): 465–472.

Central Dogma of Molecular Biology



Nucleus

Cytoplasm

DNA

Transcription
RNA polymerase

Genome replication

mRNA

Translation
Ribosome

Proteins

Folded functional product
enzyme

Genomics

Transcriptomics: Gene expression

Proteomics

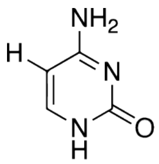
DNA methylation

A process by which **methyl groups (CH_3)** are added to the DNA molecule

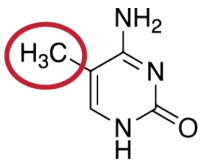
Normally in humans it’s the C (cytosine) bases preceding a G (guanine) which are methylated: CpG

Methylation is sparse but global, 60-80% of all CpG in the genome are methylated

Hypermethylation of CpG sites in a gene leads to **gene silencing**, *i.e*. it is no longer expressed and made into protein



Cytosine



methylated Cytosine

DNA methylation

Methylation states can persist thought cellular replication, and are a form of **epigenetics**, *i.e*. a heritable trait not directly encoded by the sequence of DNA it's self

Particular subgroups have different methylation patterns across the genome

These were first detected by methylation arrays, such as the 450k array, which can detect methylation on over 450 thousand different CpG sites, we use on 17 CpGs in our classifier

DNA methylation

Methylation status of a CpG is reported at a β-value:

$$Beta_i = \frac{\max(y_{i,methy}, 0)}{\max(y_{i,unmethy}, 0) + \max(y_{i,methy}, 0) + \alpha}$$

Where $y_{i,menty}$ and $y_{i,unmenty}$ are the intensities mea-sured by the i^{th} methylated and unmethylated probes

Alpha is a kludge factor, to give a constant offset to regularise the beta value, normally = 100

β-values are between 0 and 1

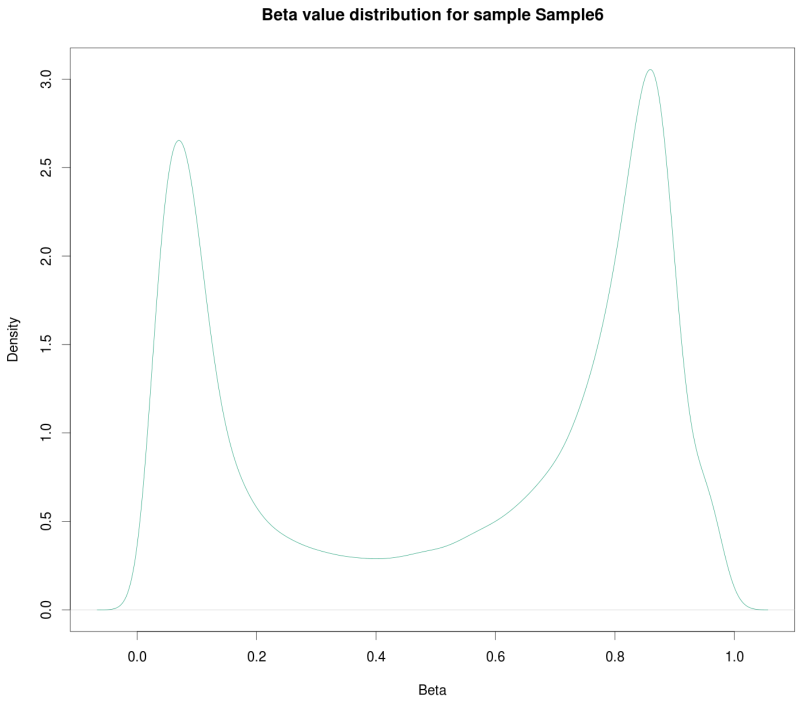
0 = all CpGs at said site in sample are completely unmethylated, 1 = all CpGs at site methylated

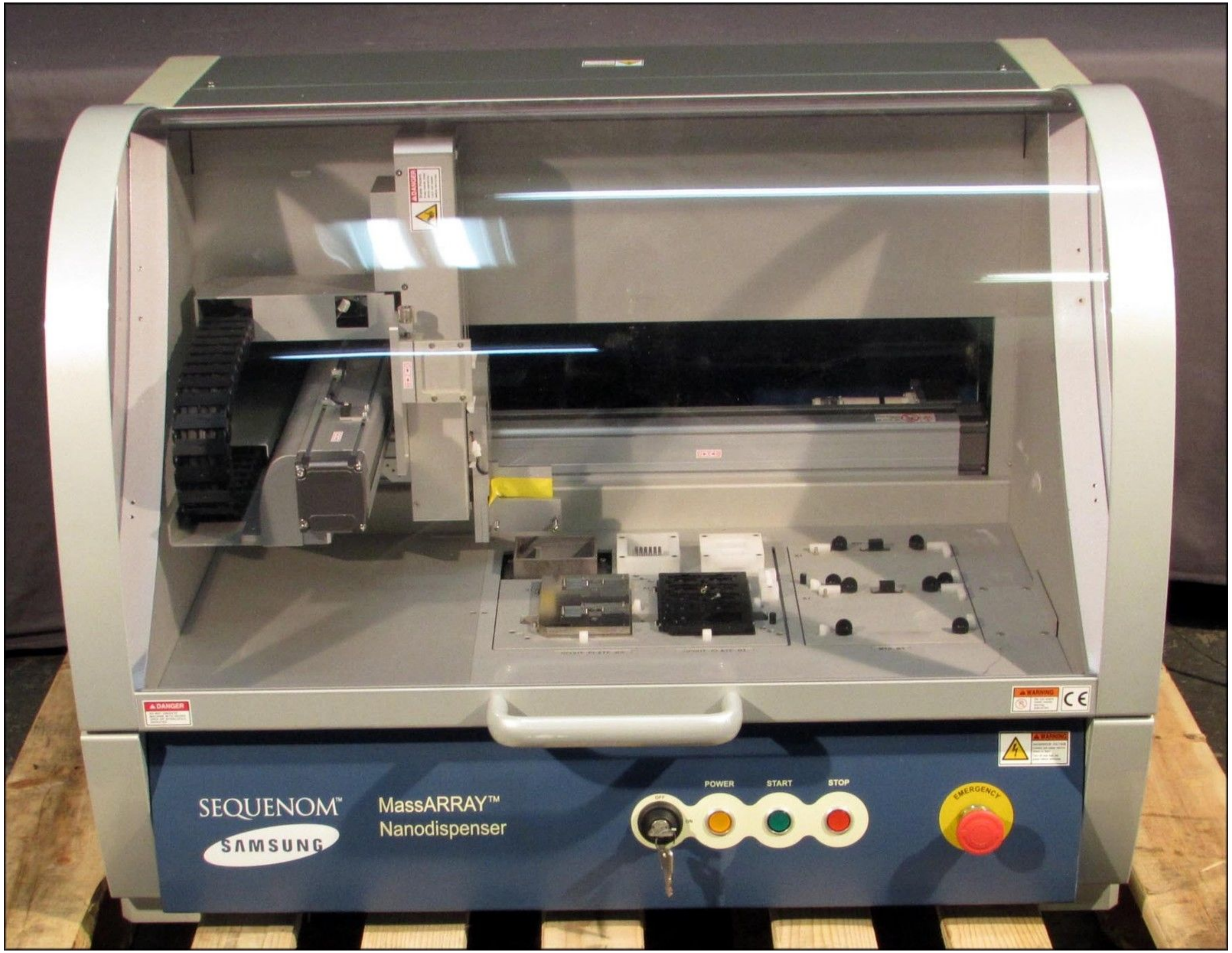
Beta value distribution

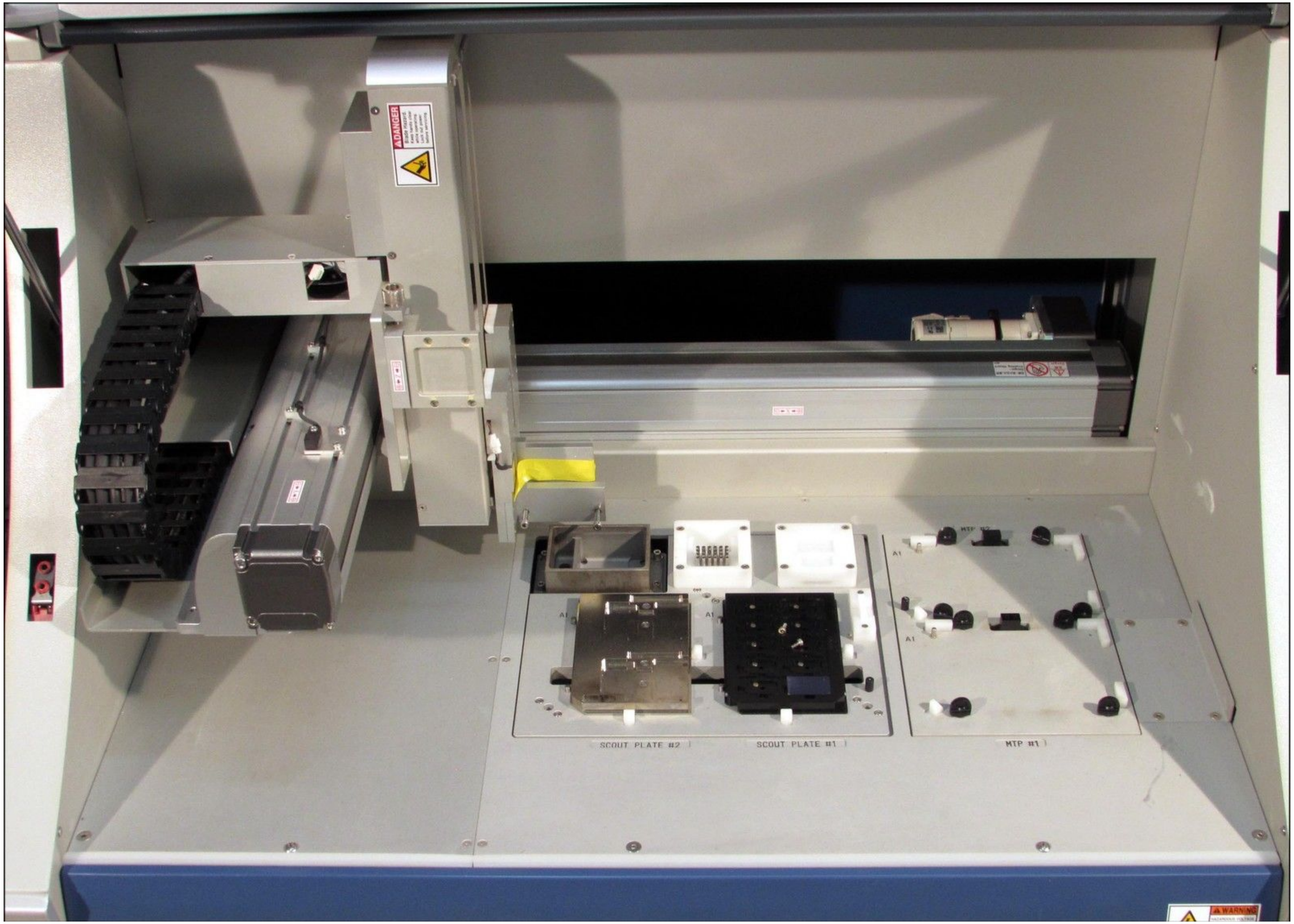
β-values are normally distributed around 0 *i.e*. unmethylated

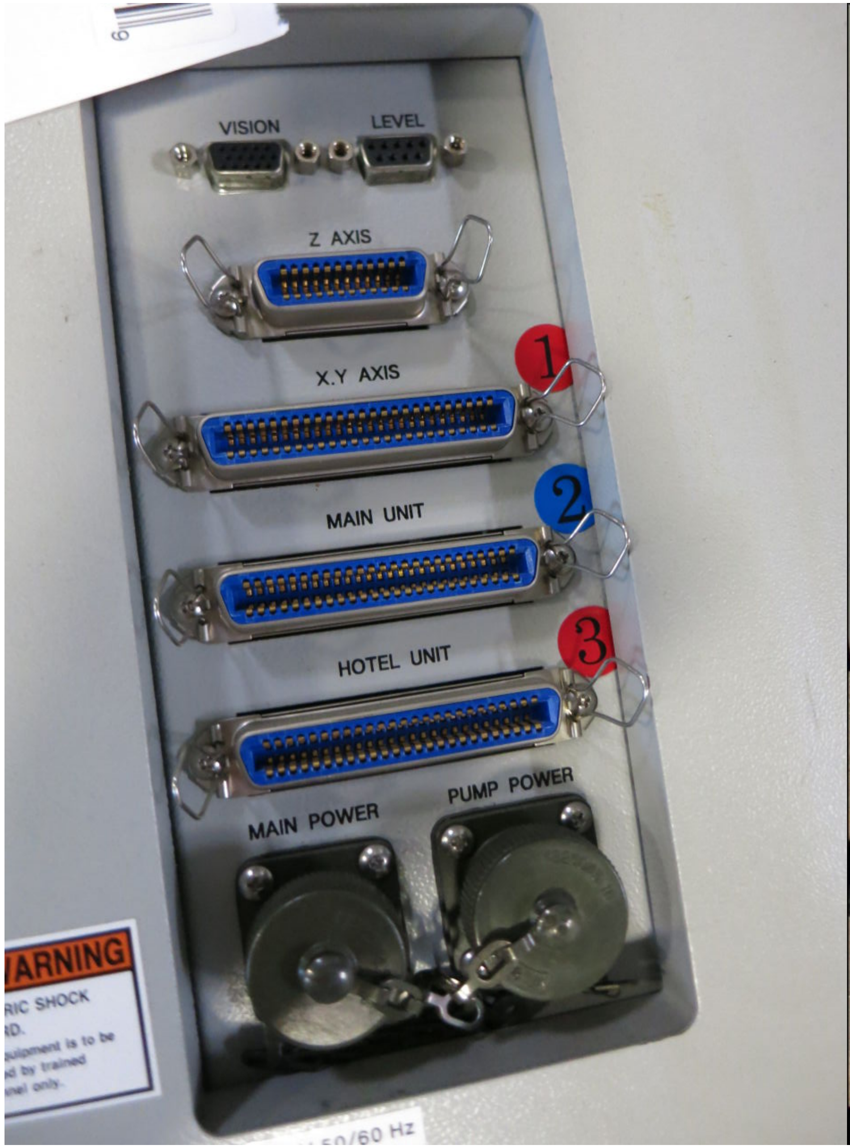
and 0.9 predominantly methylated

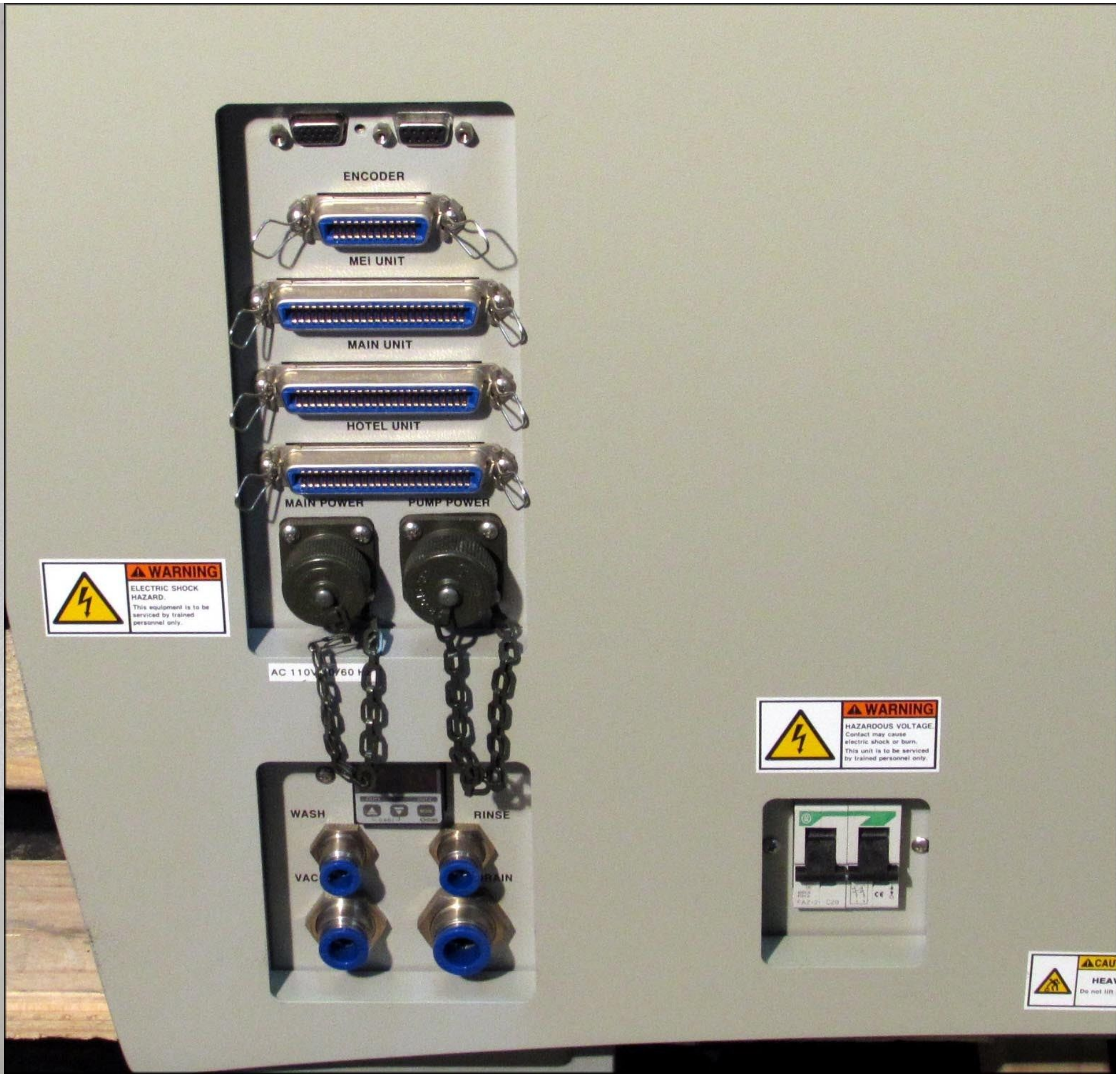
We only need 17 carefully selected Beta values to determine the type of medulloblastoma

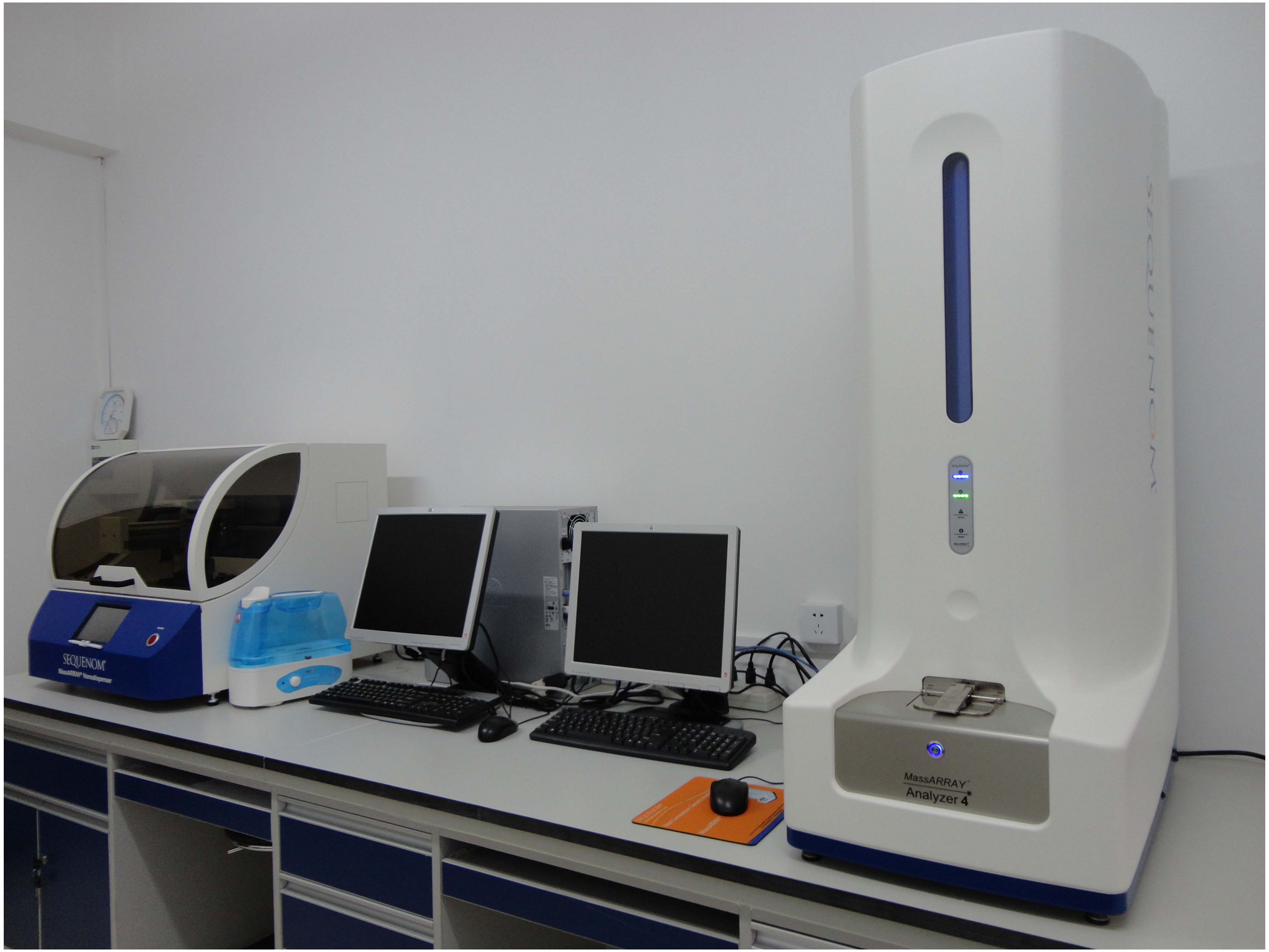


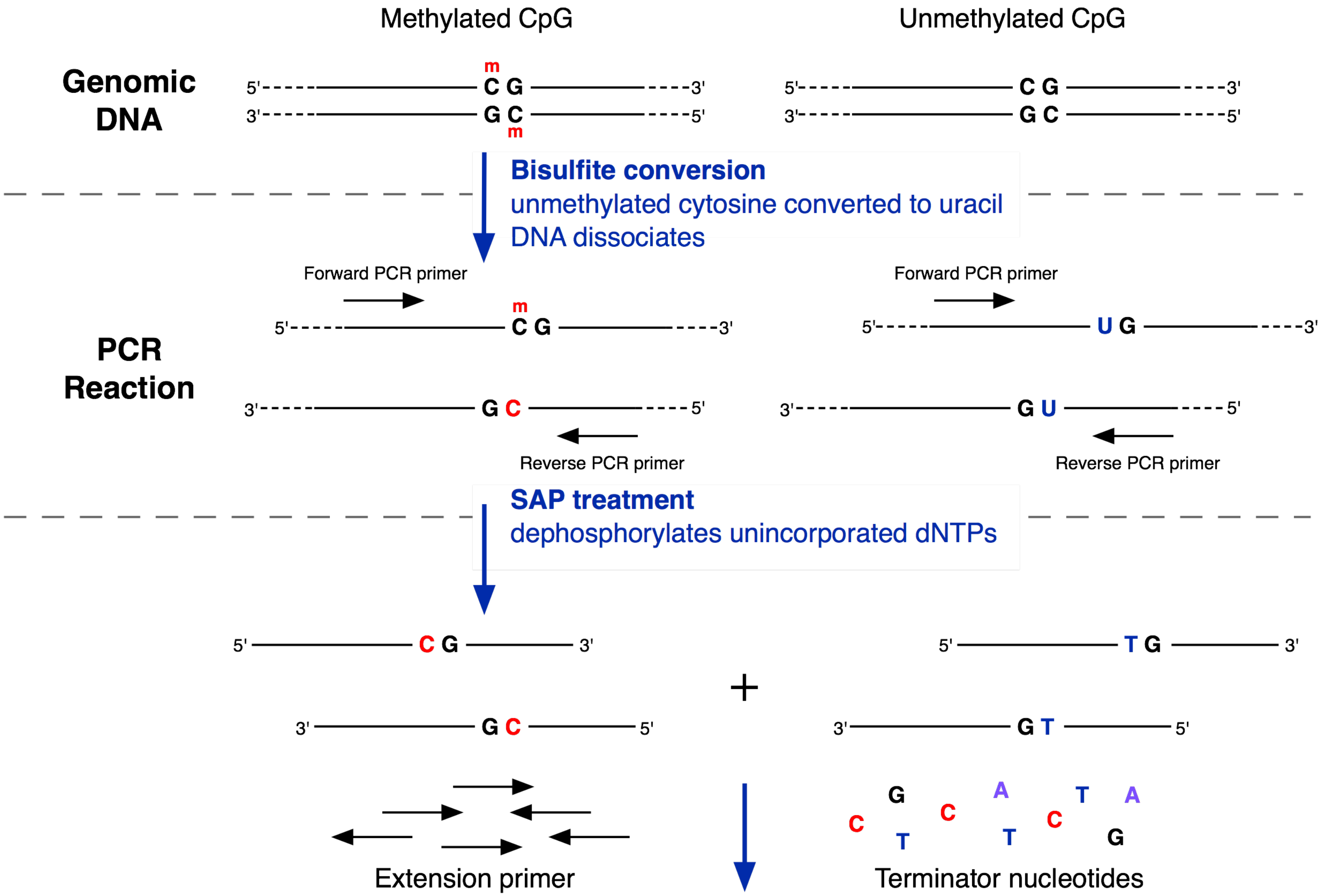

SEQUENOM™
SAMSUNG
MassARRAY™
Nanodispenser
POWER
START
STOP
EMERGENCY
DANGER
WARNING
CE


DANGER
MTP #2
A1
A1
A1
A1
SCOUT PLATE #2
SCOUT PLATE #1
MTP #1
WARNING


VISION
LEVEL
Z AXIS
X.Y AXIS
1
MAIN UNIT
2
HOTEL UNIT
3
MAIN POWER
PUMP POWER
ARNING
RIC SHOCK
RD.
uipment is to be
d by trained
nel only.
50/60 Hz


ENCODER
MEI UNIT
MAIN UNIT
HOTEL UNIT
MAIN POWER
PUMP POWER
WARNING
ELECTRIC SHOCK HAZARD.
This equipment is to be serviced by trained personnel only.
AC 110V 60 H
WASH
RINSE
VAC
RAIN
WARNING
HAZARDOUS VOLTAGE.
Contact may cause electric shock or burn.
This unit is to be serviced by trained personnel only.
CAU
HEA
Do not lift


SEQUENOM
MassARRAY Nanodispenser
SEQUENOM
MassARRAY
Analyzer 4



Methylated CpG

Unmethylated CpG

Genomic DNA

Bisulfite conversion

unmethylated cytosine converted to uracil

DNA dissociates

PCR Reaction

Forward PCR primer

Forward PCR primer

Reverse PCR primer

Reverse PCR primer

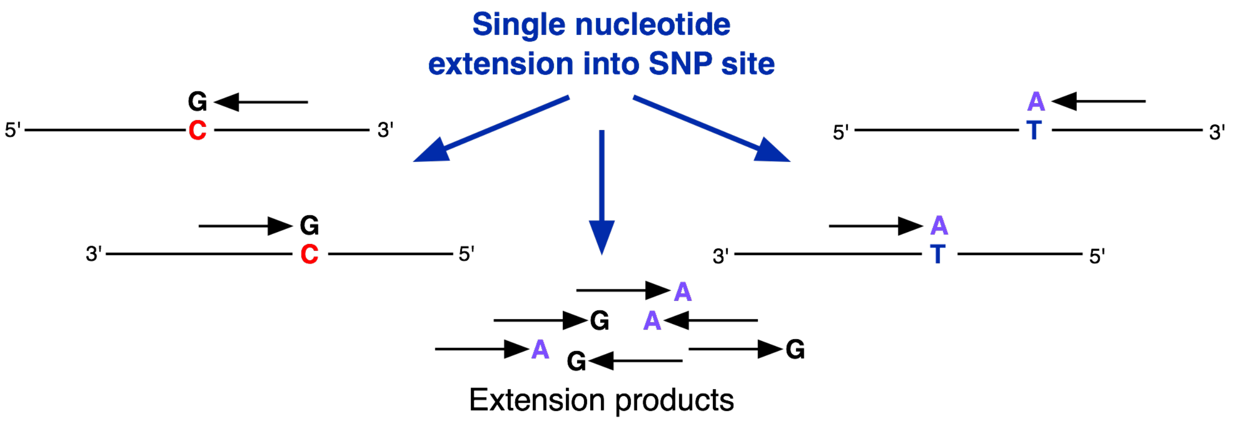
SAP treatment

dephosphorylates unincorporated dNTPs

Extension primer

Terminator nucleotides

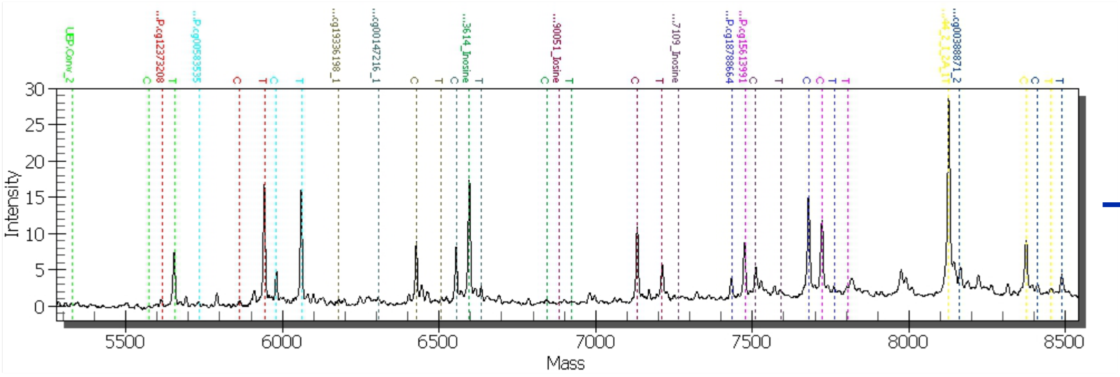
iPLEX Extension Reaction



Extension products

Desalt, dispense onto SpectroCHIP Array
analyse with MALDI-TOF MS

MALDI-TOF Mass Spectrometry Analysis



CSV file of peak heights

Shiny

Interactive web development framework for R

Shiny is reactive, this means when inputs change outputs which use this input change

Very different to the old CGI-BIN style of web development

Everything is encoded as functions with `input` and `output` objects passed between them different slots are used for each input/output, reactive flow is said connect functions

Reactive plumbing can get a bit complicated

Allows for computationally expensive classification step to be separated from display/output logic

http://shiny.rstudio.com

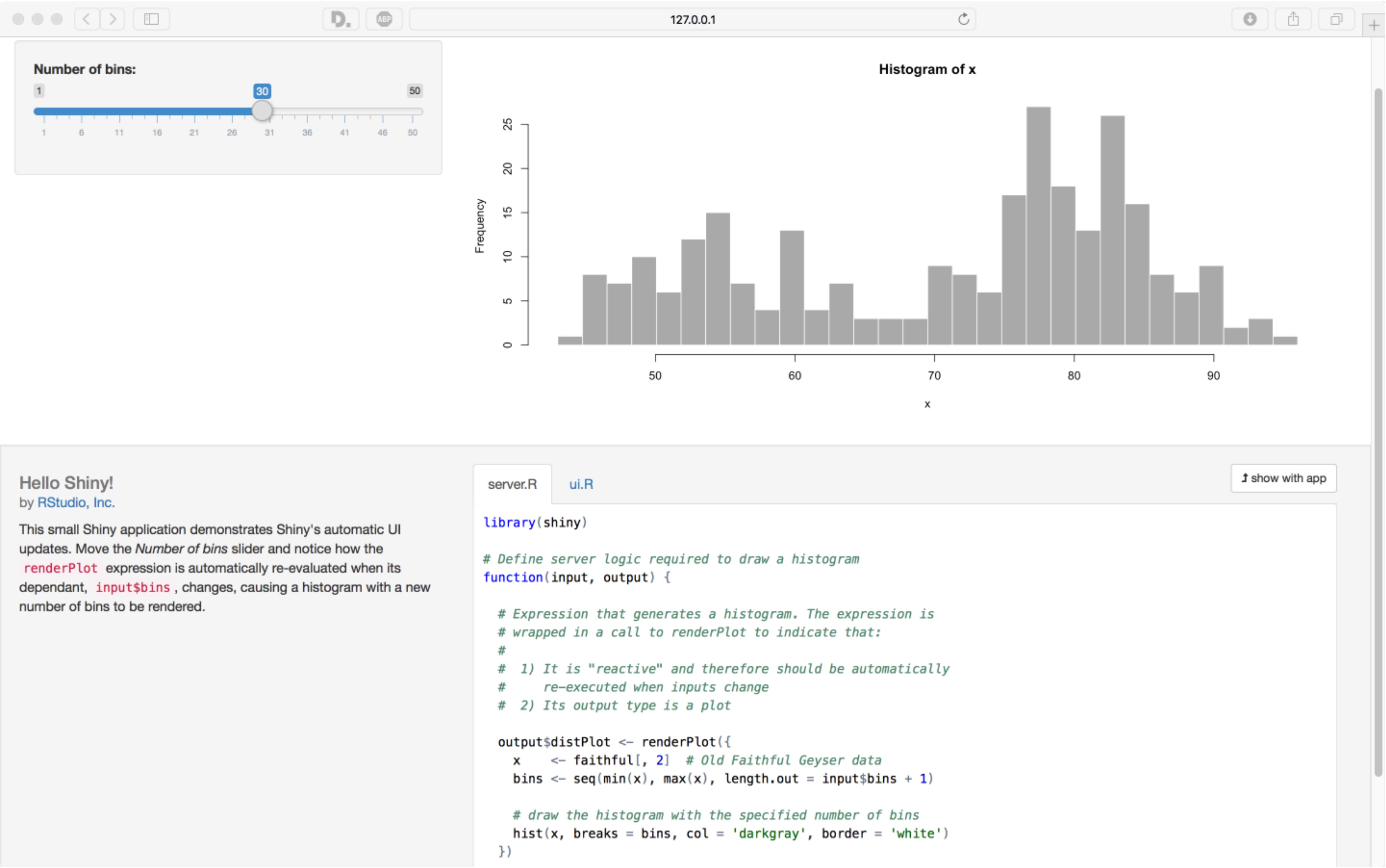
An example app: `ui.R`

```
> library(shiny)
> runExample("01_hello")
```

```
library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

    # Application title
    titlePanel("Hello Shiny!"),

    # Sidebar with a slider input for the number of bins
    sidebarLayout(
        sidebarPanel(
            sliderInput("bins", "Number of bins:", min = 1, max = 50, value = 30)
        ),

    # Show a plot of the generated distribution
    mainPanel(
        plotOutput("distPlot")
    )
  )
))
```

An example app: server.R

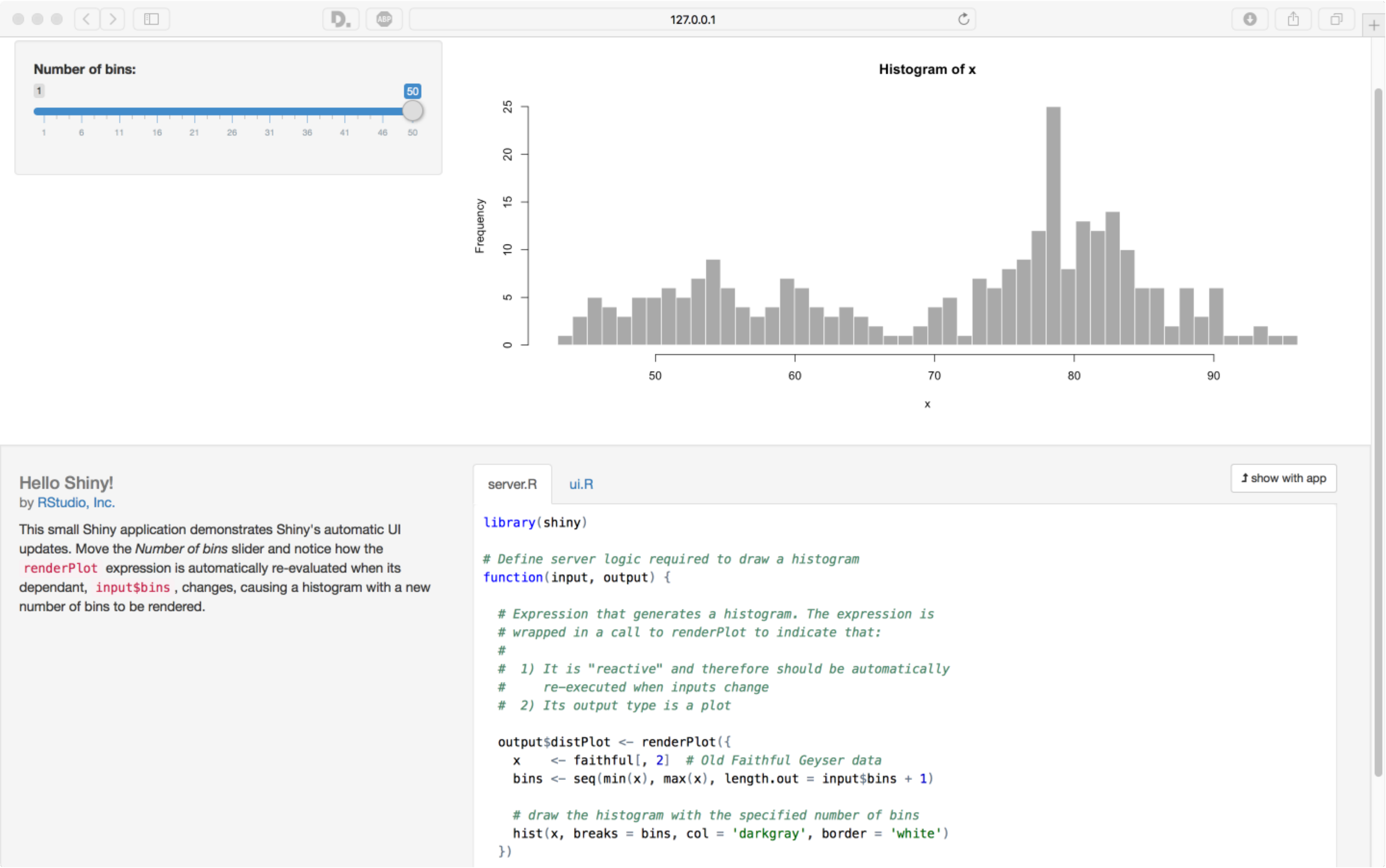
```r
library(shiny)
# Define server logic required to draw a histogram
shinyServer(function(input, output) {

# Expression that generates a histogram, wrapped in a call to renderPlot
# 1)Its "reactive" and is automatically re-executed when inputs change
# 2)Its output type is a plot

    output$distPlot <- renderPlot({
        x <- faithful[, 2] # Old Faithful Geyser data
        bins <- seq(min(x), max(x), length.out = input$bins + 1)

        # draw the histogram with the specified number of bins
        hist(x, breaks = bins, col = 'darkgray', border = 'white')

     })
})
```



Hello Shiny!
by RStudio, Inc.

This small Shiny application demonstrates Shiny's automatic UI updates. Move the *Number of bins* slider and notice how the `renderPlot` expression is automatically re-evaluated when its dependant, `input$bins`, changes, causing a histogram with a new number of bins to be rendered.

```
library(shiny)

# Define server logic required to draw a histogram
function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should be automatically
  #     re-executed when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
```



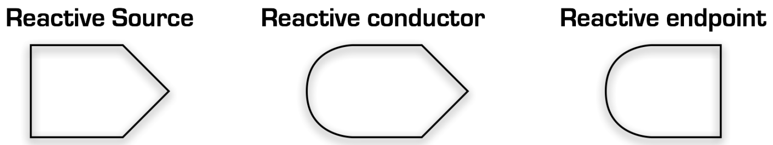
Hello Shiny!

by RStudio, Inc.

This small Shiny application demonstrates Shiny's automatic UI updates. Move the *Number of bins* slider and notice how the `renderPlot` expression is automatically re-evaluated when its dependant, `input$bins`, changes, causing a histogram with a new number of bins to be rendered.

```r
library(shiny)

# Define server logic required to draw a histogram
function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should be automatically
  #     re-executed when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
```

Shiny reactivity



The reactive input is from input slider widget `sliderInput("bins", …)` this will populate the `$bins` slot of the `input` object

On the server side we access this via `input$bins`

We don’t have a reactive conductor in this case simply input and output

The reactive output is our histogram which is captured by wrapping our plot in `renderPlot({})`:

```
output$distPlot <- renderPlot({…})
```

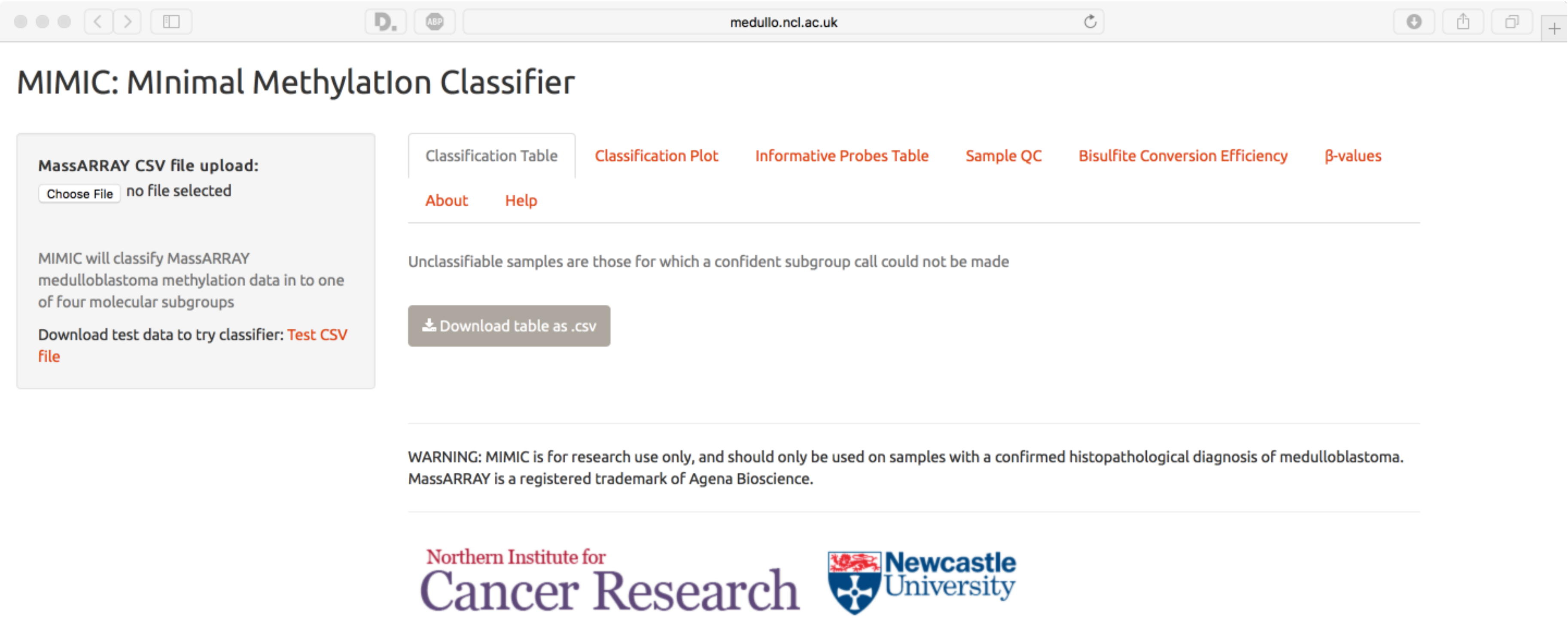
On the UI side we access the plot via:

```
plotOutput("distPlot")
```

When `input$bins` changes, `output$distPlot` will update accordingly with server side code re-executing, we don’t need to code explicit event handlers or arrange objects to be updated this happens automatically

Demo App

http://medullo.ncl.ac.uk



MIMIC: MInimal MethylatIon Classifier

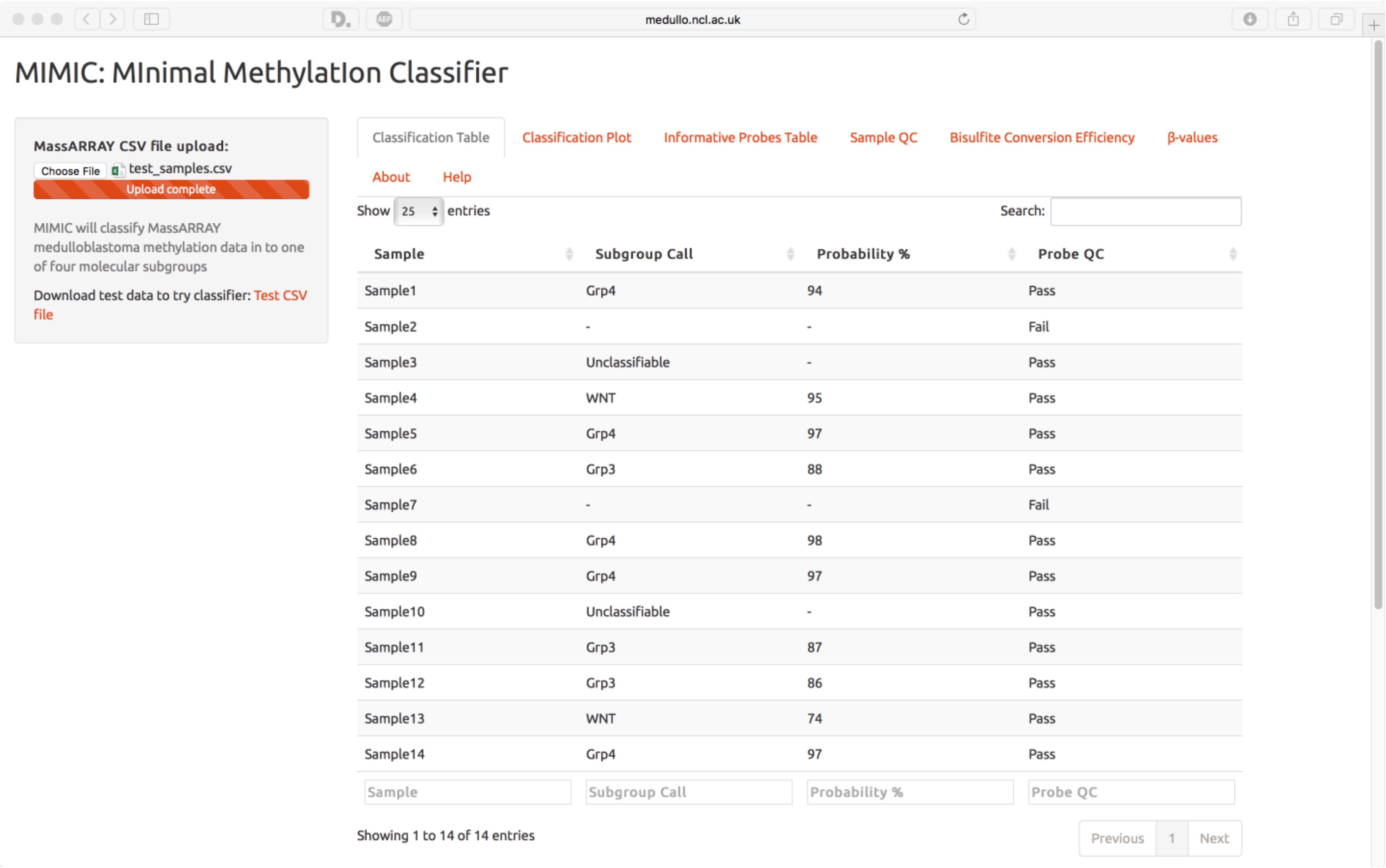
MassARRAY CSV file upload:

MIMIC will classify MassARRAY medulloblastoma methylation data in to one of four molecular subgroups

Download test data to try classifier: Test CSV file

Unclassifiable samples are those for which a confident subgroup call could not be made

WARNING: MIMIC is for research use only, and should only be used on samples with a confirmed histopathological diagnosis of medulloblastoma. MassARRAY is a registered trademark of Agena Bioscience.



MIMIC: MInimal MethylatIon Classifier

MassARRAY CSV file upload:

test_samples.csv

MIMIC will classify MassARRAY medulloblastoma methylation data in to one of four molecular subgroups

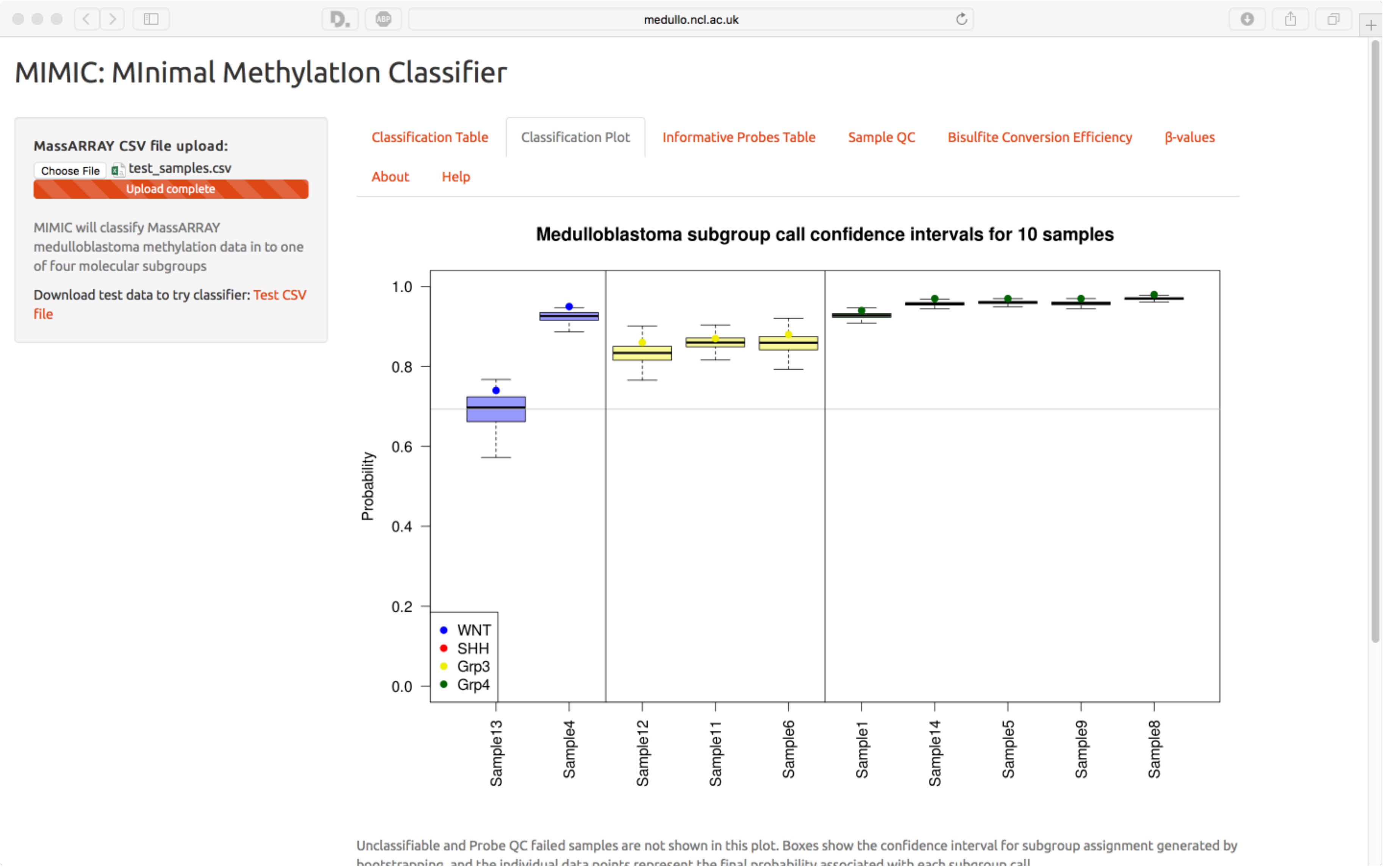
Download test data to try classifier: Test CSV file

Show 25 entries

Search:

Sample	Subgroup Call	Probability %	Probe QC
Sample1	Grp4	94	Pass
Sample2	-	-	Fail
Sample3	Unclassifiable	-	Pass
Sample4	WNT	95	Pass
Sample5	Grp4	97	Pass
Sample6	Grp3	88	Pass
Sample7	-	-	Fail
Sample8	Grp4	98	Pass
Sample9	Grp4	97	Pass
Sample10	Unclassifiable	-	Pass
Sample11	Grp3	87	Pass
Sample12	Grp3	86	Pass
Sample13	WNT	74	Pass
Sample14	Grp4	97	Pass
Sample	Subgroup Call	Probability %	Probe QC

Showing 1 to 14 of 14 entries



MIMIC: MInimal MethylatIon Classifier

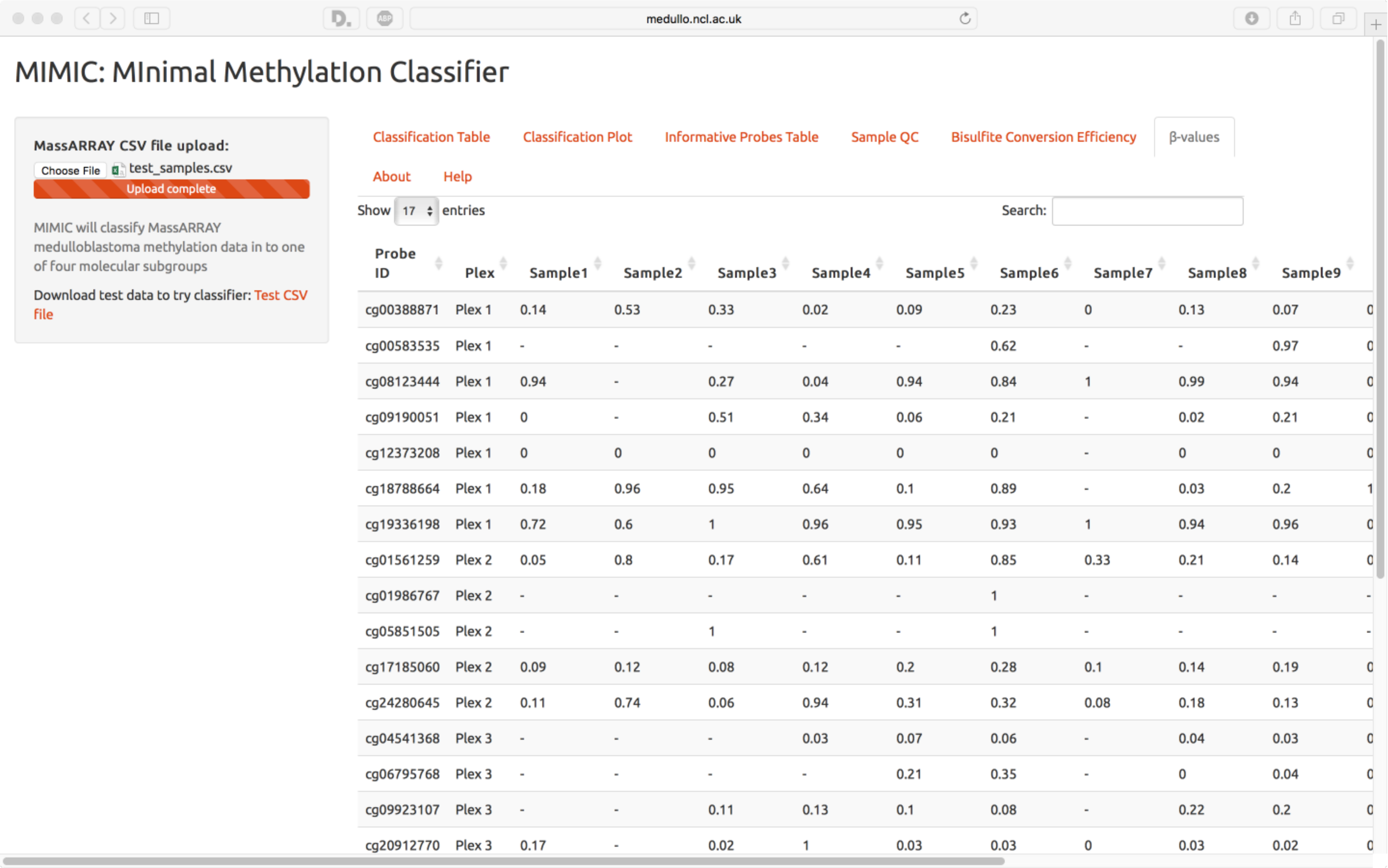
MassARRAY CSV file upload:

MIMIC will classify MassARRAY medulloblastoma methylation data in to one of four molecular subgroups

Download test data to try classifier: Test CSV file

Medulloblastoma subgroup call confidence intervals for 10 samples

Unclassifiable and Probe QC failed samples are not shown in this plot. Boxes show the confidence interval for subgroup assignment generated by bootstrapping, and the individual data points represent the final probability associated with each subgroup call.



MIMIC: MInimal MethylatIon Classifier

MassARRAY CSV file upload:

test_samples.csv

MIMIC will classify MassARRAY medulloblastoma methylation data in to one of four molecular subgroups

Download test data to try classifier: Test CSV file

Show 17 entries

Search:

Probe ID	Plex	Sample1	Sample2	Sample3	Sample4	Sample5	Sample6	Sample7	Sample8	Sample9	
cg00388871	Plex 1	0.14	0.53	0.33	0.02	0.09	0.23	0	0.13	0.07	0
cg00583535	Plex 1	-	-	-	-	-	0.62	-	-	0.97	0
cg08123444	Plex 1	0.94	-	0.27	0.04	0.94	0.84	1	0.99	0.94	0
cg09190051	Plex 1	0	-	0.51	0.34	0.06	0.21	-	0.02	0.21	0
cg12373208	Plex 1	0	0	0	0	0	0	-	0	0	0
cg18788664	Plex 1	0.18	0.96	0.95	0.64	0.1	0.89	-	0.03	0.2	1
cg19336198	Plex 1	0.72	0.6	1	0.96	0.95	0.93	1	0.94	0.96	0
cg01561259	Plex 2	0.05	0.8	0.17	0.61	0.11	0.85	0.33	0.21	0.14	0
cg01986767	Plex 2	-	-	-	-	-	1	-	-	-	-
cg05851505	Plex 2	-	-	1	-	-	1	-	-	-	-
cg17185060	Plex 2	0.09	0.12	0.08	0.12	0.2	0.28	0.1	0.14	0.19	0
cg24280645	Plex 2	0.11	0.74	0.06	0.94	0.31	0.32	0.08	0.18	0.13	0
cg04541368	Plex 3	-	-	-	0.03	0.07	0.06	-	0.04	0.03	0
cg06795768	Plex 3	-	-	-	-	0.21	0.35	-	0	0.04	0
cg09923107	Plex 3	-	-	0.11	0.13	0.1	0.08	-	0.22	0.2	0
cg20912770	Plex 3	0.17	-	0.02	1	0.03	0.03	0	0.03	0.02	0

How it works (Briefly)

Main reactive classification function `classifier()` does the classification inside `server.R`

A support function `cleanSeq4()` handles processing CSV mass spec peaks input data

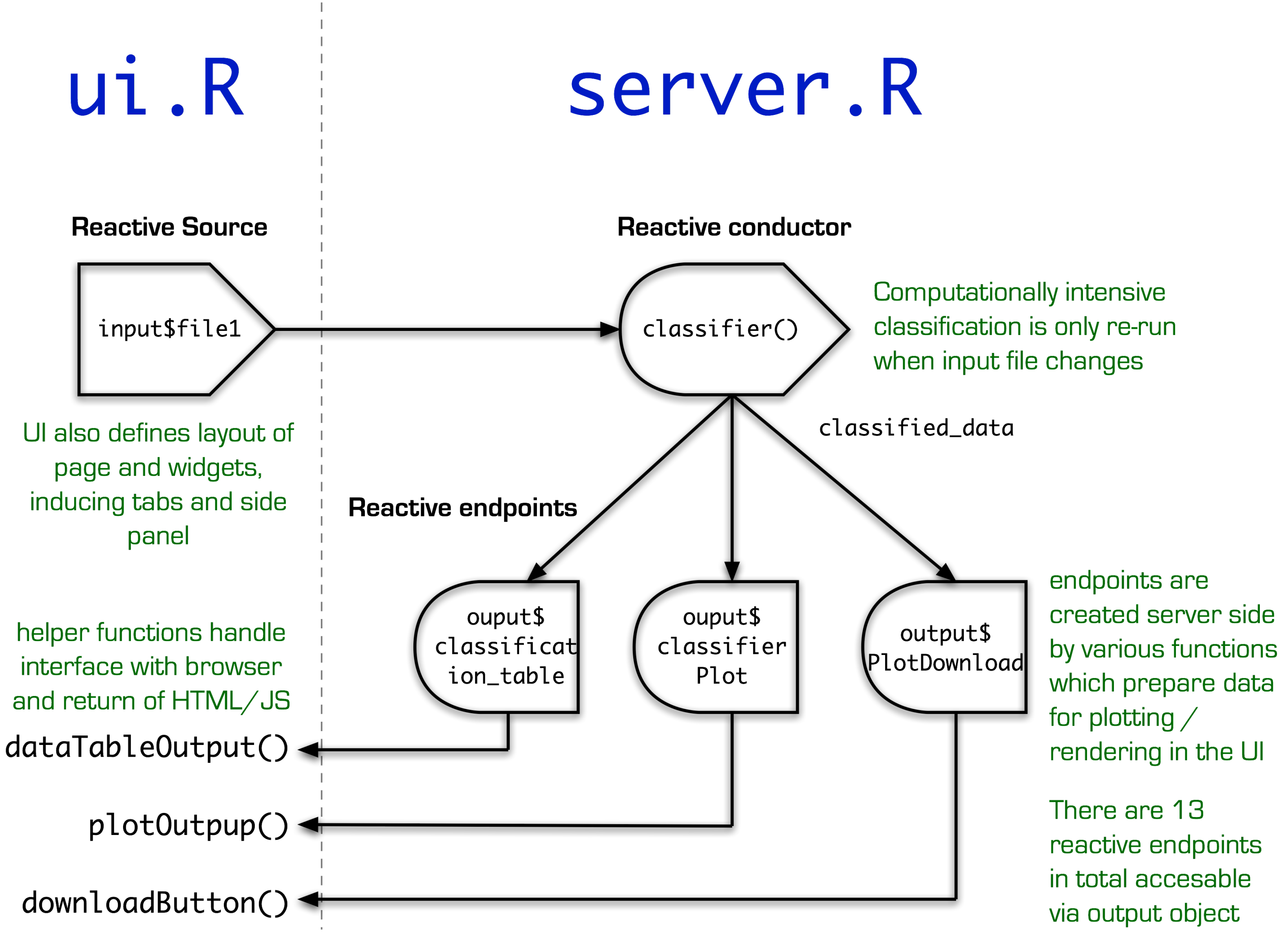
`classifier()` returns a list `classified_data` with all data needed to populate reactive endpoints on the server side, quite a lot of post processing done on server side to get data ready to plot, tabulate etc.

Various plots, tables and downloads are handled by wrapping munging/plotting code in reactive functions in `server.R`: `renderDataTable({})`, `downloadHandler({})`, `renderText({})`, `renderPlot({})`

The output of each of the above is assigned to a reactive end point - a slot in the output object: `output$X` <- where X is the name of the output you want to render/plot in the UI

On the `ui.R` side the shiny helper functions: `dataTableOutput()`, `plotOutput()`, `textOutput()`, `helpText()`, `downloadButton()` create HTML/JS to be rendered in the browser

App is hosted on NUIT provided Ubuntu VM, which surprisingly works quite well



ui.R

server.R

Reactive Source

input$file1

Reactive conductor

classifier()

Computationally intensive classification is only re-run when input file changes

classified_data

UI also defines layout of page and widgets, inducing tabs and side panel

Reactive endpoints

ouput$ classificat ion_table

ouput$ classifier Plot

output$ PlotDownload

endpoints are created server side by various functions which prepare data for plotting / rendering in the UI

helper functions handle interface with browser and return of HTML/JS

dataTableOutput()

plotOutpup()

downloadButton()

There are 13 reactive endpoints in total accesable via output object

Overview

MIMIC will classify MassARRAY medulloblastoma methylation data in to one of four molecular subgroups: WNT, SHH, Group 3 and Group 4.

In summary the classifier works as described below:

1. We use 17 different methylation probes in an Agena iPLEX assay , the readout is performed by the MassARRAY mass spectrometer.
2. Peak heights from the Mass Spectrometer, corresponding to the 17 probes for each sample are outputted as a comma separated .csv file; these values are submitted to MIMIC and converted to β values for each probe.
3. The number of probes successfully reporting β values out of the 17 is assessed for each sample, imputation (exploiting our own MassARRAY cohort) is used to impute any missing values using multiple imputation (MI) modelling utilising a Bootstrap Expectation Maximisation (BEM) algorithm implemented in the Amelia package. We can efficiently impute missing values of up to 6 missing probes, if a sample has more missing values it is said to have failed Probe QC and is not classified.

Overview

4. A multi-class optimised Support Vector Machine (SVM) validated and trained on our extensive 450k medulloblastoma cohort is used to robustly assign a subgroup to samples by their 17 β values.
   - Our SVM is validated using a bootstrapping technique via 1,000 random iterations of 80% of the training set, confidence interval derived from this is plotted on the Classification Graph as a box plot.
   - The final probability assignment for a subgroup call is made by creating an SVM model with the whole 450k training set; these probabilities are given in the Classification Table in the initial tab.
   - Calls made with a probability below our predefined threshold are considered unreliable and samples will be labeled as Unclassifiable in the Classification Table, these samples will not be plotted in the Classification Graph.

5. Various post processing and formatting operations on the data take place with the interactive website being implemented in the R Shiny reactive web application framework.

Things to consider in Shiny app development

Easiest shiny app is simply to wrap all calculations + plot in a single `renderPlot({})` function

- Will cause re-run of classification each time you resize the plot, need to **isolate calculations** from output

Speeding up code, I added `mclapply` calls in place of `lapply` where performance gains could be had (not all uses warranted `mclapply`)

- Run time down from +30 seconds to around 5 seconds

Shiny app development started after the classifier was almost finished, app code need to work like a shim around an existing code base

- **If you can develop app from ground up**

Most **R scripts not suited to an app** in terms of output weird tables and vectors only author understands, in addition no error handling either, script **run by hand line by line interactively**

More lines of code for web app now than classifier, most of which deals with applying thresholds, handling QC failure at sample level and or classification failure, and giving graceful output in addition to lots of formatting and data munging

Acknowledgments

Feature selection of informative probes from 450k cohort: **Amir Enshae** & **Reza**

Machine learning and classifier code: **Reza Rafiee**

Shiny web app code and adaptation: **Matthew Bashton**

Project concept and MassARRAY file parser: **Ed Schwalbe**