meadq package: from A to Z

Diana Hall

June 10, 2014

Introduction

This is an introduction to the abilities of the meadq package for analysis of multielectrode array data. The program is specifically suited to analyze data from the 12 & 48 well MEA plates, such as those from Axion Biosystems. 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.

Installation

To install this package, a binary source code may be used or github may be used for latest version. Package sources via the URL `http://github.com/dianaransomhall/meadq`.and then view this introductory vignette

Installation using source

```
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")
```

Installation using github

Alternatively, most recent version meadq may be downloaded from github. "devtools" R-package will need to be installed to enable the install_github functionality.

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

1 R software reference

For those new to R, the free software may be downloaded from `http://www.cran.r-project.org`. A highly user-friendly interface may be downloaded here `http://www.rstudio.com` with a guide here `http://www.dss.princeton.edu/training/RStudio101.pdf`. A beginners guide to R is available here `http://www.cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf` and inevitable trouble shooting is best done by googling, as web documentation on R abounds.

Vignette

To view the introductory vingette:

```
vignette("meadq-intro", package = "meadq")
```

Setup

Install and load packages into current R session.

```
install_github("sje30/sjemea")  # companion pkg to meadq
install_github("yihui/knitr")  #to compile vignette
source("http://bioconductor.org/biocLite.R")  #for HDF5 files
biocLite("rhdf5")
# load packages into current session
require(rhdf5)
```

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

```
require(meadq)
require(knitr)
opts_chunk$set(cache = TRUE)  #to save time if results previously generated
opts_chunk$set(dev = "pdf")
```

Help pages

A list of help pages associated with the package is given by:

```
help(package = "meadq")
```

File naming conventions and experiment log file

File names help store meta-data about file as well as provide a unique identifying used in the link to the experiment log file. The experiment log file serves as more in depth documentation of what was done in experment. Together the log file and the file name provide redundancy in experimental protocol useful as a check and also make data sets readily usable for those unfamiliar with data.

File naming conventions

Aspects of the file naming conventions must be followed in order that the code executes without error. Those aspects which much be followed are that the first 4 chunks must be seperated by "_" and must give meta-data that matches with the experimental log file .csv file. This is necessary for meadq to match the input file to experimental meta-data in the log file and combine them in the resulting hdf5 file.
Let's use the example file provdied within the package subdirectory"extdata":

spike train text file ON_20140205_MW1007-26_DIV07_001.mapTimestamps

log file Experiment_LogFileExample.csv

Project "ON" codes the project or group of experiments: "ON" for ontogeny.

position in file name immediately preceeding the first '_'

format The project desination may be of any length or capitalization but needs to begin with a letter.

designation in log file The corresponding column in the log file must be entitled "Project". The contents of the "Project" column in the log file and the project designation in file name must be identical.

Experiment Date "20140205" denotes the date of the plating of the neurons, February 5, 2014.

position in file name immediate following the first '_'

format YYYYMMDD

designation in log file The corresponding column in the log file must be entitled "Experiment Date". The contents of the "Experiment Date" column in log file must be identical to the Experiment Date designation in file name.

Plate SN "MW1007-26" is the serial number (SN) of the plate used in the recording.

position in file name The serial number always follows the second '_'.

format any format alpha or numeric, so long as serial number contains no _

designation in log file The corresponding column in the log file must be entitled "Plate SN". The contents of column entitled "Plate SN" in the log file must be identical to the plate serial number designation in the file name.

DIV "DIV07" denotes the 7th day in vitro (DIV) of the plated cells.

position in file name After the 3rd '_'.

format "DIV07", "DIV7", "07", "7" are all permissible so long as a exact match is made to log file.

designation in log file The DIV column in the log file must be entitled "DIV". The contents of the "DIV" column in the log file and the DIV designation in the file name must be identical.

Users Choice Any other naming conventions may be added to end of file name. Example data has '001' to indicate sequential order of recording should more than one be made with same meta-data. Other examples may include indication of 'pre' or 'post' should there be pre and pose dose recordings.

Experiment log file conventions

Experiment log file must be a .csv (comma seperated value) file that contain the names and information listed below. See "Experiment_LogFileExample.csv" located in the "extdata" subdirectory of package for an example.

Column names in log file must match verbatim to list below (case sensitive):

Project

format Must match file name. See "Project" in above subsection.

Experiment Date

format Must match file name. See “Experiment Date” in above subsection.

Plate SN

format Must match file name. See “Plate SN” in above subsection.

DIV

format Must correspond to file name. See “DIV” in above subsection.

Well

format one row in log file for each well; must be a capital letter preceeded by a number e.g. "A1"or "F3"

Treatment

format any character string without spaces is permissible.

Size

format give any size data about anything or list NA in column if no size specification exist

Dose

format dose of treatment. Any rational, or integer number is permissible

Units

format any-alpha numeric unit specification is permissible

Creating HDF5 Files

Converting data into HDF5 file format is beneficial for many reasons. Firstly, converting the recorded data from a proprietary or otherwise restrictive file type into a universal file type, such as HDF5, facilitates data sharing and provides flexibility in analysis software choice. Secondly, the “program on standards for datasharing” by the International Neuroinformatics Coordinating Facility `http://www.datasharing.incf.ord/ep/HDF5_data_standard` has recommended the HDF5 format. Finally, HDF5 files are designed to store meta-data and data seperately for maximum efficiency. For more information of the HDF5 format see `http://www.hdfgroup.org/HDF5/doc/H5.into.html`. Code is provided to convert axion alpha map files (.map extension) to text files and finally to HDF5 (.h5 extention) format files and in the process store meta-data inside recording file. The intermediate step of converting alpha map files to text files is necessary since alpha map files are in binary and may only be decoded by a third party software such as NeuroExplorer (**?**).

1.1 create text files from alpha map files

NeuroExplorer has a scripting language that may be used to batch create text files (.mapTimestamps extension) of spike trains from apha map files. meadq needs text spike trains to use the ".mapTimestamps" file extension. For information regarding script language see manual here `http://www.neuroexplorer.com/downloads/Nex3Manual.pdf`. A neuroExplorer script entitled "Export_timestamps_from_map_files_meadqExample.nsc" is provided in the "extdata" subdirectory of meadq folder.

Execution

interactive session The script may be executed by initiating an interactive session of NeuroExplorer, opening the script and clicking the green play button. A dialogue box will appear in the NeuroExplorer main page prompting for user to enter directory containing .map files to be converted. Enter the file path where the .map files are located (make sure to keep the ".*map") and press ok. The script outputs the .mapTimestamp files in the same folder as the .map files. Try running the script on the example .map file provided in the "extdata" folder entitled "ON_20140205_MW1007-26_DIV05_001.map "

"Export_timestamps_from_map_files_meadqExample.nsc"

```
doc = 0
filefilter ="C:/Users/dhall05/Desktop/*.map"
res = Dialog(doc, filefilter, "File Filter:", "string" )
% create save folder
position1 = Find(filefilter, "*" )
save_folder = Left( filefilter, position1-1)

Trace(save_folder, " timestamps have been saved in a text file")
n= GetFileCount(filefilter)

Trace(n, "files")
for i=1 to n
name = GetFileName(i)
doc = OpenDocument (name)
doc_title=GetDocTitle(doc)
length_doc_title = StrLength(doc_title)-4
doc_title=Mid(doc_title,1,length_doc_title )
save_path= save_folder + doc_title + ".mapTimestamps"

stampslabel = GetDocPath(doc) + "Timestamps"
if doc > 0
SaveAsTextFile(doc, save_path)
Trace(name, "timestamps have been saved in a text file")
CloseDocument(doc)
end
end
```

1.2 create HDF5 files from text files

```
##
data.file <- system.file("extdata", "ON_20140205_MW1007-26_DIV07_001.mapTimestamps",
    package = "meadq")
# user will be promted to navigate to .mapTimestamps files & log file
make.axion.map.to.h5.dh()
```

Create well summary spreadsheet

Create two csv files :

ont_data_summary_AEfilt.csv computes well summaries for each file using all AE (active electrodes: ≥ 5 spikes/min)

ont_data_summary_ABEfilt.csv computes well summaries for each file using all ABE (actively bursting electrodes: ≥ 1 burst/min)

```
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)
param.file <<- system.file("data", "chgv_parameters.rda", package = "meadq")

create_ont_csv(h5Files = h5Files, save.rdata = TRUE, param.file = param.file)
```

In the process of making the well summary, a rdata (R's native data type) has been created in the same directory where the .h5 files are located, as save.rdata argument was set to TRUE. Rdata or .rda extention is easily loadable by R using the following commands:

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

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.

Burst analysis

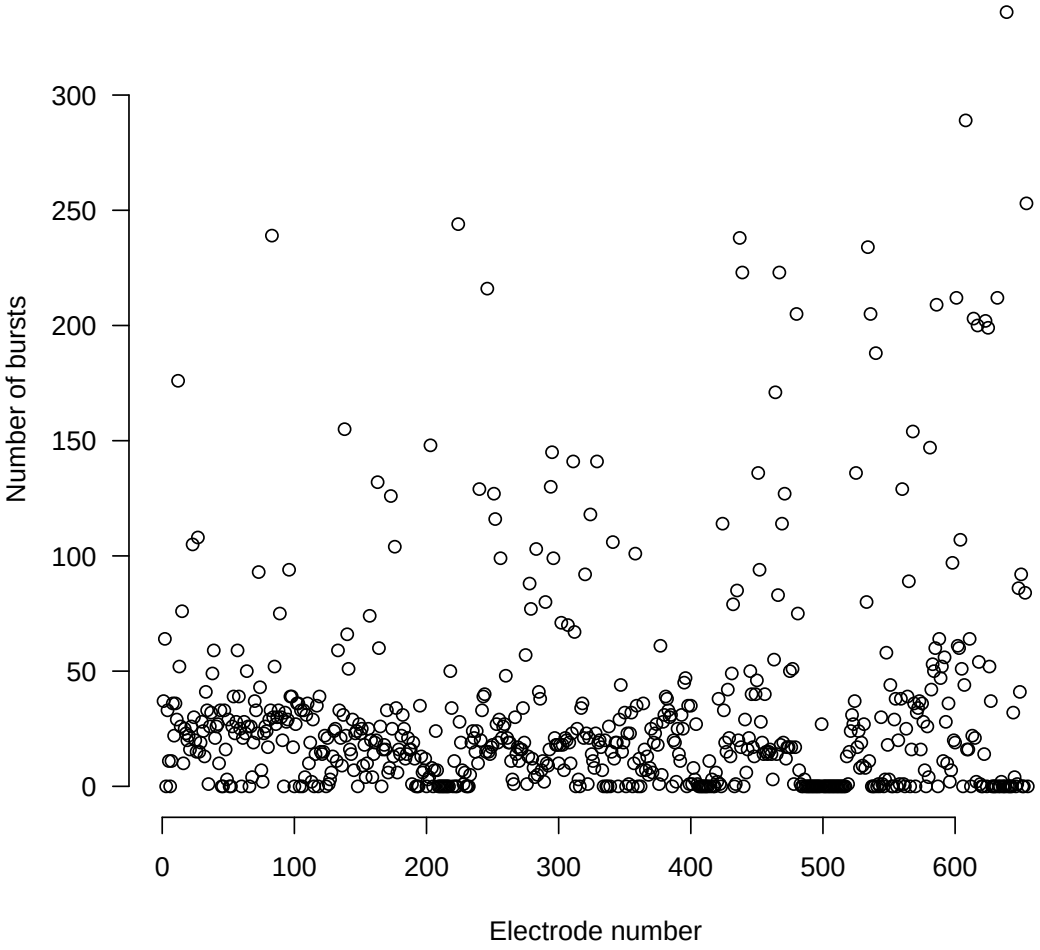
There are several routines for burst analysis. The most common is implemented:

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

```
data("example_ont_data")
```

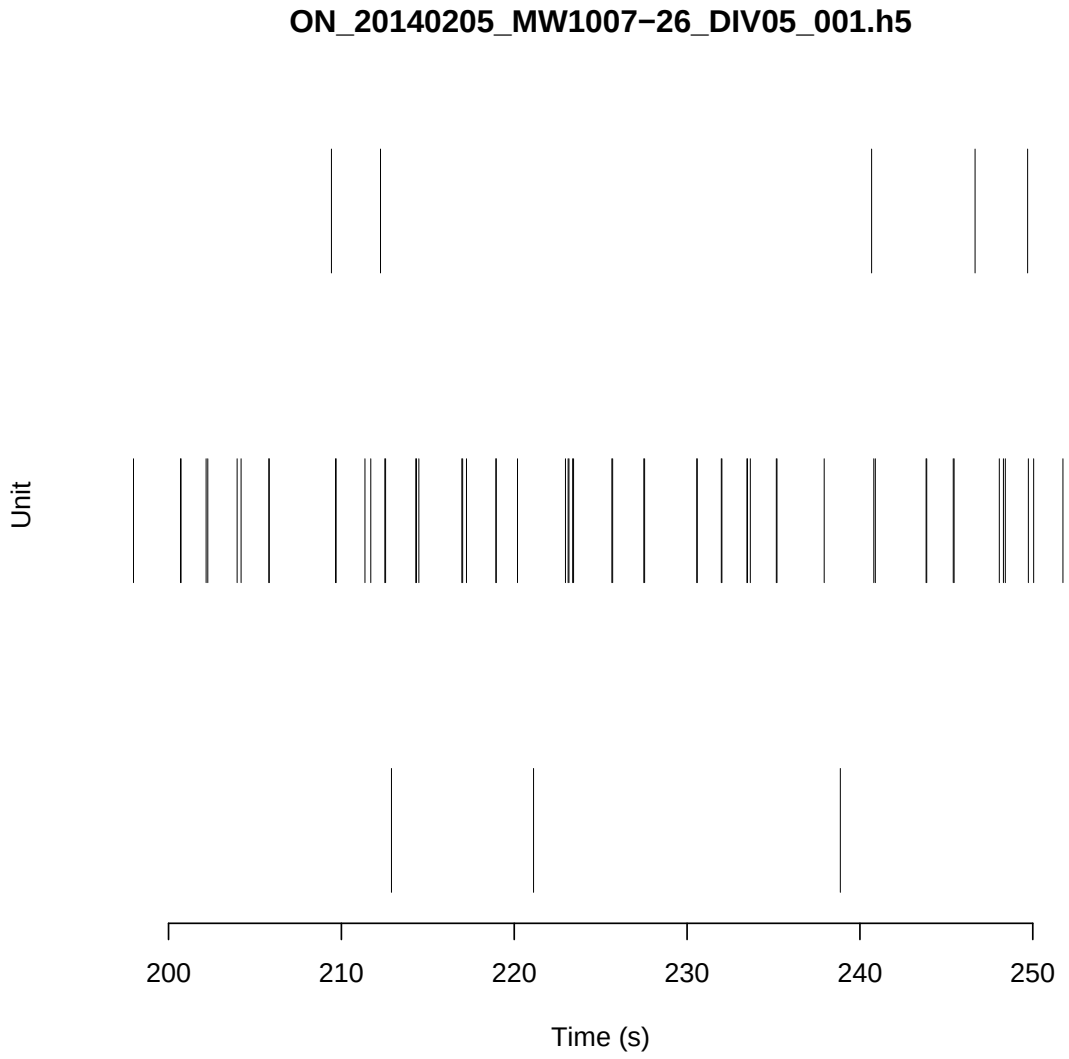
So, for example, for electrode 2, we see the following bursts (just taking the head as there are many of them. We can also easily plot the number of bursts on each electrode.

```
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.

```
plot(s[[2]], 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`.

Plate Summary Graphics

Get summary graphics of one of multiple files.

```
h5Files = system.file("data", "example_ont_data.rda", package = "meadq")
param.file = system.file("data", "chgv_parameters.rda", package = "meadq")

burst_table_Plots(param.file = param.file, h5Files = h5Files)
```

PCA: Principle components analysis

PCA is a useful technique in the MEA data framework. Many variables or "features" may be derived from a spike train without knowing which among these best capture the inforation in data. PCA takes high-dimensional data, so called for the many variables such as mean firing rate, inter burst interval, etc. comprising the data, and identifies combinations of those variables, or PC dimensions, that are responsible for the greatest percentage of variation. In this manner, a few PC dimensions describe a large percentage of variation in the data thereby reducing the number of variables, or dimensionality, of the data.

```
filename.data = system.file("extdata", "ont_data_summary_AEfilt.csv", package = "meadq")
trt.params.wanted = list(DIV.wanted = c("7", "9", "12"), trt.wanted = c("Acetaminophen"),
    dose.wanted = c(1, 3, 10))
ctr.params.wanted = list(DIV.wanted = c("7", "9", "12"), trt.wanted = c("Acetaminophen"),
    dose.wanted = c(0))
output.folder = dirname(filename.data)
vars.wanted = c("dose", "meanfiringrate", "burst.per.min", "mean.isis", "per.spikes.in.burst",
    "mean.dur", "mean.IBIs", "nAE", "nABE", "ns.n", "ns.peak.m", "ns.durn.m")

PCA.by.well(filename.data = filename.data, trt.params.wanted = trt.params.wanted,
    output.folder = output.folder, ctr.params.wanted = ctr.params.wanted, vars.wanted = vars.wanted)
```

Network spikes: sjemea pacakge

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.

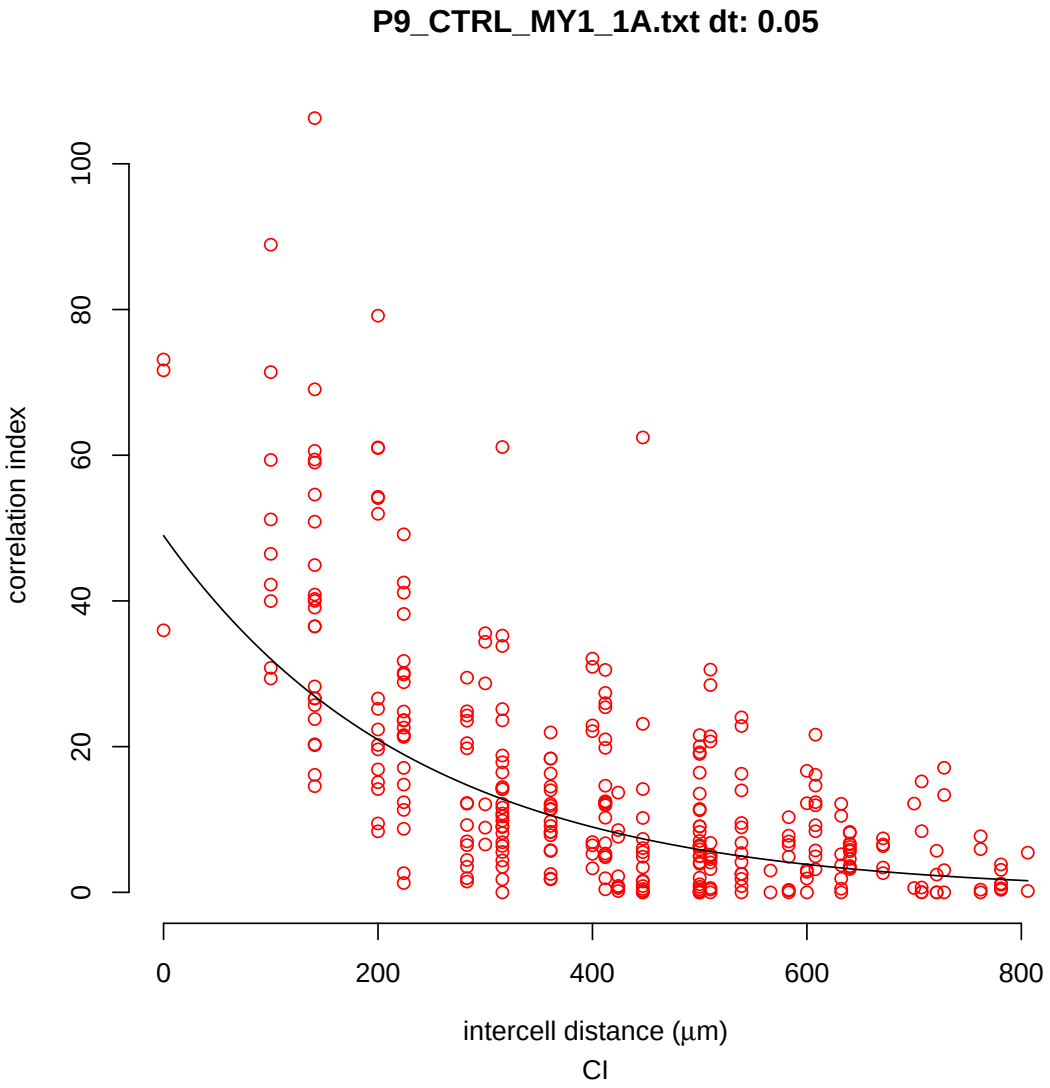
The mean network spike is also shown, averaged across all the network spikes in the recording.

```
example(compute.ns)
```

Correlation index: sjemea package

The correlation index plot was devised by **?** 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.

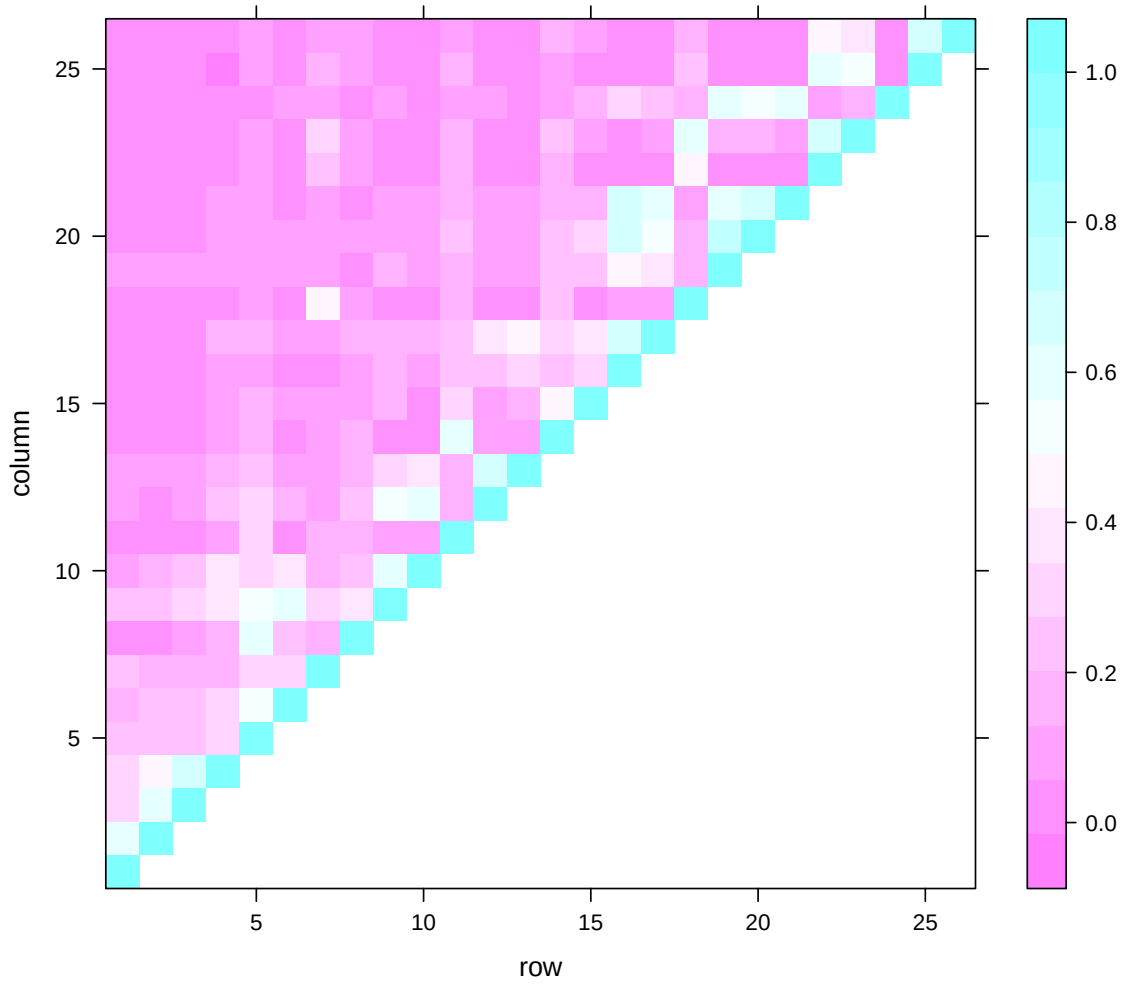
```
jay.data.file <- system.file("examples", "P9_CTRL_MY1_1A.txt", package = "sjemea")
jay.s <- jay.read.spikes(jay.data.file)
plot.corr.index(jay.s)

## Warning:  removing 17 zero entries
```



Correlation analysis: sjemea package

We propose a new tiling-based measure for measuring the correlation between pairs of spike trains (ongoing work by Catherine Cutts). Here is an example of how to compute a tiling correlation matrix for a group of spike trains.

```
data.file <- system.file("examples", "P9_CTRL_MY1_1A.txt", package = "sjemea")
s <- jay.read.spikes(data.file)
t2 <- tiling.allpairwise(s)
require(lattice)
levelplot(t2)
```



Acknowledgements

Thanks to Stephen Eglen at DAMTP, Cambridge UK for his help in creating this package by building off of his package "sjemea", available `http://github.com/sje30/sjemea`. Thanks to Tim Shafer, US EPA for his help and data.

References

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

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

```
require(knitr)
knit2pdf("meadq-intro.Rnw")
```