Analyzing Olfactory Response Data in ABF Files

Stowers Institute for Medical Research
R/Bioconductor Discussion Group

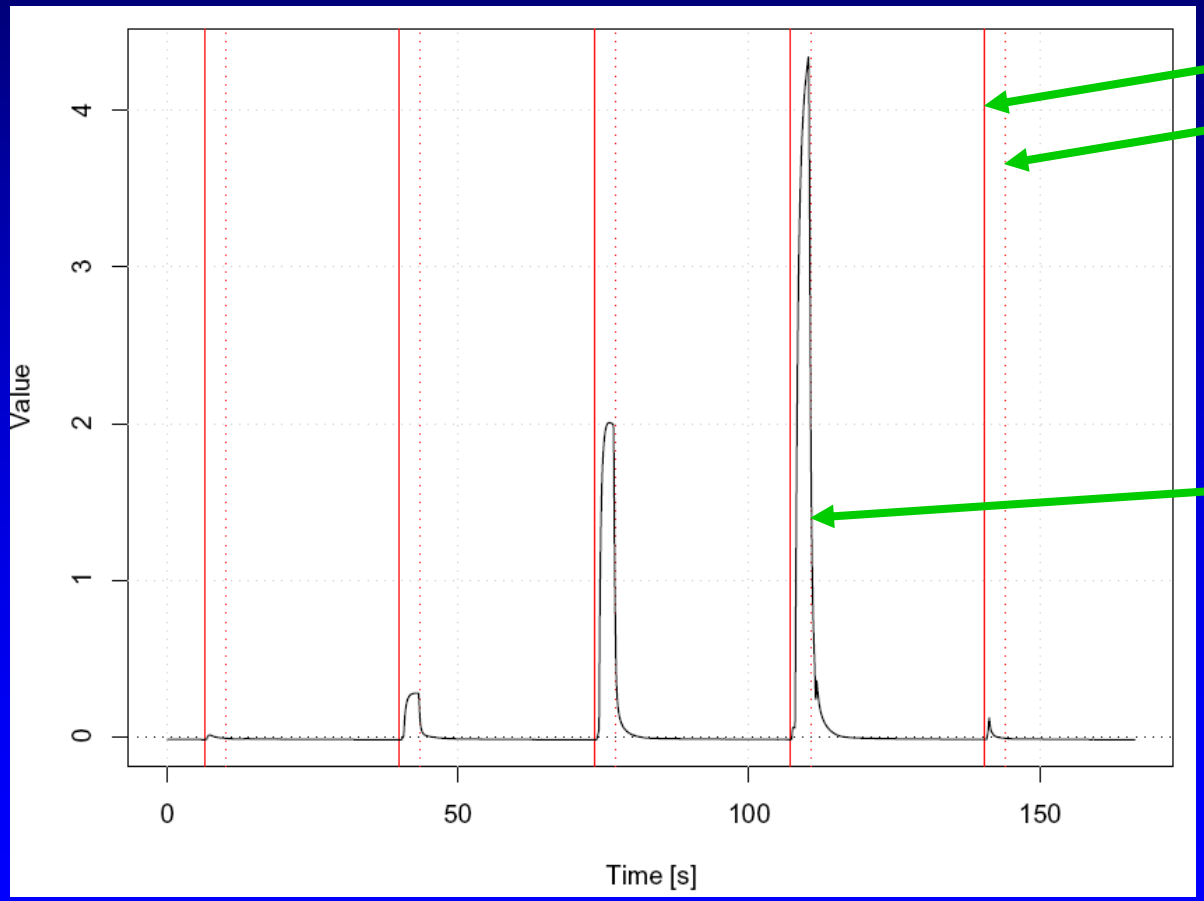
Earl F. Glynn
Scientific Programmer
25 Sept 2008

Analyzing ABF Files

- Converting Axon Binary File to CSV
- Processing directory of ABF files in R
- Finding events in TTL channel
- Finding features and areas in olfactory response “Value” channel
- Reporting results in file and charts

Analyzing ABF Files

U:\efg\Research\RonYu\Limei\R\071225Z-AA-200cm-3s.pdf



Odorant "on"

Odorant "off"

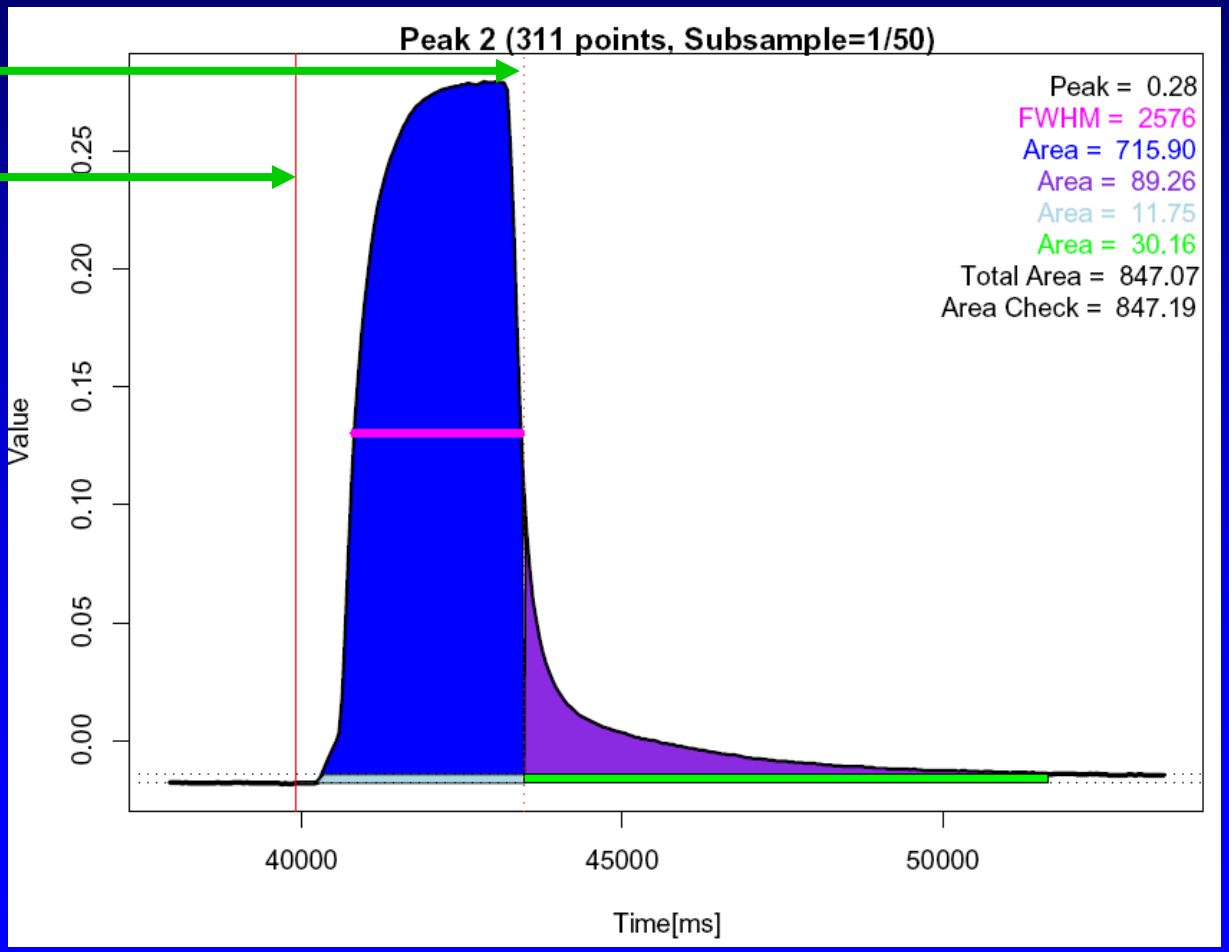
Olfactory response pulse

Analyzing ABF Files

U:\efg\Research\RonYu\Limei\R\071225Z-AA-200cm-3s.pdf

Odorant "off"

Odorant "on"



Olfactory response pulse

Converting Axon Binary File to CSV

U:\efg\Research\RonYu\ABF\Converting-abf-to-csv.doc

- Developed “C” program to extract data from binary ABF file
- Only process “Gap free files” at present
- Only process needed subset of data
- Requires proprietary abffio.dll at run-time
- CSV files are ~4X larger than ABF files
- Many programs can read CSVs; few programs can read ABFs

Converting Axon Binary File to CSV

U:\efg\Research\RonYu\Limei\071225-Length

Use system to run command-line program

e.g., abf2csv 071225A-AA-25cm-1s.abf 071225A-AA-25cm-1s.csv

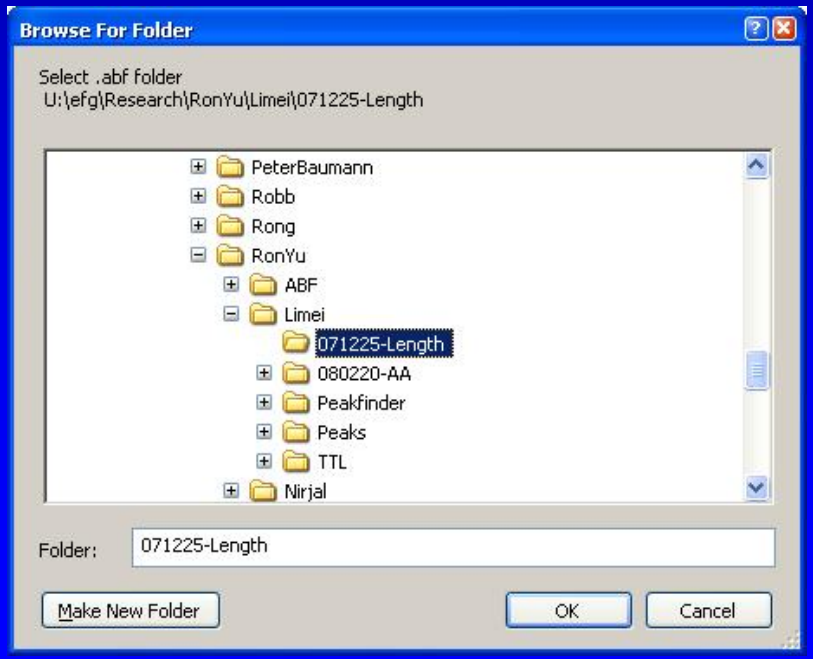
```
read.abf.file <- function(abffile, delete.csv=TRUE)
{
  basedir <- substr(abffile,1,nchar(abffile)-3)
  csvfile <- paste(basedir, "csv", sep="")

  system(paste("abf2csv", abffile, csvfile))

  raw <- read.csv(csvfile, as.is=TRUE)
  colnames(raw) <- c("Value", "TTL")
  raw$Time <- 0:(nrow(raw)-1)

  if (delete.csv)
  {
    file.remove(csvfile)
  }

  return(raw)
}
```

CSV files from ABFs can have too many lines for Excel: e.g., 176,377 in the case above

Processing directory of files in R

U:\efg\Research\RonYu\Limei\R\ComputeAreas.R

```
pick.and.process.directory <- function()
{
  abf.folder  <- choose.dir(default = "U:/efg/Research/RonYu/Limei/",
                            caption = "Select .abf folder")
  process.directory(abf.folder)
}
```

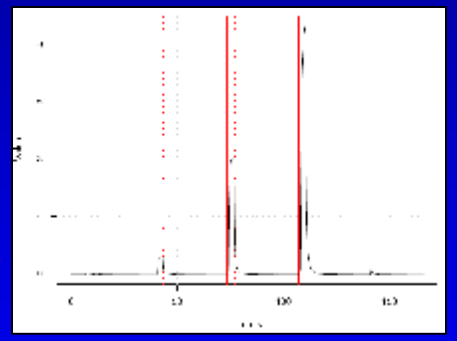


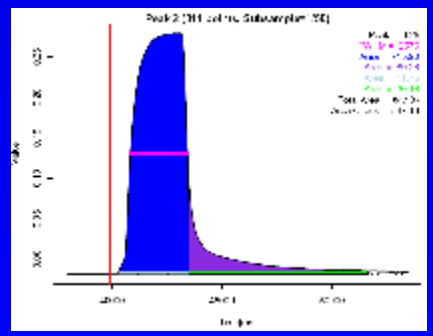
Processing directory of ABF files in R

```
process.directory <- function(abf.folder)
{
  abflist <- dir(path=abf.folder, pattern="\\.abf$")

. . .

  for (file.index in 1:length(abflist))
  {
    abffile <- file.path(abf.folder, abflist[file.index])
. . .

  }

. . .
}
```

Processing directory of files in R

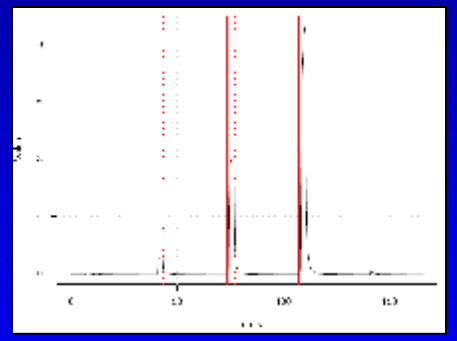
```
process.directory <- function(abf.folder)
{
  abflist <- dir(path=abf.folder, pattern="\\.abf$")
  basefile <- substr(abflist,1,nchar(abflist)-3)
  pdflist  <- paste(basefile, "pdf", sep="")

  . . .

  for (file.index in 1:length(abflist))
  {
    abffile <- file.path(abf.folder, abflist[file.index])
    raw <- read.abf.file(abffile)
    TTL <- process.TTL.data(raw)

    subsample <- subsample.raw.data(raw, SUBSAMPLE.FACTOR)

    pdf(file.path(abf.folder, pdflist[file.index]), width=8, height=10)
      par(oma=c(2,0,3,0))
      plot.subsample(subsample, TTL)
      plot.header.and.footer(abffile)

      for (i in 1:length(TTL$start))
      {
        peak.results <- plot.peak(abffile, i, subsample, TTL)
        plot.header.and.footer(abffile)

        . . .
      }

    dev.off()

  }
  . . .
}
```

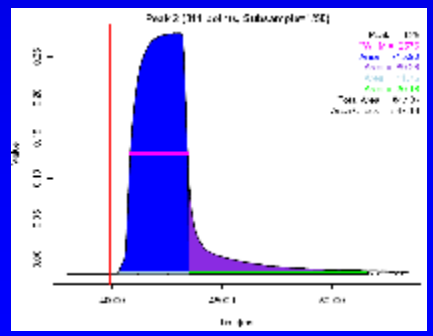




Processing directory of files in R

```
process.directory <- function(abf.folder)
{
  abflist <- dir(path=abf.folder, pattern="\\.abf$")
  basefile <- substr(abflist,1,nchar(abflist)-3)
  pdflist  <- paste(basefile, "pdf", sep="")

  . . .

  for (file.index in 1:length(abflist))
  {
    abffile <- file.path(abf.folder, abflist[file.index])
    raw <- read.abf.file(abffile)
    TTL <- process.TTL.data(raw)

    subsample <- subsample.raw.data(raw, SUBSAMPLE.FACTOR)

    pdf(file.path(abf.folder, pdflist[file.index]), width=8, height=10)
      par(oma=c(2,0,3,0))
      plot.subsample(subsample, TTL)
      plot.header.and.footer(abffile)

      for (i in 1:length(TTL$start))
      {
        peak.results <- plot.peak(abffile, i, subsample, TTL)
        plot.header.and.footer(abffile)

        . . .
      }

    dev.off()

  }
  . . .
}
```



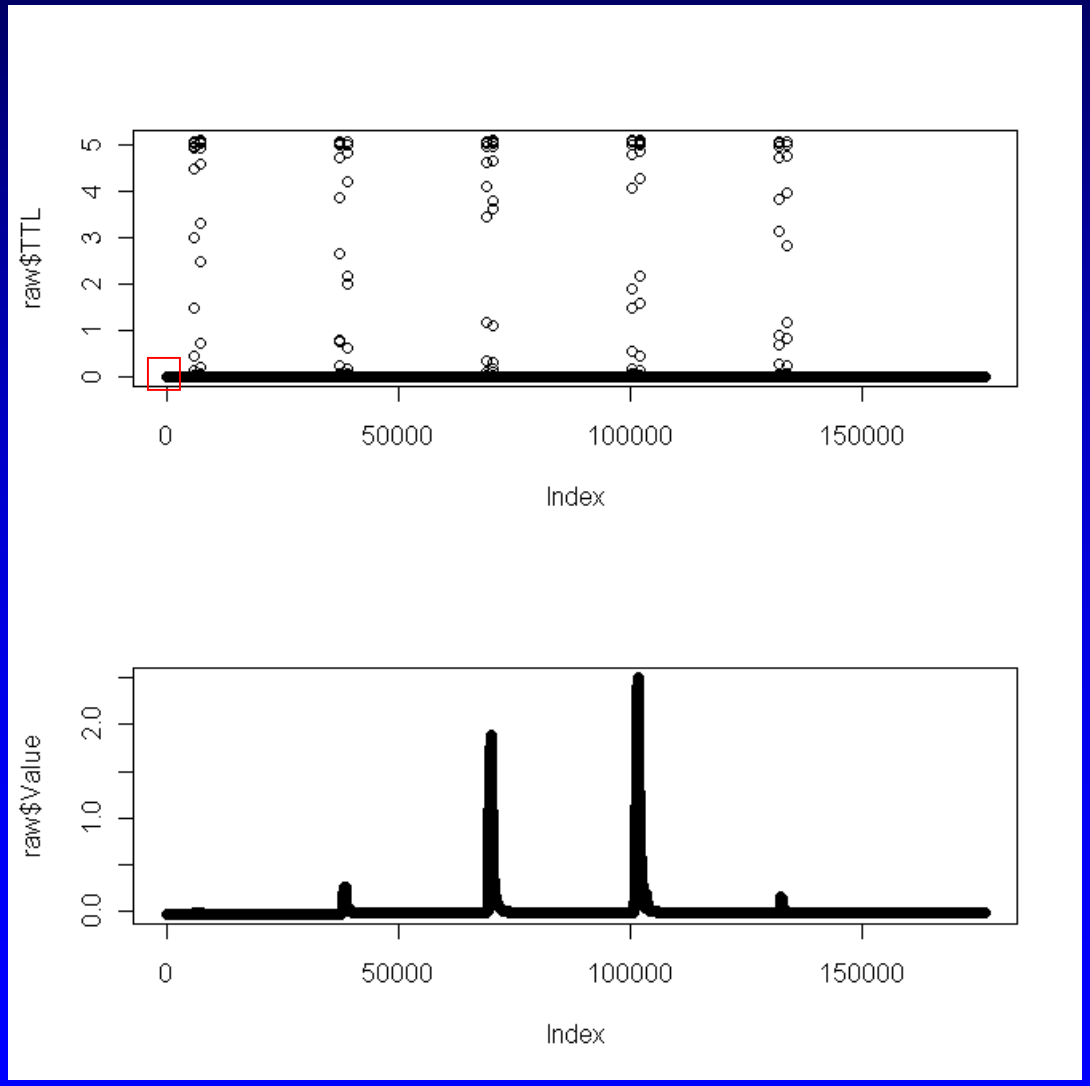


Finding events in TTL channel

```
raw <- read.abf.file(abffile)
```

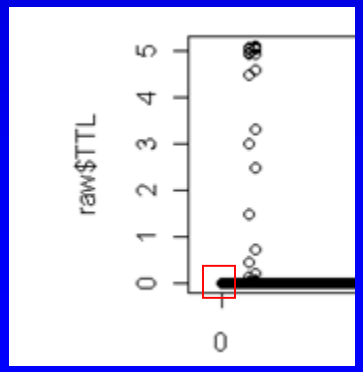
```
par(mfrow=c(2,1))
plot(raw$TTL)
plot(raw$Value)
```

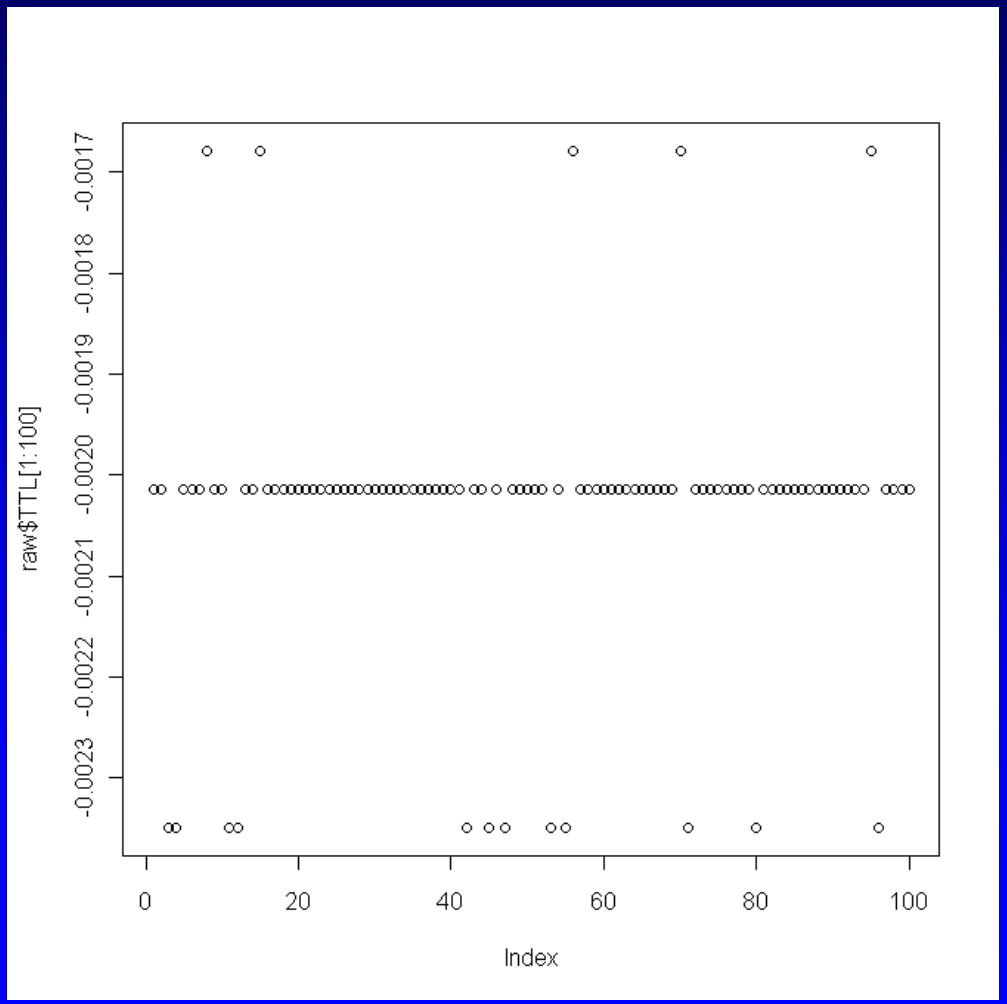
Use TTL events to process peaks in olfactory response “Value” data



Finding events in TTL channel

```
plot(raw$TTL[1:100])
```



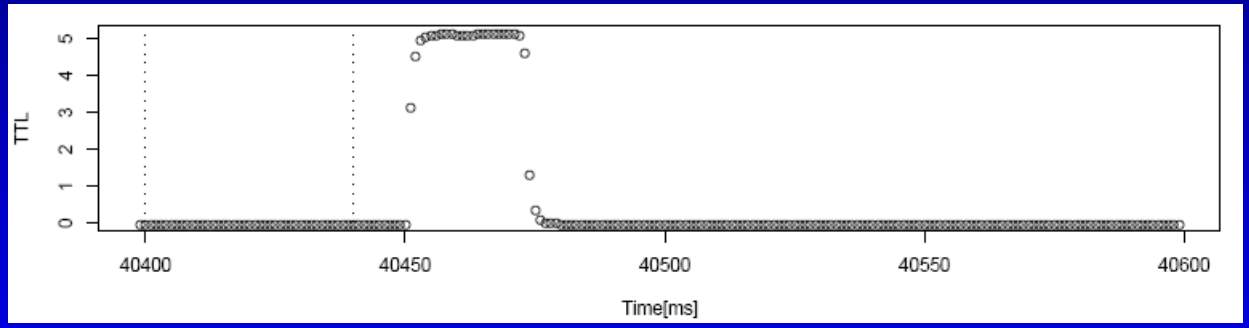


Finding events in TTL channel

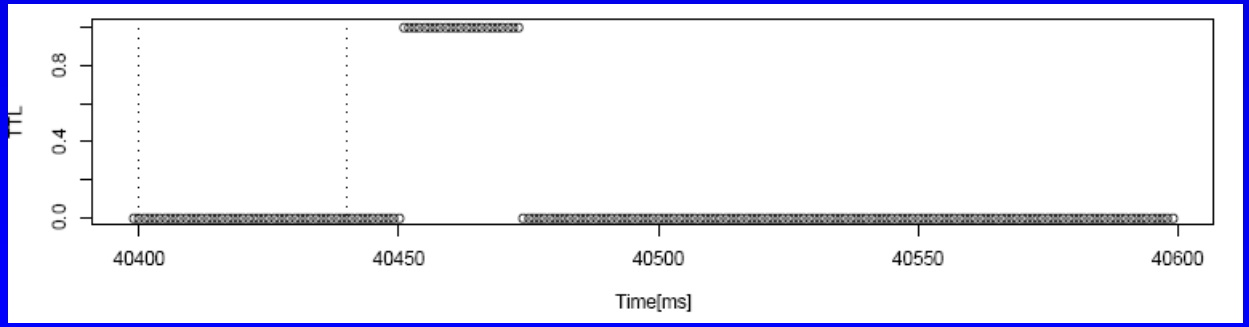
```
TTL.midpoint <- (max(raw$TTL) - min(raw$TTL)) / 2
```

```
fixed.TTL <- ifelse(raw$TTL <= TTL.midpoint, 0, 1)
```

Raw



Fixed

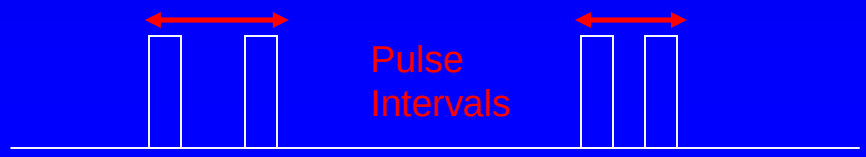


Finding events in TTL channel

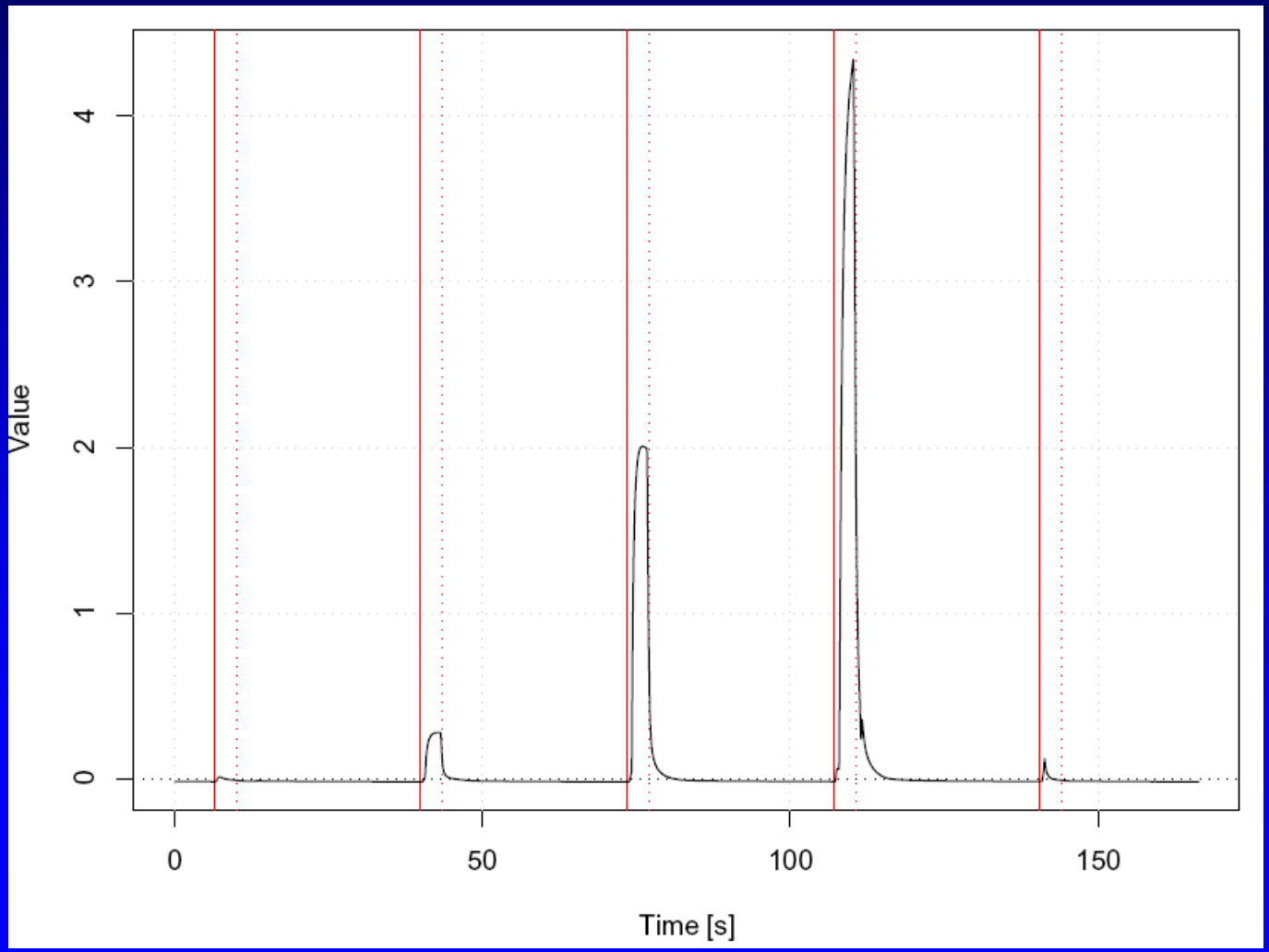
Intervals of Interest

```
TTL.delta <- c(diff(fixed.TTL), 0)

TTL.up    <- which(TTL.delta ==  1)
TTL.up
[1]   5835   7384  37388  38938  68941  70490 100494 102043 132045 133595

TTL.down  <- which(TTL.delta == -1)
TTL.down
[1]   5847   7400  37402  38949  68953  70505 100510 102055 132057 133607

plot(fixed.TTL[5825:5857])
```

```
interval.start <-  TTL.up[2*( 1:(length(TTL.up)   %/%2) )-1]
interval.start
[1]   5835  37388  68941 100494 132045

interval.stop  <-  TTL.down[2*( 1:(length(TTL.down) %/%2) )  ]
interval.stop
[1]   7400  38949  70505 102055 133607
```

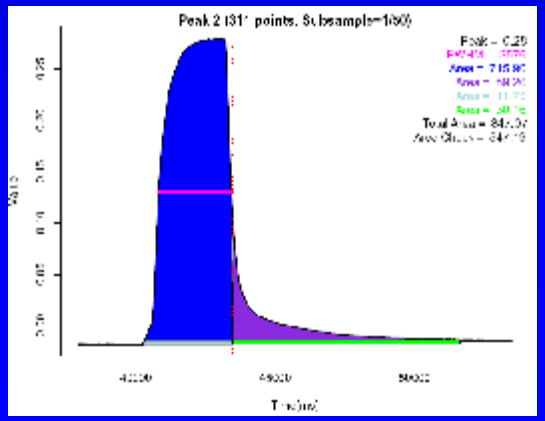


Finding events in TTL channel



Finding features and areas in Value channel

- Need high sampling rate for exact timing of events in TTL channel
- Do not need high sampling rate for most olfactory response features in the Value channel, e.g., area.
- Considerable speedup after subsampling



Subsampling

```
SUBSAMPLE.FACTOR    <- 50
```

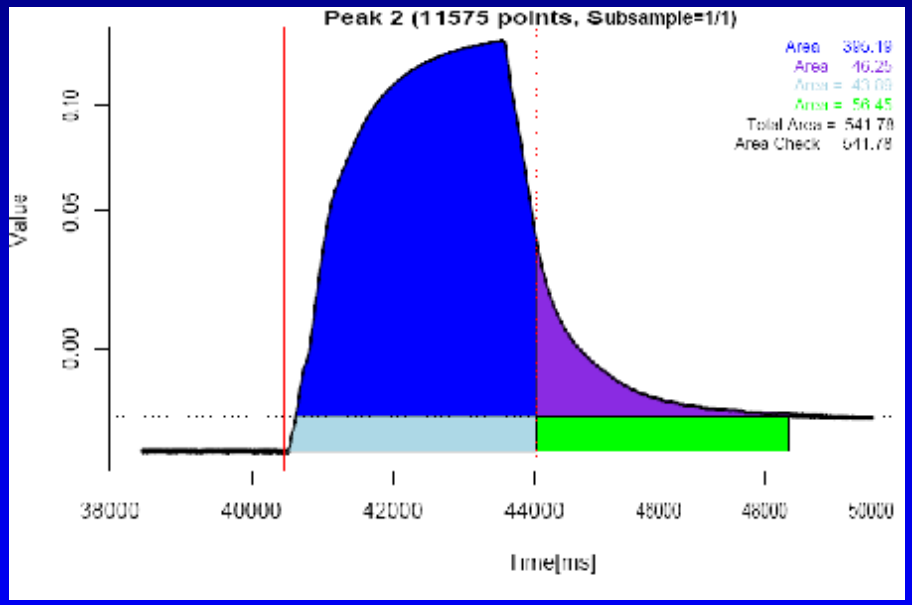
```
subsample <- subsample.raw.data(raw, SUBSAMPLE.FACTOR)
```

```
subsample.raw.data <- function(raw, subsample.frequency)
{
  # Subsample data -- millisecond resolution too high
  Subsample.Time  <- subsample.frequency * 0:((nrow(raw) %/% subsample.frequency) - 1)
  Subsample.Value <- raw$Value[Subsample.Time]

  return(list(Time=Subsample.Time, Value=Subsample.Value, frequency=subsample.frequency))
}
```

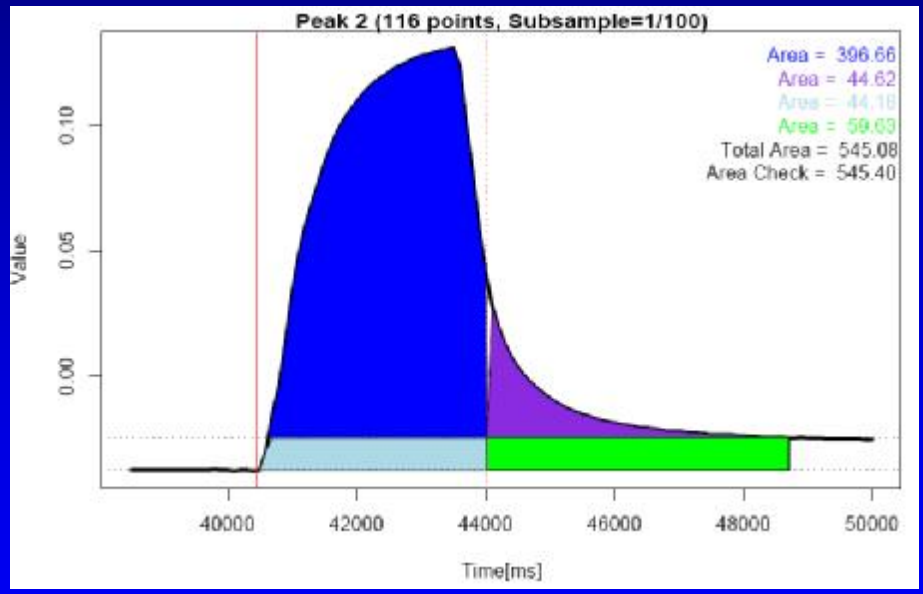
Pick every 50th point to speed up processing.
Output PDFs are much smaller with subsampling.

How does Subsampling affect area?

11,575 points



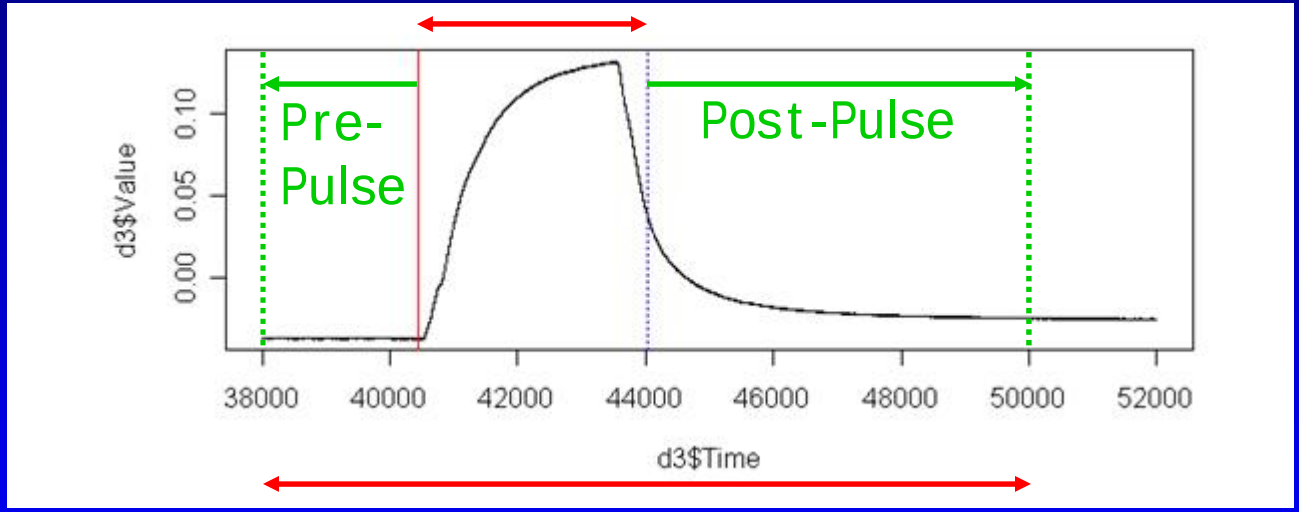
116 points



<1% difference on large areas
up to 6% difference on small areas

Finding features and areas in Value channel

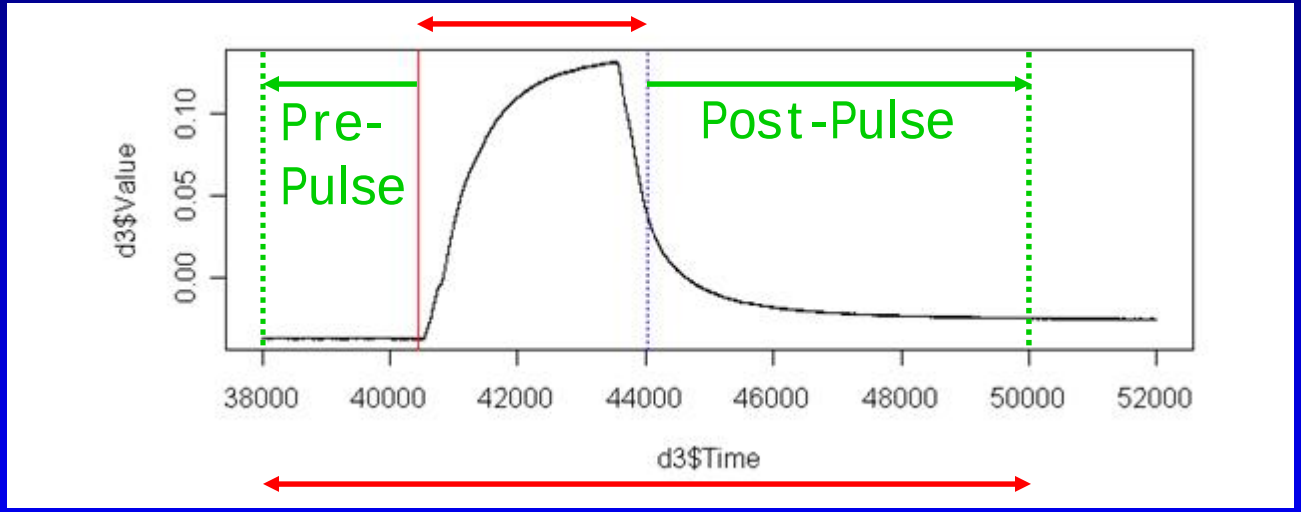
Odorant Pulse



Analysis Interval

Finding features and areas in Value channel

Odorant Pulse



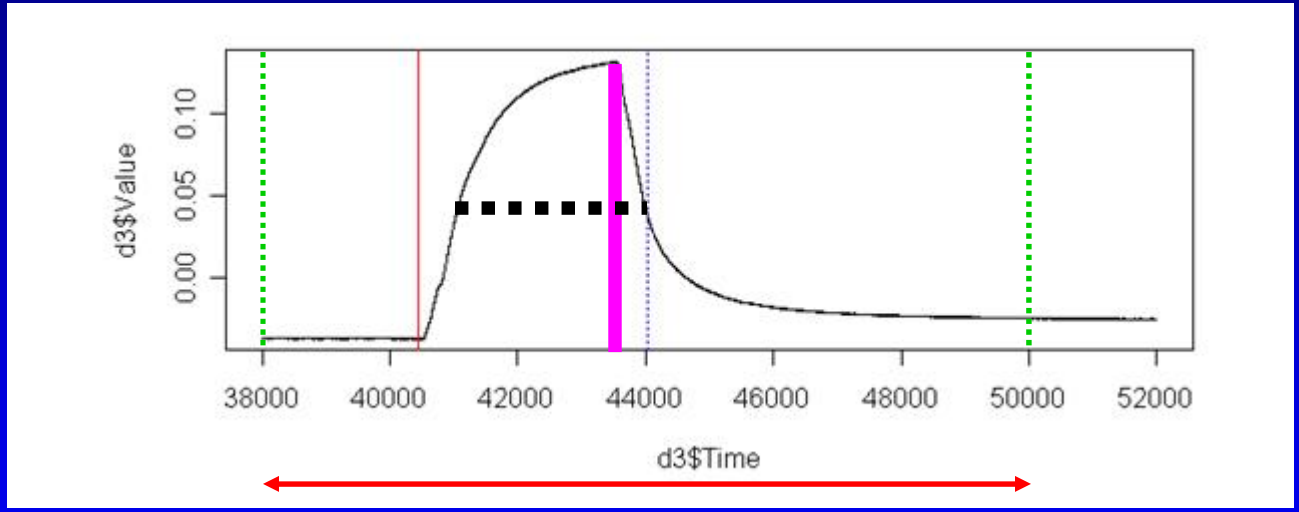
Analysis Interval

```
show.start <- TTL$start[i] - PREPULSE.PERIOD *SAMPLING.FREQUENCY
show.stop  <- TTL$stop[i]  + POSTPULSE.PERIOD*SAMPLING.FREQUENCY
interval.range <- (subsample$Time >= show.start) &
                  (subsample$Time <= show.stop)
```

```
interval.Time  <- subsample$Time[interval.range]
interval.Value <- subsample$Value[interval.range]
```

Finding features and areas in Value channel

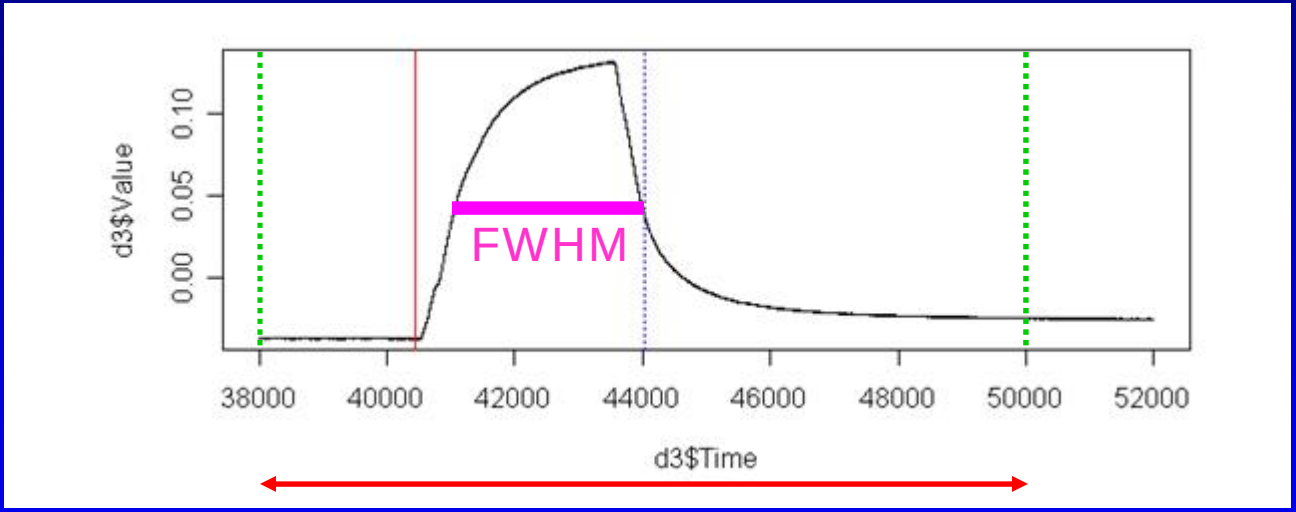
Peak and Full-Width at Half Max (FWHM)




```
# Compute FWHM
Value.max <- max(interval.Value)
Value.min <- min(interval.Value)

Value.halfmax <- (Value.max + Value.min) / 2
# Pick first if more than one
Index.max <- which(interval.Value == Value.max)[1]
```

Finding features and areas in Value channel

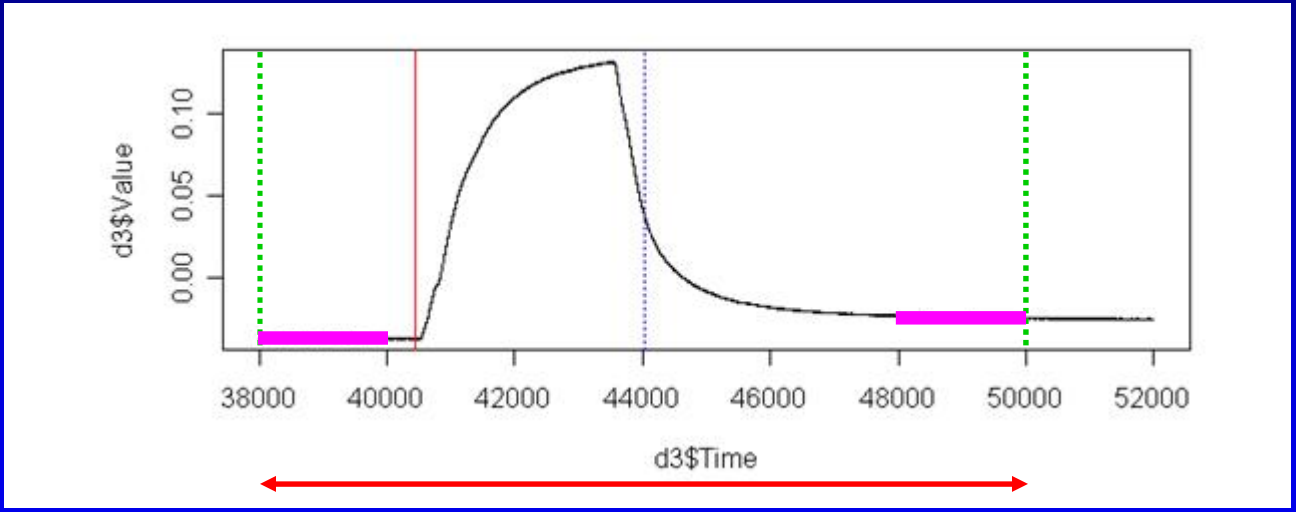
Full-Width at Half Max (FWHM)




```
suppressWarnings(
  side.left  <- approx(interval.Value[1:Index.max],
                       interval.Time[1:Index.max], Value.halfmax))

suppressWarnings(
  side.right <- approx(interval.Value[Index.max:length(interval.Value)],
                       interval.Time[Index.max:length(interval.Value)],
                       Value.halfmax))

FWHM <- side.right$y - side.left$y
```

Finding features and areas in Value channel

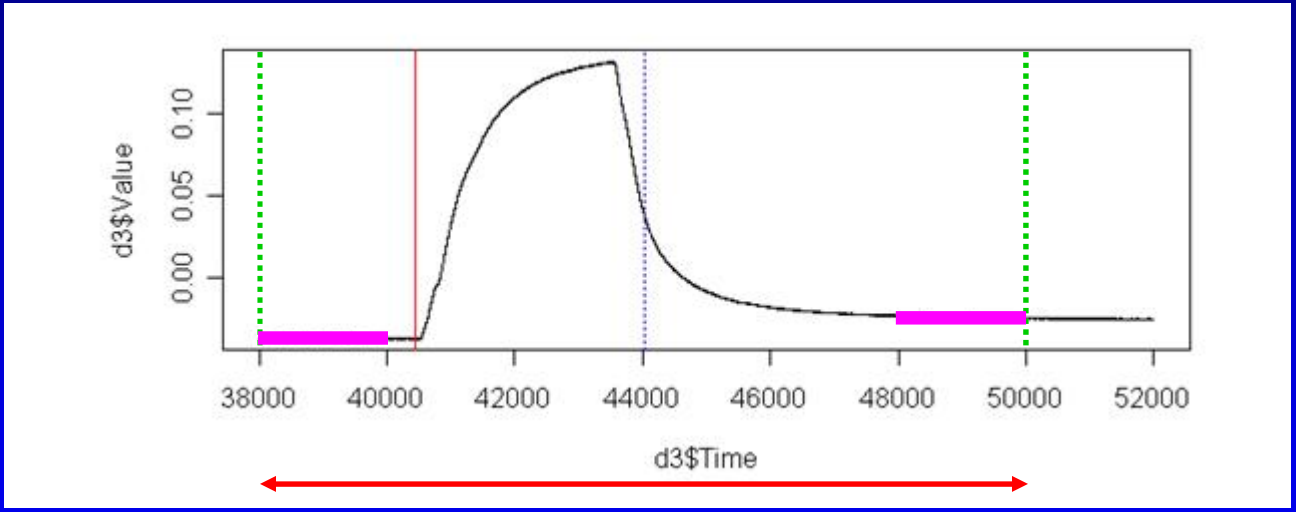
Left and Right "Plateaus"




```
plateau.points <- 1:(2*SAMPLING.FREQUENCY %/% SUBSAMPLE.FACTOR)
plateau.left   <- median(interval.Value[plateau.points])
plateau.right  <- median(interval.Value[length(interval.Value) -
                         plateau.points + 1])
plateau.delta  <- plateau.right - plateau.left
plateau.max    <- max(plateau.left, plateau.right)
```

Finding features and areas in Value channel

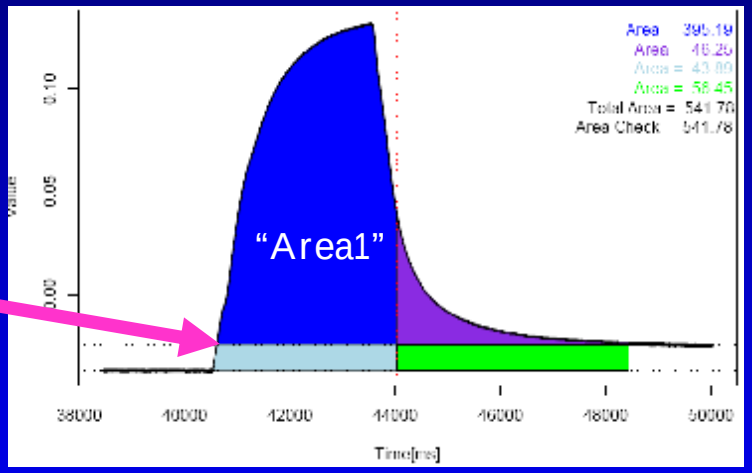
Left and Right "Plateaus"




```
plateau.points <- 1:(2*SAMPLING.FREQUENCY %/% SUBSAMPLE.FACTOR)
plateau.left   <- median(interval.Value[plateau.points])
plateau.right  <- median(interval.Value[length(interval.Value) -
                         plateau.points + 1])
plateau.delta  <- plateau.right - plateau.left
plateau.max    <- max(plateau.left, plateau.right)
```

Finding features and areas in Value channel

Areas



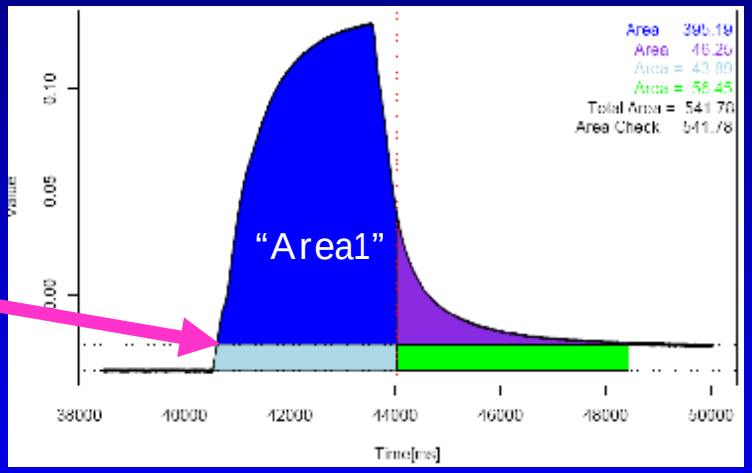

```
### Area 1
response.Range <- (interval.Time >= TTL$start[i]) &
                  (interval.Time <= TTL$stop[i])
response.Time  <- interval.Time[response.Range]
response.Value <- interval.Value[response.Range]

first.above  <- which( response.Value > plateau.right)[1]
```

```
corner <- approx(response.Value[(first.above-1):first.above],
                 response.Time[(first.above-1):first.above],
                 plateau.right)
```

Finding features and areas in Value channel

Areas

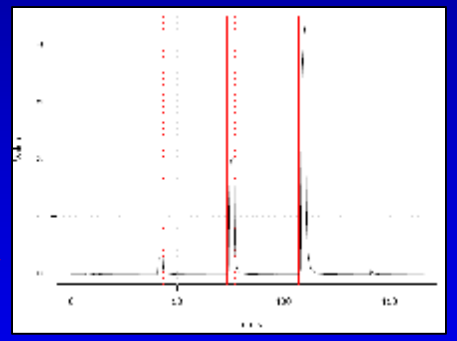
"corner"

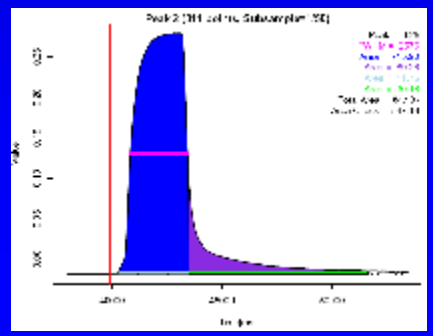



```
response.Subrange <- (response.Time >= corner$y) & (response.Time <= TTL$stop[i])

polygon1.x <- c(response.Time[response.Subrange],  TTL$stop[i],   corner$y)
polygon1.y <- c(response.Value[response.Subrange], plateau.right, plateau.right)

polygon(polygon1.x, polygon1.y, col="blue")
```

Reporting results in charts

```
process.directory <- function(abf.folder)
{
  abflist <- dir(path=abf.folder, pattern="\\.abf$")
  basefile <- substr(abflist,1,nchar(abflist)-3)
  pdflist  <- paste(basefile, "pdf", sep="")

  . . .

  for (file.index in 1:length(abflist))
  {
    abffile <- file.path(abf.folder, abflist[file.index])
    raw <- read.abf.file(abffile)
    TTL <- process.TTL.data(raw)

    subsample <- subsample.raw.data(raw, SUBSAMPLE.FACTOR)

    pdf(file.path(abf.folder, pdflist[file.index]), width=8, height=10)
      par(oma=c(2,0,3,0))
      plot.subsample(subsample, TTL)
      plot.header.and.footer(abffile)

      for (i in 1:length(TTL$start))
      {
        peak.results <- plot.peak(abffile, i, subsample, TTL)
        plot.header.and.footer(abffile)

        . . .
      }

    dev.off()

  }
  . . .
}
```





Reporting results in file

```
process.directory <- function(abf.folder)
{
  abflist <- dir(path=abf.folder, pattern="\\.abf$")
  basefile <- substr(abflist,1,nchar(abflist)-3)
  pdflist  <- paste(basefile, "pdf", sep="")

  All.Results <- NULL

  for (file.index in 1:length(abflist))
  {
    abffile <- file.path(abf.folder, abflist[file.index])
    cat( format(Sys.time(), "%Y-%m-%d %H:%M:%S"), " Reading", abffile, "\n")
    flush.console()
    raw <- read.abf.file(abffile)
    TTL <- process.TTL.data(raw)

    subsample <- subsample.raw.data(raw, SUBSAMPLE.FACTOR)

    pdf(file.path(abf.folder, pdflist[file.index]), width=8, height=10)
      par(oma=c(2,0,3,0))  # Leave room for footer
      plot.subsample(subsample, TTL)
      plot.header.and.footer(abffile)

      for (i in 1:length(TTL$start))
      {
        peak.results <- plot.peak(abffile, i, subsample, TTL)
        plot.header.and.footer(abffile)

        All.Results <- rbind(All.Results, peak.results)
      }

    dev.off()

  }
  cat( format(Sys.time(), "%Y-%m-%d %H:%M:%S"), "\n")

  write.csv(All.Results, file=file.path(abf.folder,"PeakSummary.csv"), row.names=FALSE)
}
```

Reporting results in file

FilePeakSummary.csv

	A	B	C	D	E	F	G	H	I	J
1	file	peak.index	peak	FWHM	area1	area2	area3	area4	area.total	area.check
2	071225A-AA-25cm-1s-clamp9.abf	1	0.0	2541.2	3.0	1.1	2.4	2.5	9.0	9.0
3	071225A-AA-25cm-1s-clamp9.abf	2	0.3	717.9	188.6	42.9	9.3	34.7	275.6	276.0
4	071225A-AA-25cm-1s-clamp9.abf	3	1.9	769.0	1420.8	213.8	15.0	52.0	1701.6	1701.6
5	071225A-AA-25cm-1s-clamp9.abf	4	2.5	674.6	1639.8	426.5	13.1	46.1	2125.5	2125.6
6	071225A-AA-25cm-1s-clamp9.abf	5	0.2	444.6	87.0	12.7	3.9	12.1	115.6	115.6
7	071225A-AA-25cm-1s.abf	1	0.0	2541.2	3.0	1.1	2.4	2.5	9.0	9.0
8	071225A-AA-25cm-1s.abf	2	0.3	717.9	188.6	42.9	9.3	34.7	275.6	276.0
9	071225A-AA-25cm-1s.abf	3	1.9	769.0	1420.8	213.8	15.0	52.0	1701.6	1701.6
10	071225A-AA-25cm-1s.abf	4	2.5	674.6	1639.8	426.5	13.1	46.1	2125.5	2125.6
11	071225A-AA-25cm-1s.abf	5	0.2	444.6	87.0	12.7	3.9	12.1	115.6	115.6

Several improvements in charts and output file are under consideration.

Take Home: "R" and Analysis Tips

- system
- processing directory of files
  - choose.dir
  - dir
  - file.path
  - rbind (form composite data.frame)
  - write.csv
- cleanup noise in data: threshold, median
- subsampling data
- area computations
- approx to interpolate
- suppressWarnings
- FWHM (full width at half max)
- polygon

Acknowledgments

Yu Lab

- Nirjal Sapkota
  (now at North Carolina State University)
- Limei Ma
- Ron Yu