Examples With Peak Finder Script

Michael Olvera

September 8, 2016

Using Peak Finder

This is a quick guide on how to use the Peak Finder script. Feel free to follow along with the code in this file to make sure everything is working well. For this file, R code is in boxes, and outputs are boxes with ## before the code.

Before starting, include the following line of code at the top of your script:

```
source("pf/peakFinder.R")
```

Preprocessing

Samples must be preprocessed to be a dataframe with **Time** included in column 1, and all other columns samples. A preprocessed dataset is included in the zip folder:

```
example <- read.csv('pf/data/CaHandUT1.csv')
head(example) # Prints first 6 rows
```

```
##      Time       R1       R2       R3       R4       R5       R6       R7
## 1  0.0000 187.6669 387.1738 176.6555 168.4455 167.4844 179.7831 184.2782
## 2 10.0075 187.4253 387.0821 176.1735 170.1468 169.4760 180.8299 184.5747
## 3 20.0149 186.4995 387.4237 176.3546 170.1221 169.2472 181.1079 184.0144
## 4 30.0224 186.3735 386.6219 175.9589 170.0625 169.4075 180.7528 184.4155
## 5 40.0298 185.2577 386.7345 176.1445 170.1175 169.1206 180.9563 183.9717
## 6 50.0373 185.3023 385.3170 175.7450 169.8538 169.0676 180.4453 184.2486
##         R8       R9      R10      R11      R12      R13      R14
## 1 351.8155 184.4244 340.8282 176.8191 213.6833 184.8174 182.6327
## 2 352.0044 184.4453 341.1342 178.9146 212.3781 185.0817 181.5208
## 3 353.9073 184.8587 340.2362 178.8213 211.2374 184.4887 181.5315
## 4 353.3250 184.4861 337.6030 178.8120 210.5008 184.8378 180.6417
## 5 355.3002 184.7740 336.4125 178.6491 209.4161 184.0433 180.7844
## 6 354.0300 184.2582 333.7079 178.8892 208.3467 184.4472 180.0346
```

As a side note, make sure all of your comlumes are numeric/integers. A common mistake with R is that .csv files are loaded with columns being factors.

```
str(example)
```

```
## 'data.frame':    1967 obs. of  15 variables:
##  $ Time: num  0 10 20 30 40 ...
##  $ R1  : num  188 187 186 186 185 ...
##  $ R2  : num  387 387 387 387 387 ...
##  $ R3  : num  177 176 176 176 176 ...
##  $ R4  : num  168 170 170 170 170 ...
##  $ R5  : num  167 169 169 169 169 ...
##  $ R6  : num  180 181 181 181 181 ...
##  $ R7  : num  184 185 184 184 184 ...
##  $ R8  : num  352 352 354 353 355 ...
##  $ R9  : num  184 184 185 184 185 ...
```

```
##  $ R10 : num  341 341 340 338 336 ...
##  $ R11 : num  177 179 179 179 179 ...
##  $ R12 : num  214 212 211 211 209 ...
##  $ R13 : num  185 185 184 185 184 ...
##  $ R14 : num  183 182 182 181 181 ...
```

As you can see above, all columns are of **type** numeric (**num**). Looking a the second row of the output, *$ Time* indicates the title of the column, then the datatype, followed by the first few data entries.

Analysis

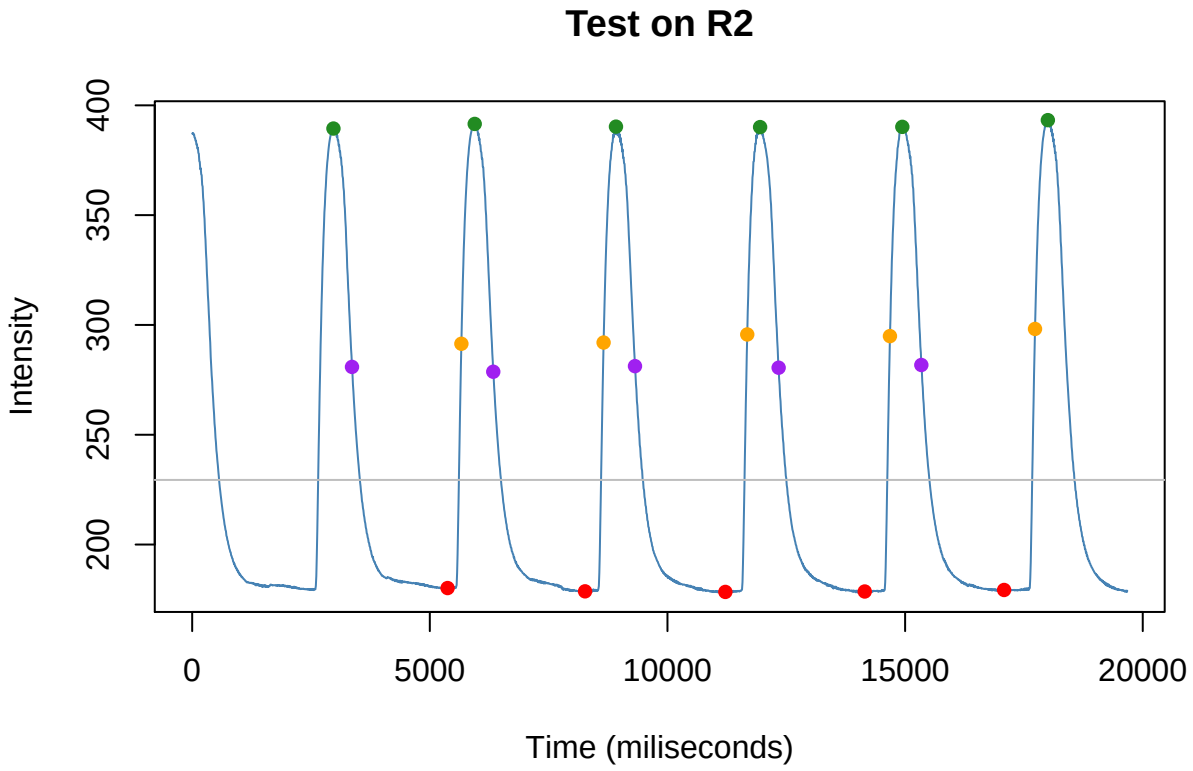
The main functions are *runTestGraph()* and *analyzeExperiemnt()*. You can call runTest() on you dataset to see how the function calls each peak.

The result will be a graph of the GcAMP flow including dots coorisponding to where the algorithm calls certain fetures. This includes:

- Peak calls (green).
- Trough calls (red).
- T50 mids (purple and red).

```
runTestGraph(example[,c(1,3)]) #test only the second sample
```

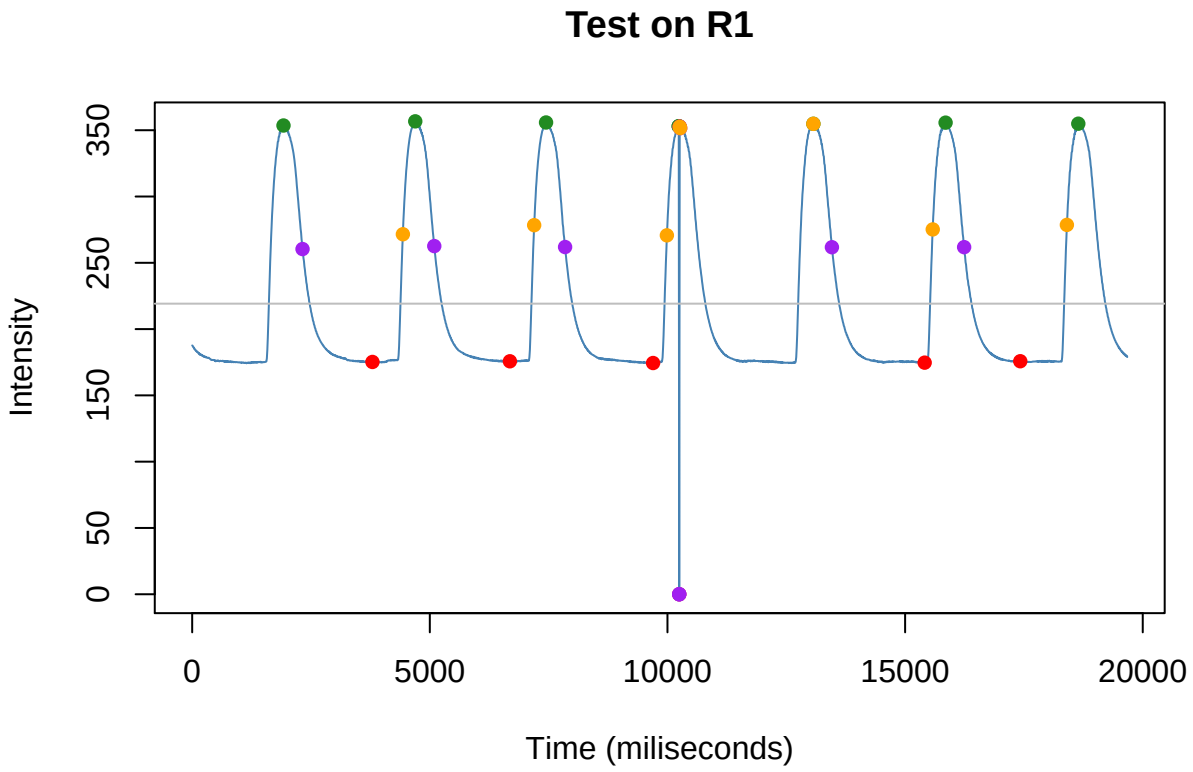
```
## No timing errors detected.
```



Looks good. Lets look at our first sample.

```
runTestGraph(example[,1:2]) #test only the first sample
```

```
## No timing errors detected.
```



Looks like there is something wrong with the data. We can run the experiment and see if the program can correct it.

```
analyzeExperiment(example)
```

```
## No timing errors detected.
## Checking sample R1
## Caution: Erronious intensity detected at index 1025.
## Attempting fix.
## Patching Sucessfull!
## Checking sample R2
## Checking sample R3
## Checking sample R4
## Checking sample R5
## Checking sample R6
## Checking sample R7
## Checking sample R8
## Checking sample R9
## Checking sample R10
## Checking sample R11
## Checking sample R12
## Checking sample R13
## Checking sample R14
```

```
##    Peak (AU) Min (AU) F/Fn (Amplitude) RightT50 (ms) LeftT50 (ms)      BPM
## 1   354.9449 175.0736         2.027404      398.6305     256.8581 25.11592
## 2   390.8022 179.0497         2.182646      394.2939     268.1999 23.95018
## 3   344.1514 173.4455         1.984204      420.3133     256.8581 24.37207
## 4   305.6978 168.5545         1.813644      405.3021     263.5297 24.37207
## 5   299.7838 166.6581         1.798795      388.6230     258.5260 24.37207
## 6   380.3940 178.8540         2.126841      391.9588     266.8656 24.23135
```

```
## 7   404.8609 182.9148         2.213385      385.2872     245.1828 24.07843
## 8   358.7807 178.4878         2.010113      408.3043     270.2014 24.86052
## 9   378.6903 183.4988         2.063721      391.9588     273.5372 23.00915
## 10  356.6691 176.0513         2.025938      396.2954     260.1939 24.17552
## 11  390.6983 178.4407         2.189513      383.1427     253.0457 26.57298
## 12  452.5925 190.3254         2.377993      400.2983     262.1954 22.88370
## 13  428.0520 182.4512         2.346118      351.9290     246.8506 24.14770
## 14  349.1165 172.8960         2.019229      411.9737     266.8656 26.23044
```

Looks like the first sample had a missing value at index 1025. The program thankfully fixed it automaticall.

The output can be saved, and extracted to be used in Excel/Prism.

```
output <- analyzeExperiment(example)
write.csv(output, "~/Desktop/results.csv")
```

If you have any questions or bugs, let me know (michael.olvera@gladstone.ucsf.edu).

```
sessionInfo()
```

```
## R version 3.3.1 (2016-06-21)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.11.3 (El Capitan)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] magrittr_1.5    formatR_1.4     tools_3.3.1     htmltools_0.3.5
##  [5] yaml_2.1.13     Rcpp_0.12.7     stringi_1.1.1   rmarkdown_1.0
##  [9] knitr_1.14      stringr_1.1.0   digest_0.6.10   evaluate_0.9
```