An introduction to the sjemea package

S. J. Eglen

March 17, 2014

Installation

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

```
install.packages(c("sjemea"), type = "source", contriburl = "http://damtp.cam.ac.uk/user/eglen/r/")
vignette("sjemea-intro", package = "sjemea")
```

Setup

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

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

Introduction

This is a short introduction to the abilities of the sjemea 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

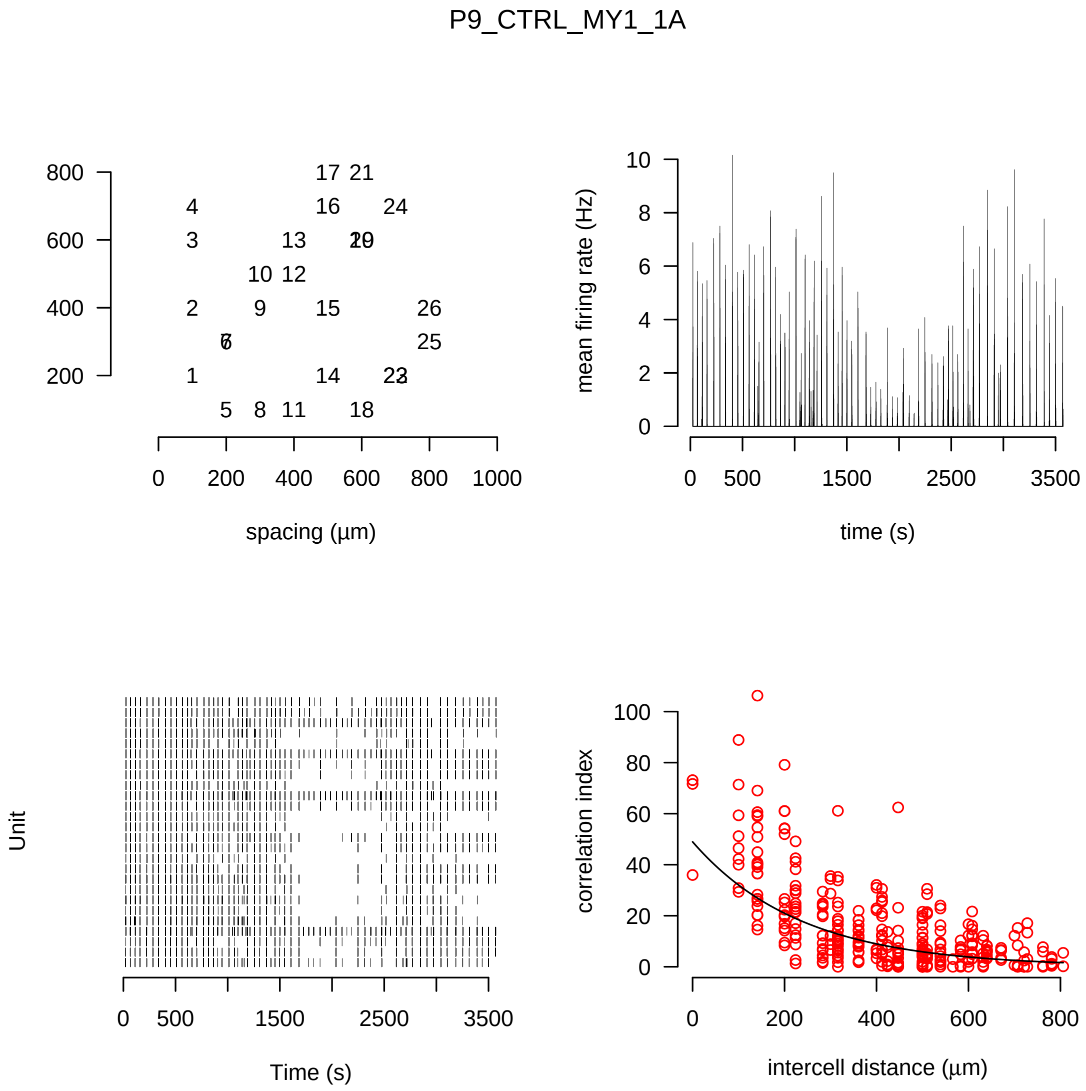
A list of help pages associated with the package is given by `help(package='sjemea')` command:

```
##
##  Information on package 'sjemea'
##
## Description:
##
## Package:            sjemea
## Description:        Description: Multi-electrode analysis (St
##                     Louis, Sanger, Ncl, IIT, Axion)
## Title:              Multi electrode analysis
## Version:           0.41
## Date:              2014-02-10
## Depends:           R (>= 1.4.0)
## Suggests:          R.matlab, RSQLite, rhdf5, testthat, knitr
## Author:            Stephen Eglen <sje30@cam.ac.uk>
## Maintainer:        Stephen Eglen <sje30@cam.ac.uk>
## License:           GPL (>= 2)
## URL:               http://www.damtp.cam.ac.uk/user/sje30/r
## VignetteBuilder:   knitr
## Built:             R 3.0.3; x86_64-pc-linux-gnu; 2014-03-17
##                    14:26:21 UTC; unix
##
## Index:
##
## compute.ns              Compute network spikes
## feller.read.spikes      Read in a directory of spike times from Marla
##                         Feller and create a "spikes" data structure.
## hist.ab                 Histogram routines to help compute the
##                         cross-correlation between a pair of spike
##                         trains.
## jay.read.spikes         Read in the .txt file from Neuroexplorer and
##                         create a "spikes" data structure.
## litke.read.spikes       Read in the .mat file containing MEA data from
##                         Alan Litke.
## litke1layout            Layout of the MEA provided by Alan Litke, as
##                         used in the 2009 Neuron paper.
## sanger.read.spikes      Read in a nexTimestamps file from Neuroexplorer
##                         and create a "spikes" data structure.
## spikes.to.bursts        Burst detection of MEA spike trains.
```

Fourplot

The fourplot gives a quick overview for a data file, showing (a) the positions of recorded neurons (b) the array-wide firing rate, estimated by default every 1 s (c) the spike raster for the entire recording (d) the correlation index plot (Wong et al., 1993).

```
data.file <- system.file("examples", "P9_CTRL_MY1_1A.txt", package = "sjemea")
s <- jay.read.spikes(data.file)
fourplot(s)

## Warning:  removing 17 zero entries
```



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).

Burst analysis

We have several routines implemented for burst analysis:

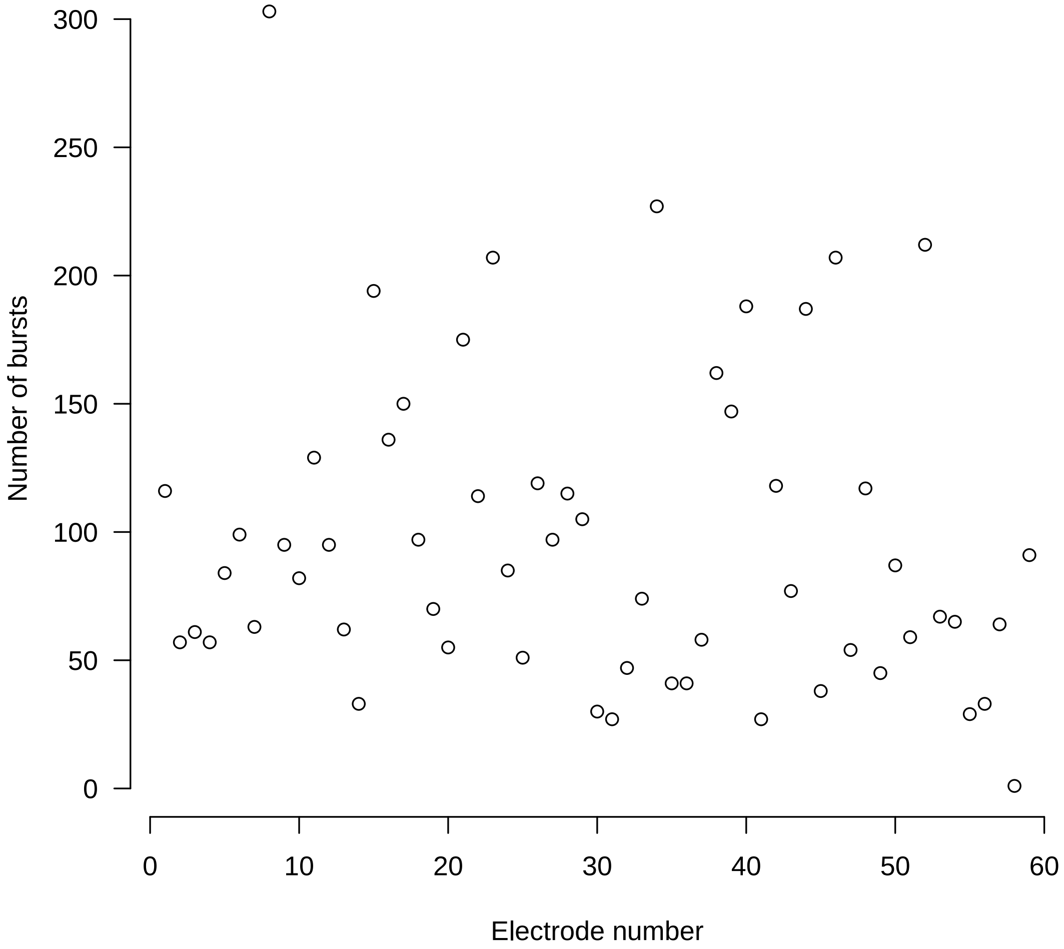
1. Max Interval method, as described by Neuroexplorer (NexTechnologies, 2012)
2. Poisson surprise (Legéndy and Salcman, 1985)
3. Rank suprise (Gourévitch and Eggermont, 2007)

Out of these, the most tested has been the MaxInterval method.

```
data.file <- system.file("examples", "TC89_DIV15_A.nexTimestamps", package = "sjemea")
s <- sanger.read.spikes(data.file)
s$allb <- spikes.to.bursts.surprise(s)
```

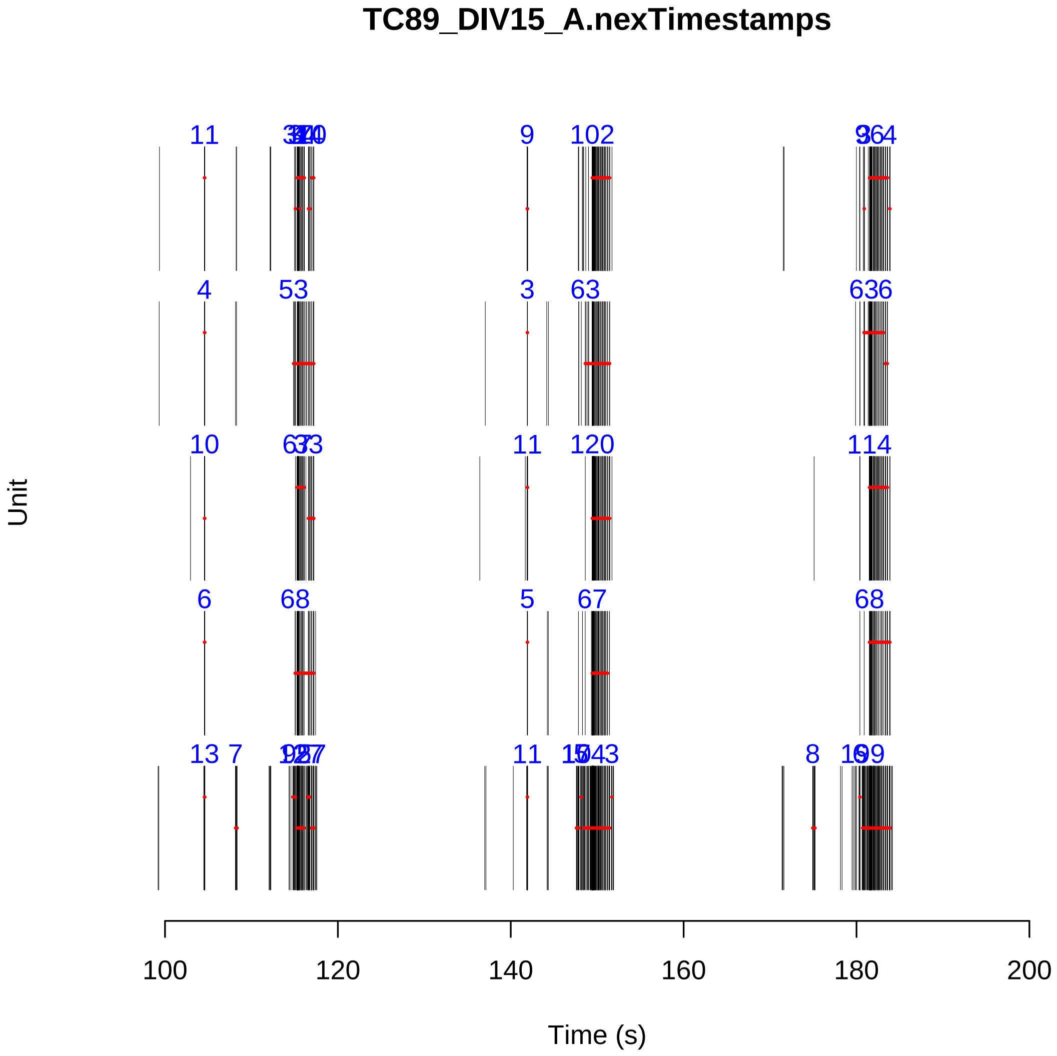
So, for example, for electrode 2, we see the following bursts (just taking the head as there are many of them. We can also easily plot the number of bursts on each electrode.

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

##      beg len     SI    durn mean.isis
## [1,]   1  19  70.76 0.06848  0.003804
## [2,]  21  60 103.03 1.95984  0.033218
## [3,]  85  14  44.38 0.08668  0.006668
## [4,] 100  56  93.56 1.92436  0.034988
## [5,] 158   6  15.44 0.05552  0.011104
## [6,] 164  68 118.08 2.17580  0.032475

nbursts <- sapply(s$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, beg = 100, end = 200, 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 (Tokdar et al., 2010) 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`.

Log interspike intervals

To help determine burst parameters, it is often helpful to look at a histogram of the interspike interval, plotted on a log scale.

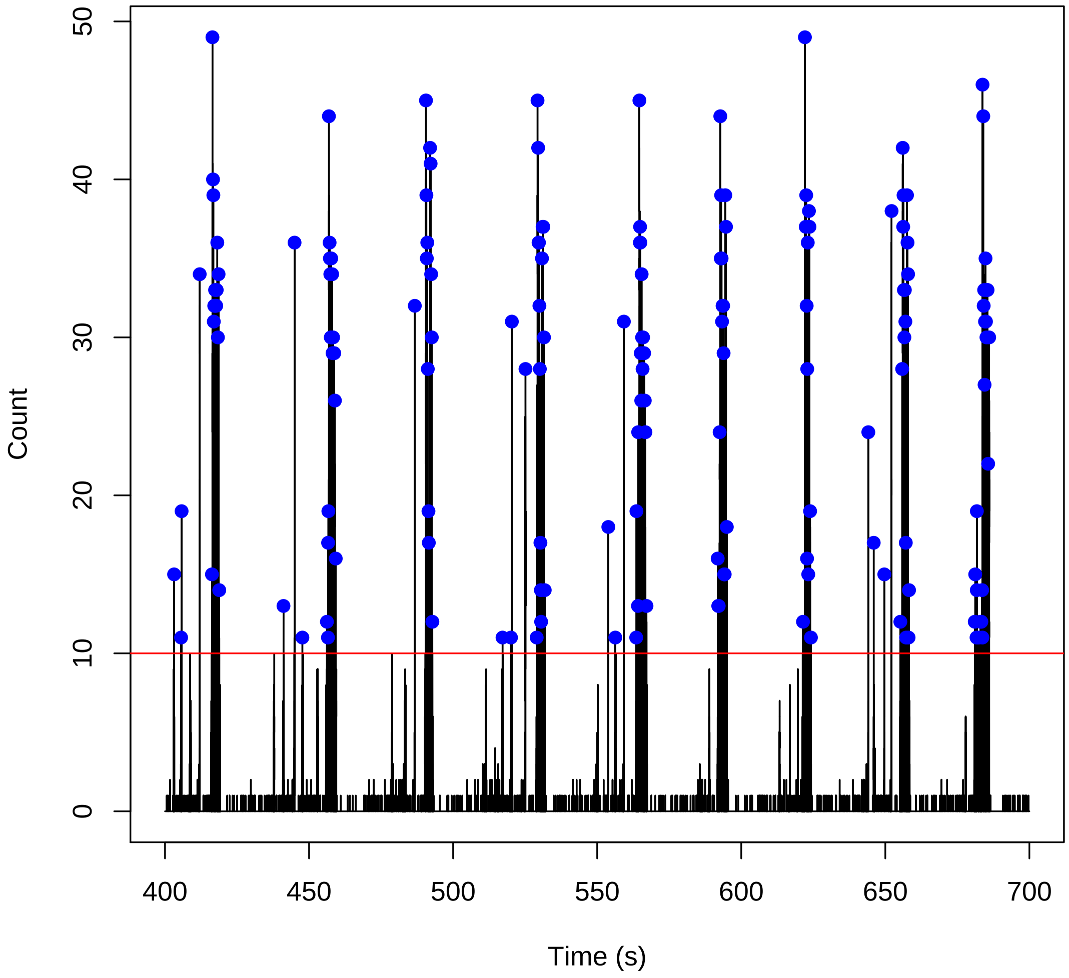
TODO: add this code from `~/proj/sangermea/pdn/logisi_condtable.R`.

Network spikes

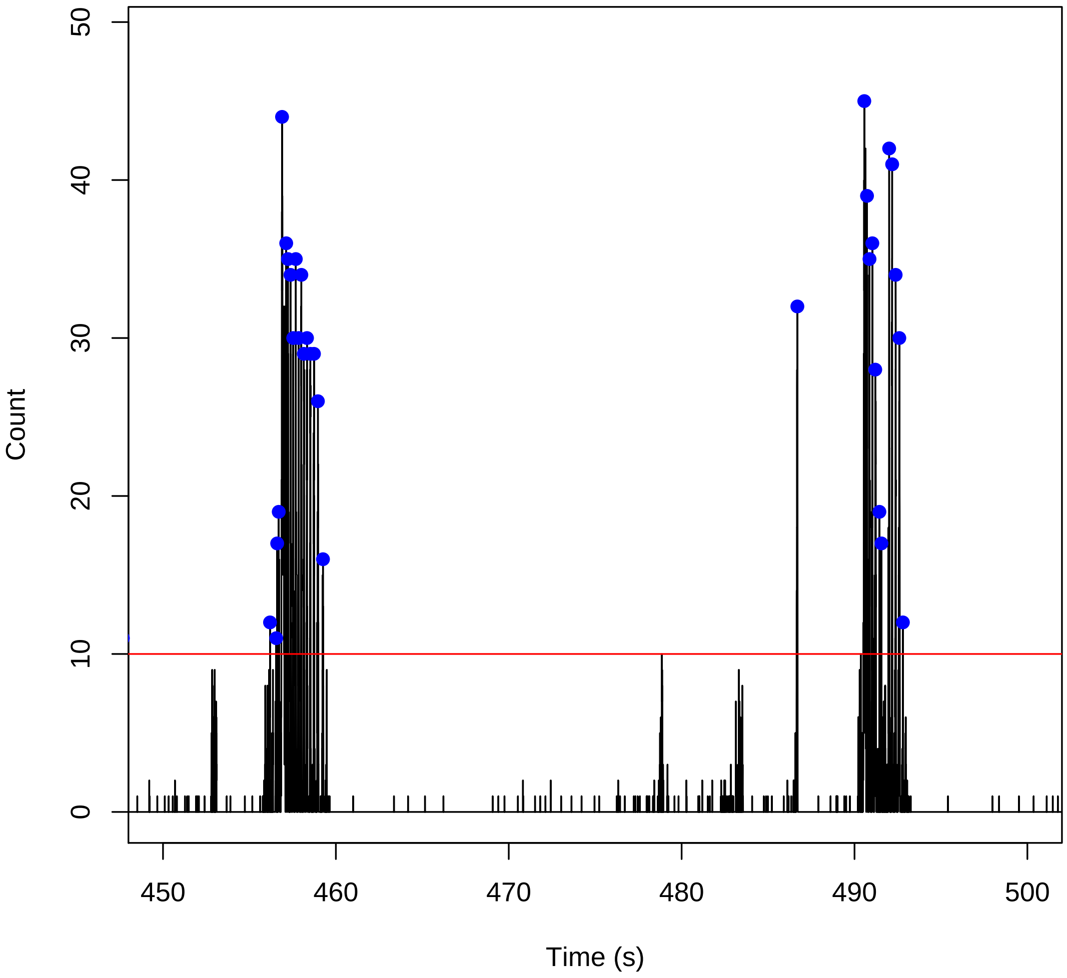
Network spikes are periodic elevations in activity across the whole array (Eytan and Marom, 2006). The following example shows how they are computed. In the resulting graph, the population "firing rate" (the number of active electrodes here) is shown on the y axis, time (in seconds) on the x axis. The 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.

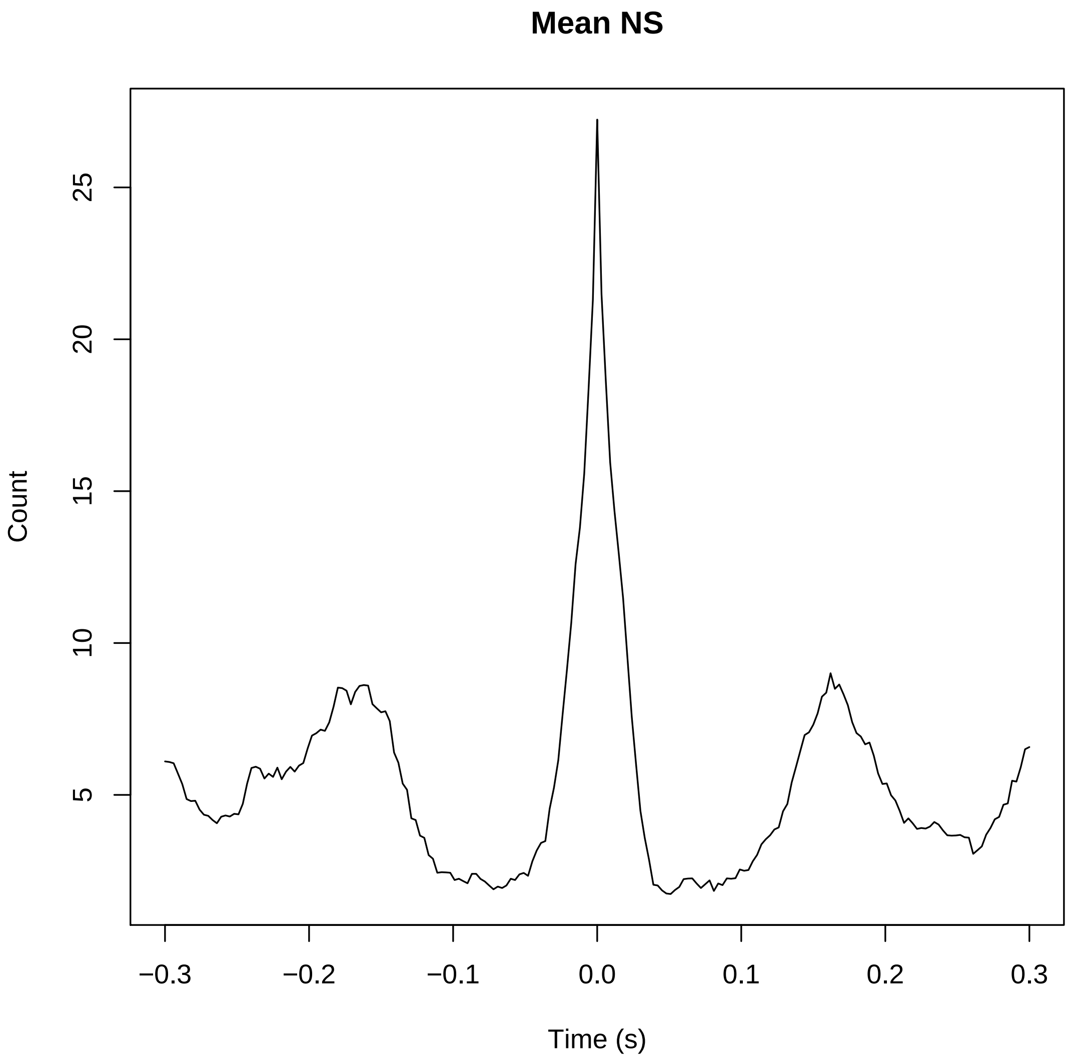
```r
example(compute.ns)

##
## 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)')
```



```r
##
## 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')
```

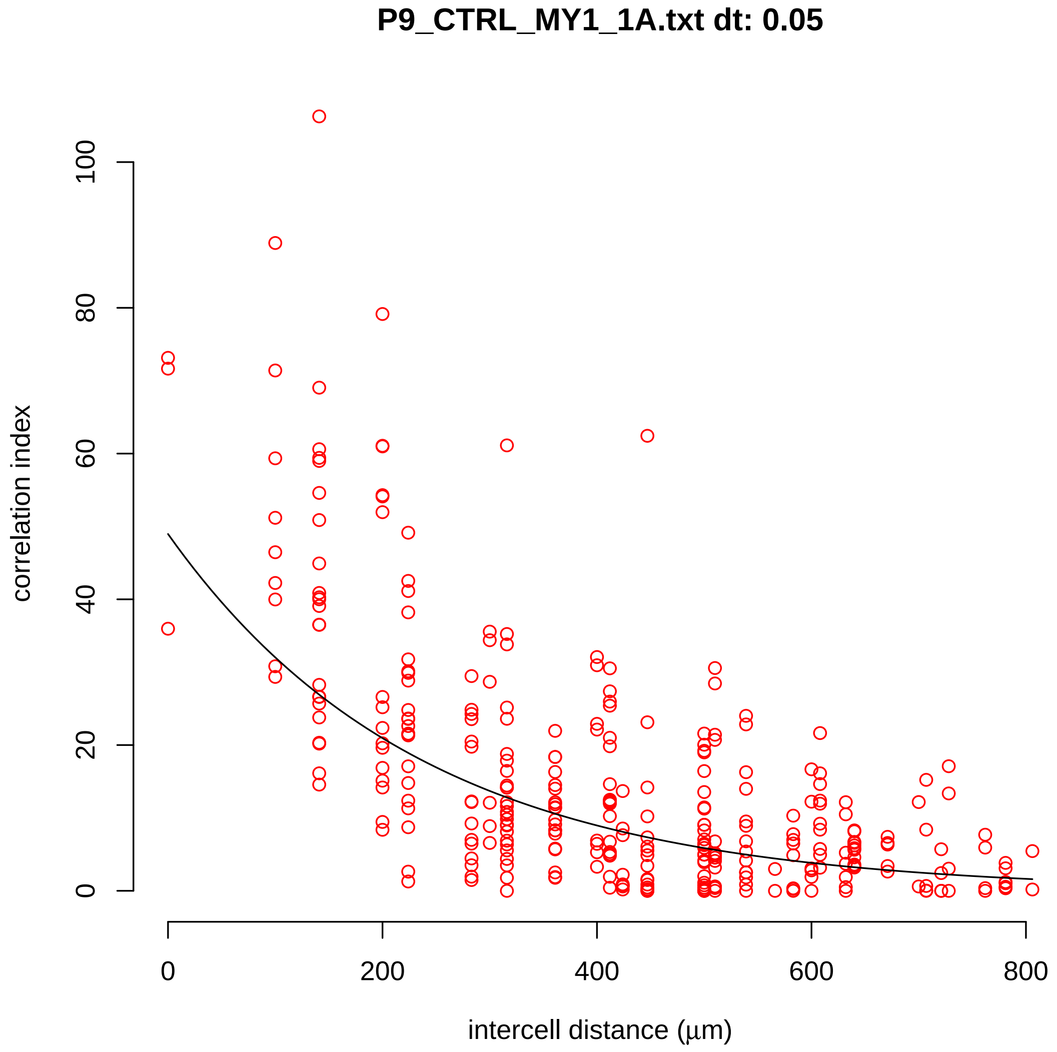


```
##
## cmpt.n> summary(s$ns)
## 167 network spikes
## recruitment 27.23 +/- 10.52
## FWHM 0.023 +/- 0.013 (s)
##
## cmpt.n> s$ns$brief
##         n    peak.m   peak.sd    durn.m   durn.sd
## 167.00000  27.23353  10.51559   0.02315   0.01264
##
## cmpt.n> ## show.ns(s$ns)  # This shows each network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

Correlation index

The correlation index plot was devised by Wong et al. (1993) as a method to estimate how correlation between any pair of neurons on the array depends (if at all) upon the distance separating the pair. For retinal waves, the correlation index usually has an exponentially-decaying profile. For other recordings, (e.g. hippocampal cultures), the profile tends to be flatter.

```
jay.data.file <- system.file("examples", "P9_CTRL_MY1_1A.txt", package = "sjemea")
jay.s <- jay.read.spikes(jay.data.file)
plot.corr.index(jay.s)

## Warning:  removing 17 zero entries
```



```
##
## Call:
## lm(formula = y.log ~ x)
##
## Coefficients:
## (Intercept)            x
##     3.89108     -0.00424
```

Batch analysis

As this code is all written in the R progamming language, it scales itself well to the notion that the analysis can be automated to run over many data files in batch, rather than running one at a time. To this end, for the Genes to Cognition project, we devised a system where data files were expected in one directory, and output files would be written to a particular directory. See the R function `sanger.init` and in particular the variables `mea.data.dir, mea.table.dir, mea.op.dir`. A spreadsheet containing the files to be analysed is passed to a script that analyses each row of the spreadsheet independently. This system was used to analyse several hundreds of files simultaneously.

Data readers

The examples above have used example data stored within the package to read in data. Here are the names of some (not all) other readsers that are available in the package.

ncl.read.spikes – Multi-Channel system MCS output of spike times and cut outs. (Named "ncl", as the data came from Newcastle.)

jay.read.spikes – Used for Demas et al. (2003) (Named "jay" as the data was recorded by Jay Demas.)

h5.read.spikes – HDF5 format, as described in the current "waverepo" project.

sql.read.spikes – Prototype using SQL to store data.

sanger.read.spikes – Multi-channel system from the Sanger Genes to Cognition project.

nex.read.spikes – Neuroexplorer timestamp files can be read in. I have adapted the matlab code from `http://www.neuroexplorer.com/code.html` to work in R.

Handling new data

Although these old readers still work, where possible, we should use the HDF5 format for data storage. The preferred approach (as taken with the "waverepo" project), is to convert data as soon as possible into HDF5.

Some older routines exist for reading in binary format (e.g. `mm.read.spikes`) but these are quite old now and cumbersome to maintain. Whenever possible, data collected from laboratories should either be text or HDF5.

Multi-well arrays

Through a collaboration with Dr Tim Shafer's group at EPA, we are developing code for handling multi-well data. (For example, so that each well can be analysed independently in an efficient manner.) This introduces an extra layer of abstraction into the geometry of the array, in that an electrode is part of a well, and there are many wells on one array (or "plate").

Other features

Interactive facilities for viewing spike times were in earlier versions of the code, but Tk() widgets conflicted with the use of parallel() code. I think this has now been fixed, so we might be able to return to adding interactive features.

History

This package was originally written for research that was published in (Demas et al., 2003). Since then it has been used by several other research groups.

Acknowledgements

This work has been financially supported by grants from the Wellcome Trust, BBSRC and EPSRC. Zhengzheng Zhang provided the `logisi` function. I thank Paul Charlesworth for many discussions about electrophysiological analysis.

References

Demas J, Eglen SJ, Wong ROL (2003) Developmental loss of synchronous spontaneous activity in the mouse retina is independent of visual experience. *J. Neurosci.* 23:2851–2860.

Eytan D, Marom S (2006) Dynamics and effective topology underlying synchronization in networks of cortical neurons. *J. Neurosci.* 26:8465–8476.

Gourévitch B, Eggermont JJ (2007) A nonparametric approach for detection of bursts in spike trains. *J. Neurosci. Methods* 160:349–358.

Legéndy CR, Salcman M (1985) Bursts and recurrences of bursts in the spike trains of spontaneously active striate cortex neurons. *J. Neurophysiol.* 53:926–939.

NexTechnologies (2012) *NeuroExplorer Manual*.

Tokdar S, Xi P, Kelly RC, Kass RE (2010) Detection of bursts in extracellular spike trains using hidden semi-markov point process models. *J. Comput. Neurosci.* 29:203–212.

Wong ROL, Meister M, Shatz CJ (1993) Transient period of correlated bursting activity during development of the mammalian retina. *Neuron* 11:923–938.

Compiling this document

```r
require(knitr)
knit2pdf("sjemea-intro.Rnw")
```