meadq package: From A to Z

Diana Hall

June 2, 2014

Installation

To install this package, and then view this introductory vignette, do:

```
# provide local file path to tar.gz, e.g.
tarfile.path = "F:/R/Rpackage_meadq/meadq_1.0.1.tar.gz"
install.packages(pkgs = tarfile.path, type = "source", repos = NULL)
vignette("meadq-intro", package = "meadq")
```

Alternatively, if you want to get the latest version of the package, you can install it in the following manner:

```
install.package("devtools")  # note: need R v.>=3.1.0
# to install devtools, use CRAN
require(devtools)
install_github("dianaransomhall/meadq")
```

You can browse the package sources via the URL `http://github.com/dianaransomhall/meadq`.

Setup

This file is a vignette, written in R, as a reproducible research document.

```
require(meadq)
require(knitr)
opts_chunk$set(cache = TRUE)
opts_chunk$set(dev = "pdf")
```

Introduction

This is a short introduction to the abilities of the meadq package for analysis of multielectrode array data. It is not a comprehensive guide, but simply gives examples of what can be done with the package. The package contains some example data sets which are used here to demonstrate various routines.

Help pages

A list of help pages associated with the package is given by `help(package='meadq')` command:

creating .h5 Files from text files of spike trains

Convert the recorded data, which may be in a proprietary or otherwise non universal file type, into a type which may be read in by various softwares. Following suggestions from the "program on Standards for Datasharing" by the International neuroinformatics coordinating facility `datasharing.incf.ord/ep/HDF5_data_standard`, code is provided to convert axion alpha map files to .h5 files that link spike train to meta-data about the experiment. Current functionality is restricted to experimental meta data limited to a handful of variabes. However, additional information may be built in by creating a new function.

```
# For example, choose .mapTimestamps:
print(system.file("extdata", "ON_20140205_MW1007-26_DIV07_001.mapTimestamps",
    package = "meadq"))
make.axion.map.to.h5.dh()
```

0.1 What is the “s[[i]]” object?

A convention of the program is that all data referring to a recording is stored within an object of class `mm.s`, which is actually a list. So, when new data/results are collected for a recording, I tend to add the new information into that object (e.g. see how burst analysis results are stored).

The most important items in the list are:

NCells The number of units in the recording.

rec.time The start and end time of the recording.

spikes A list of vectors. Element i of the list is the vector of spike trains for unit i. Each spike train is ordered, smallest first.

nspikes A vector. $nspikes[i]$ is the number of spikes in train i.

layout Information regarding the spatial layout of the units.

create well summary spreadsheet

```
data.file1 <- system.file("extdata", "ON_20140205_MW1007-26_DIV05_001.h5", package = "meadq")
data.file2 <- system.file("extdata", "ON_20140205_MW1007-26_DIV07_001.h5", package = "meadq")
data.file3 <- system.file("extdata", "ON_20140205_MW1007-26_DIV09_001.h5", package = "meadq")
h5Files = c(data.file1, data.file2, data.file3)

create_burst_ont_Data(h5Files, save.rdata = TRUE)
```

A .csv file has been created in the library where this package is located in a folder called "prepared$_d ata''.Twowelllevelsummariesarecreated : 1.wellsummariescomputedacrossallAE(activeelectrodes :=>= 5spikes/min)2.wellsummariescomputedacrossallABE(activeburstingelectrodes :=>= 1burst/min).Uselibraryfilepath.libPat$
$TRUE, the.h5fileshavebeenconvertedintoalistobjects[[i]], (i = filenumber)thatcanbeeasilyloadedandmanipulated$

```
# to load data use the 'load' function, type ?load for more info lets use an
# example for now
data("example_ont_data")
```

Burst analysis

Burst identification is done with:

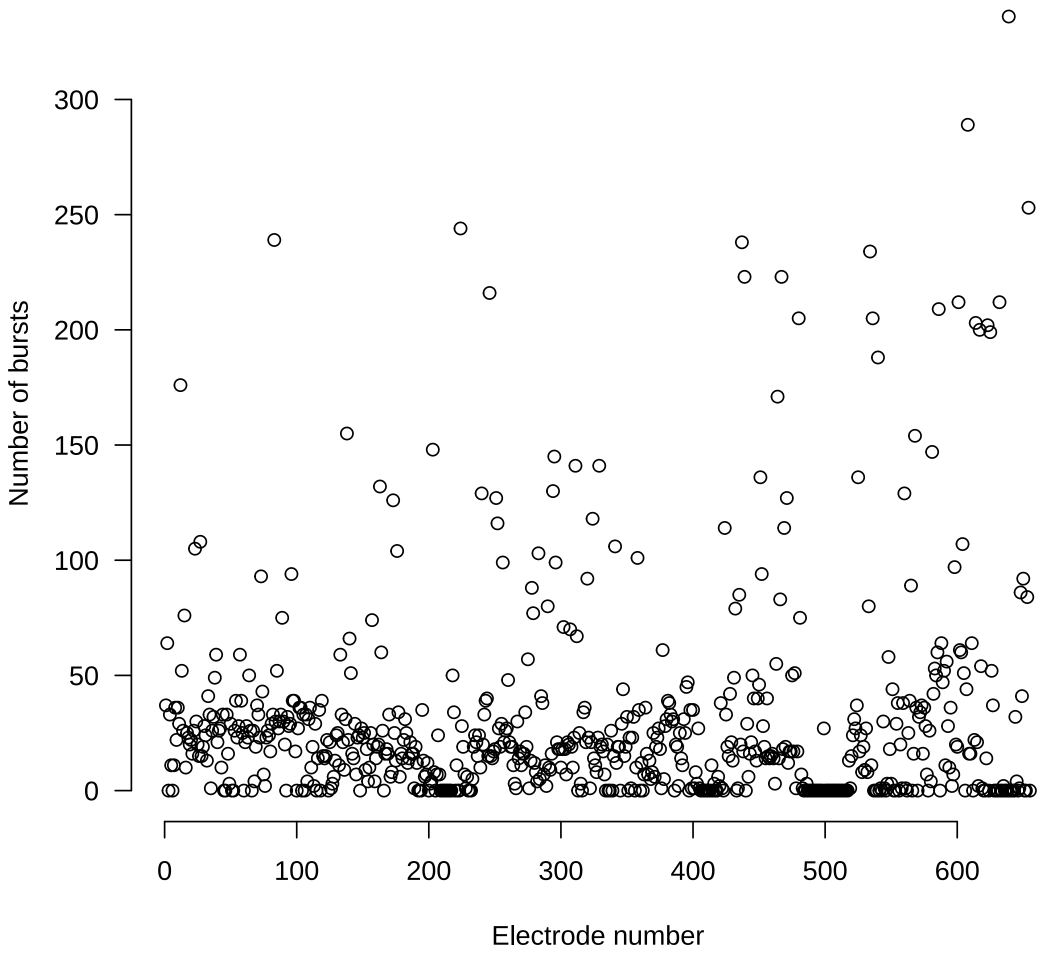
1. Max Interval method, as described by Neuroexplorer (**?**)

The MaxInterval method.

```
data.file <- system.file("data", "example_ont_data.rda", package = "meadq")
load(data.file)
```

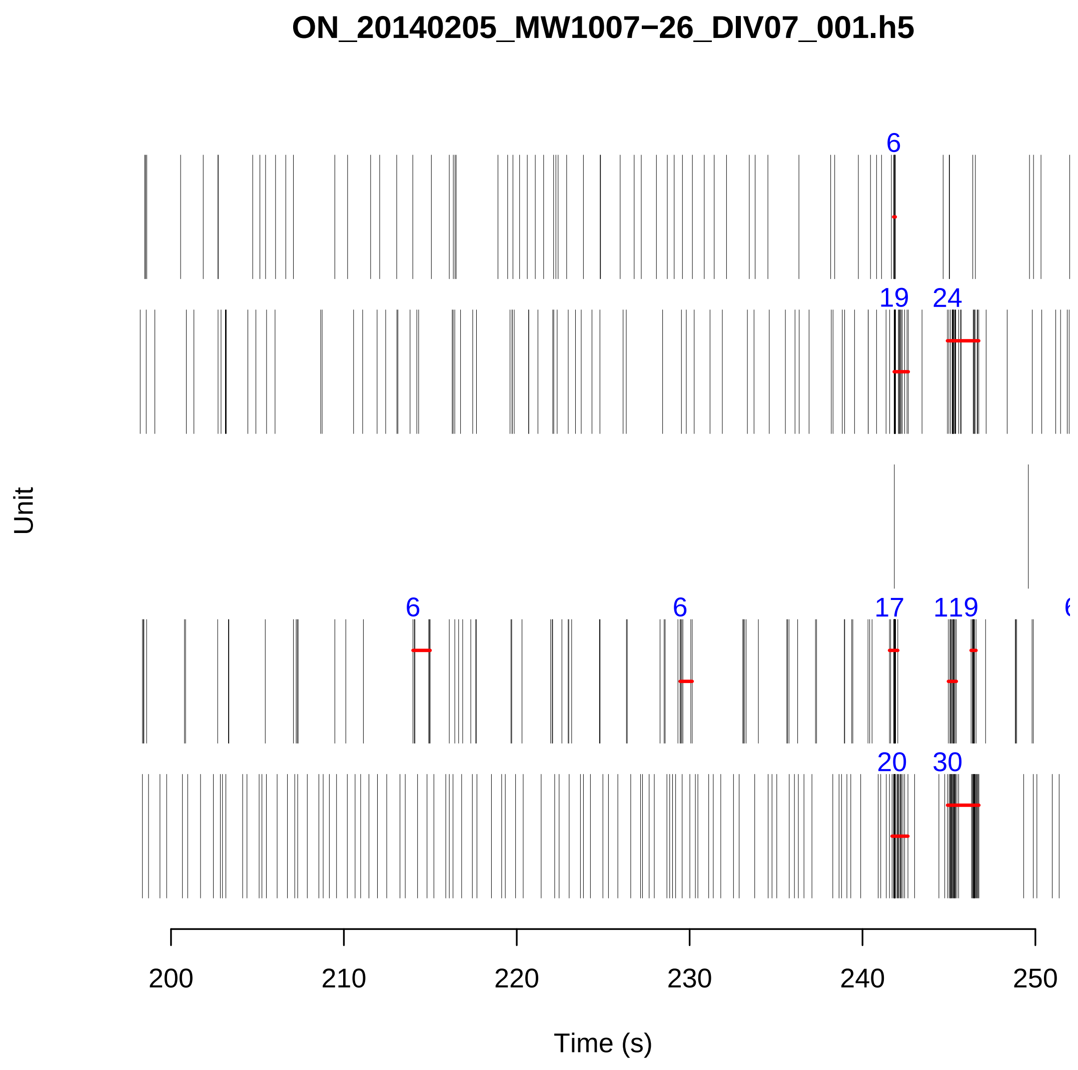
So, for example, for electrode 2, we see the following bursts (just taking the head as there are many of them. We can also easily plot the number of bursts on each electrode.

```
s[[3]]$channels[[2]]

## [1] "A1_12"

head(s[[3]]$allb[[2]])

##      beg end    IBI len   durn mean.isis SI
## [1,]   6  22     NA  17 0.3415   0.02135  1
## [2,]  27  31  4.905   5 0.2251   0.05628  1
## [3,]  36  42  2.818   7 0.8008   0.13347  1
## [4,]  55  76  9.969  22 0.6484   0.03088  1
## [5,]  98 103 21.426   6 0.9942   0.19885  1
## [6,] 129 134 14.458   6 0.6998   0.13997  1

nbursts <- sapply(s[[3]]$allb, nrow)
plot(nbursts, xlab = "Electrode number", ylab = "Number of bursts", bty = "n",
    las = 1)
```



Once bursts are computed the resulting burst information can be visualized on a raster assuming that the burst information is stored in the `s$allb` component of the object. Here we ask to see the burst information for twenty seconds of data from just the first five trains.

```r
plot(s[[3]], beg = 200, end = 250, show.bursts = TRUE, whichcells = 1:5)
```



Bursts are indicated with a red horizontal line, and the blue number indicates the number of spikes in the burst.

Note: a Hidden-Markov Model (HMM) for burst analysis in R (**?**) is available in the following package: `http://www.stat.duke.edu/~st118/Software/`.

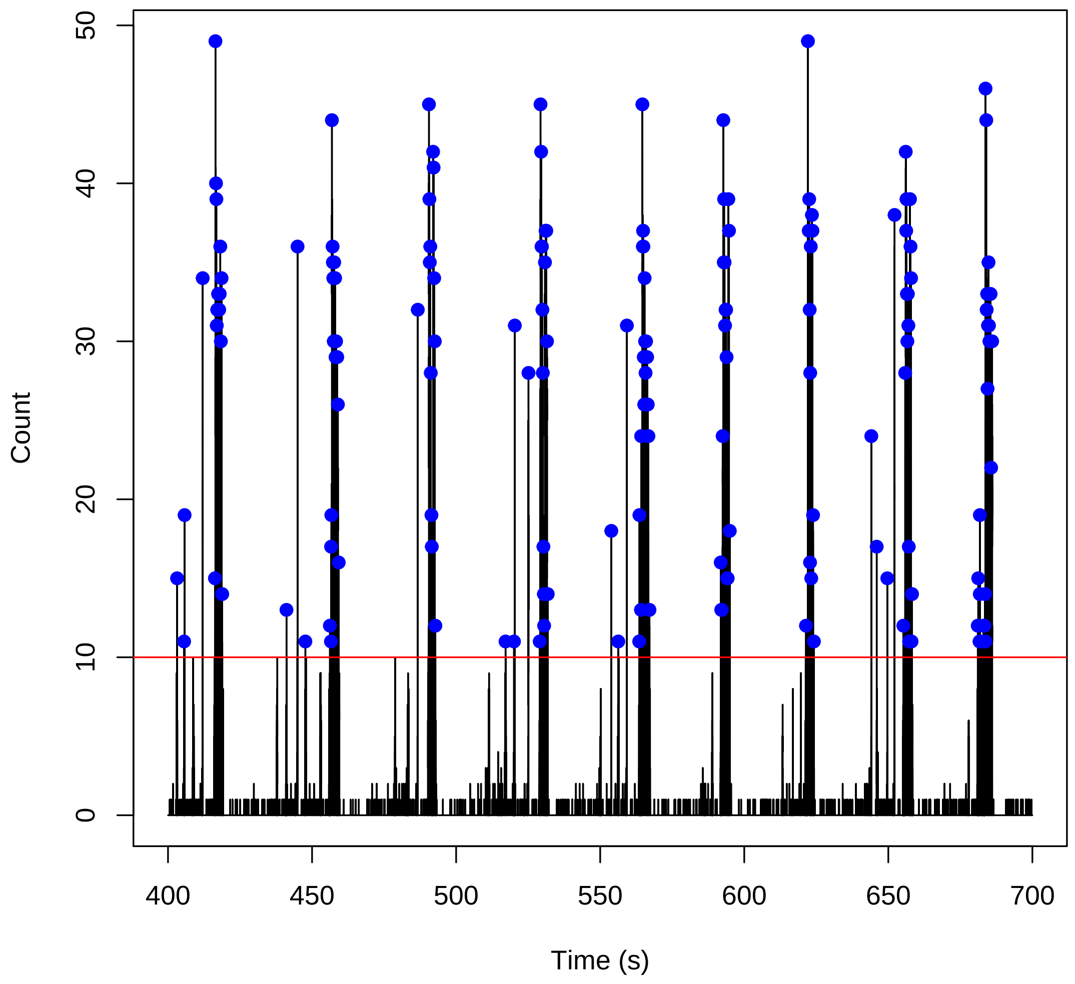
can be used within this package, but in principle (computation time aside as I expect an HMM to be slow) there should be no issue. There is also a generic "bursts" package: `http://cran.r-project.org/web/packages/bursts/bursts.pdf`.

Network spikes

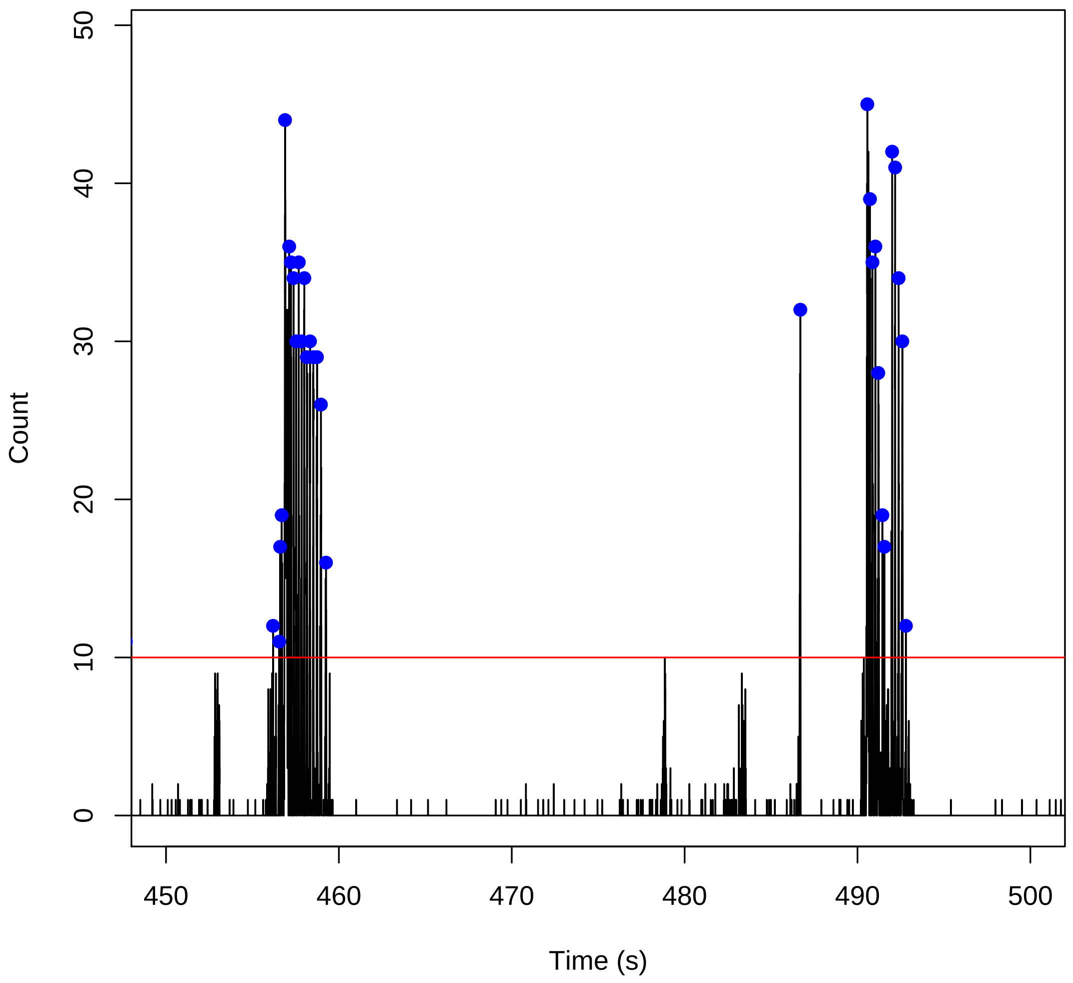
Network spikes are periodic elevations in activity across the whole array (**?**). The following example shows how they are computed. In the resulting graph, the population "firing rate" (the number of active electrodes here) is shown on the y axis, time (in seconds) on the x axis. The horizontal red line is a threshold set for the minimum number of active electrodes to determine a "network spike". The blue dots are the peak of each network spikes.

The mean network spike is also shown, averaged across all the network spikes in the recording.

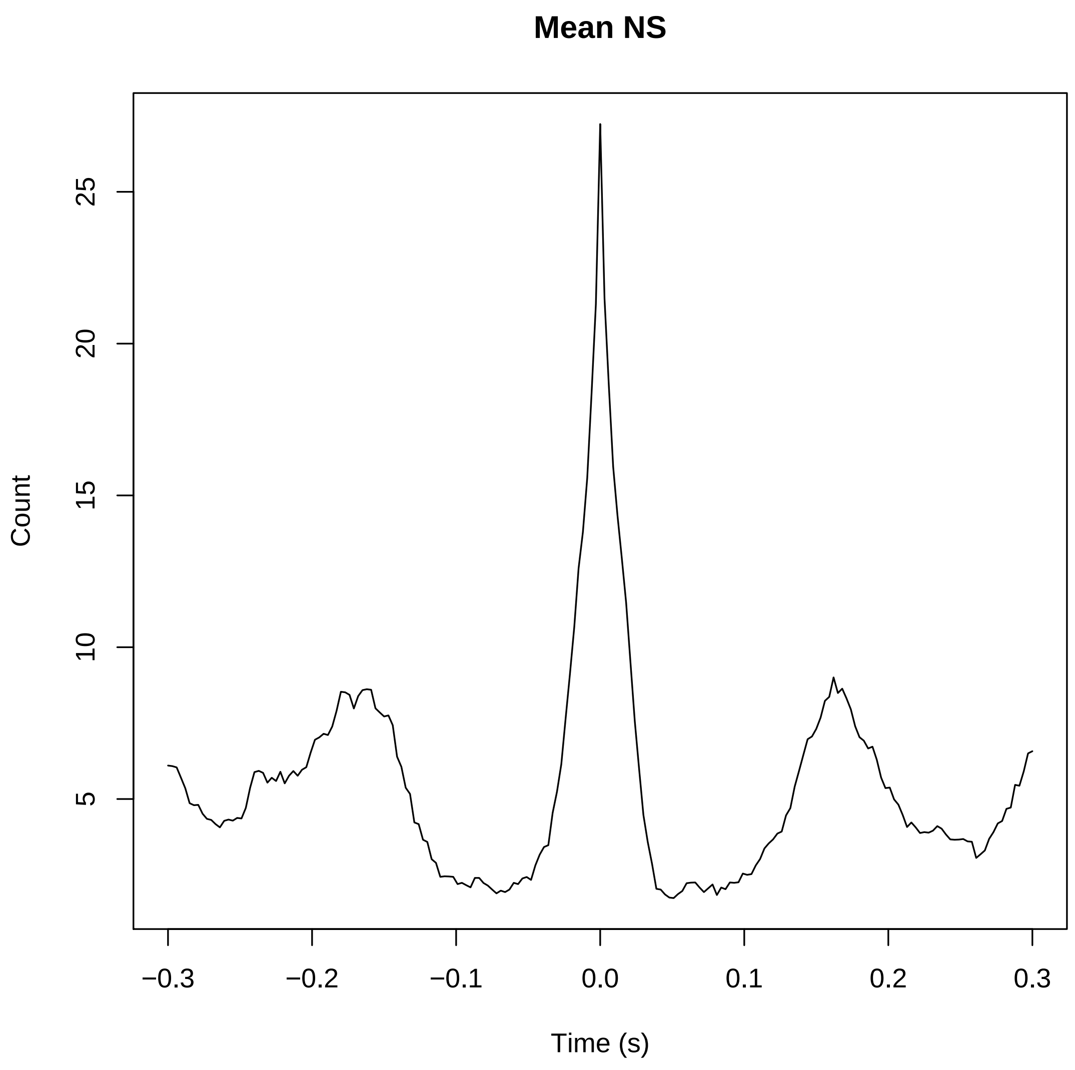
```
example(compute.ns, package = "sjemea")

##
## cmpt.n> data.file <- system.file("examples", "TC89_DIV15_A.nexTimestamps",
## cmpt.n+                           package = "sjemea")
##
## cmpt.n> s <- sanger.read.spikes( data.file, beg=400, end=700)
##
## cmpt.n> s$ns <- compute.ns(s, ns.T=0.003, ns.N=10,sur=100)
##
## cmpt.n> plot(s$ns, ylab='Count', xlab='Time (s)')
```



```
##
## cmpt.n> plot(s$ns, xlim=c(450, 500),
## cmpt.n+      xlab='Time (s)', ylab='Count')
```



```
##
## cmpt.n> plot(s$ns$mean, xlab='Time (s)', ylab='Count', main='Mean NS')
```



```
##
## cmpt.n> summary(s$ns)
## 167 network spikes
## recruitment 27.23 +/- 10.52
## FWHM 0.023 +/- 0.013 (s)
##
## cmpt.n> s$ns$brief
##         n    peak.m   peak.sd    durn.m   durn.sd
## 167.00000  27.23353  10.51559   0.02315   0.01264
##
## cmpt.n> ## show.ns(s$ns)  # This shows each network spike!  Can take a long time.
## cmpt.n>
## cmpt.n>
## cmpt.n>
```

References

Compiling this document

```r
require(knitr)
knit2pdf("meadq-into.Rnw")
```