BEFORE VENTURING INTO THE SUBJECT OF SAMPLE DEPTH AND CHRONOLOGY QUALITY, WE STATE FROM THE BEGINNING “MORE IS ALWAYS BETTER”. HOWEVER, AS MENTIONED EARLIER ON THE SUBJECT OF BIOLOGICAL-GROWTH POPULATIONS THIS DOES NOT MEAN THAT ONE COULD NOT IMPROVE A CHRONOLOGY BY REDUCING THE NUMBER OF SERIES USED IF THE PURPOSE OF REMOVING SAMPLES IS TO ENHANCE A DESIRED SIGNAL. THE ABILITY TO PICK AND CHOOSE WHICH SAMPLE TO USE IS AN ADVANTAGE UNIQUE TO DENDROCLIMATOLOGY. THAT SAID IT BEGS THE QUESTION, HOW LOW CAN WE GO?
ESPER ET AL (2003), *TREE-RING RESEARCH* **59**, 81–98.

THE COMBINATION OF SOME DATA AND AN ACHING DESIRE FOR AN ANSWER DOES NOT ENSURE THAT A REASONABLE ANSWER CAN BE EXTRACTED FROM A GIVEN BODY OF DATA.
JOHN TUKEY

AN ARTICLE ABOUT COMPUTATIONAL SCIENCE IN A SCIENTIFIC PUBLICATION IS NOT THE SCHOLARSHIP ITSELF, IT IS MERELY ADVERTISING OF THE SCHOLARSHIP. THE ACTUAL SCHOLARSHIP IS THE COMPLETE SOFTWARE DEVELOPMENT ENVIRONMENT AND THE COMPLETE SET OF INSTRUCTIONS WHICH GENERATED THE FIGURES.
JON CLAERBOUT

CHRISTOPHE POUZAT

SPIKE SORTING: A RE-PRODUCIBLE EXAMPLE USING R

First printing, May 2010

Contents

List of Figures

List of Tables

Introduction

If you are brand new to R read what follows. R is free, open source and of course *really great*. You can get it from the R project site[1].

You will also need another free and open source software: the GGobi Data Visualization System[2].

These two software can seem a bit hard to use at first sight. R does not follow the nowadays common "point and click" paradigm. That means that a bit of patience and a careful reading of the tutorials are *de rigueur*. R documentation is plentiful and goes from the very basic to the most advanced stuff. The "contributed documentation" page[3] is good place to start. Look in particular at: "R for Beginners" by Emmanuel Paradis (french and spanish versions are also available) and "An Introduction to R: Software for Statistical Modelling & Computing" by Petra Kuhnert and Bill Venables [4]. Another good place is Ross Ihaka's course: "Information Visualisation", the course uses R to generate actual examples of data visualization[5] and is a wonderful introduction to its subject. Ross Ihaka, together with Robert Gentleman, is moreover one of the original R developers. The "two-day short course in R"[6] of Thomas Lumley is also great. There is an R Wiki site[7] which is worth looking at.

Windows users can enjoy the SciViews R GUI developed by Philippe Grosjean & Eric Lecoutre[8] and are strongly encouraged to use the Tinn-R[9] editor to edit R codes, etc. Information on how to configure Tinn-R and R can be found in the lecture notes of Kuhnert and Venables. On Linux I'm using the emacs editor together with ESS : Emacs Speaks Statistics [10].

[1] http://www.r-project.org

[2] http://www.ggobi.org/

[3] http://cran.r-project.org/other-docs.html

[4] Petra Kuhnert and Bill Venables. An Introduction to R: Software for Statistical Modelling & Computing. Technical report, CSIRO Mathematical and Information Sciences, 2005. URL http://www.csiro.au/resources/Rcoursenotes.html

[5] http://www.stat.auckland.ac.nz/~ihaka/120/

[6] You can find it at the following address: http://faculty.washington.edu/tlumley/b514/R-fundamentals.pdf.

[7] http://www.sciviews.org/_rgui/wiki/doku.php

[8] http://www.sciviews.org/SciViews-R/index.html

[9] http://www.sciviews.org/Tinn-R/index.html

[10] http://ess.r-project.org/

Spike Sorting

At the beginning of this document both R commands and there results will be presented. We will then turn to a presentation of the results only as we would do to document our own analysis. But the full set of commands necessary to reproduce it exactly is available in the associated *vignette* [11], the ".Rnw" file.

We now give a complete example of reproducible research in a spike sorting context. This example make use of a set of R functions called SpikeOMatic to be released sometime, hopefully soon, as a proper R package. The present set of functions together with a tutorial can be downloaded from the SpikeOMatic web page[12].

[11] Friedrich Leisch. Sweave and Beyond: Computations on Text Documents. In Kurt Hornik, Friedrich Leisch, and Achim Zeileis, editors, *Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria*, 2003. URL `http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/`. ISSN 1609-395X

[12] `http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/newSOM/newSOMtutorial/newSOMtutorial.html`.

Starting up

After R has been started by clicking on the proper icon or by typing something like:

```
$ R
```

at the terminal (I'm assuming here that the terminal prompt is the symbol $). Since some of the functions we are going to use make use of random initialization it is important, if we want to keep our analysis strictly reproducible, to seed our (pseudo-)random number generator (rng) explicitly. When several choices of rngs are available it cannot hurt to be explicit in our choice. We therefore start by using set.seed function:

```
> set.seed(20061001,"Mersenne-Twister")
```

When we deal with long analysis (or with long simulations) storing partial results can save a lot of time while our work progresses. This done by using package cacheSweave or pgfSweave[13] and by defining a sub-directory into which partial results are going to be stored or cached. This is done with function setCacheDir:

[13] The latter calls the former.

```
> setCacheDir("cache")
```

Here our partial results are going to be stored in a sub-directory called "cache".

We keep going by loading the set of functions making up SpikeOMatic directly from the web site where they are made available:

```
> ## Define a common address part
> baseURL <- "http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat"
> ## Define last part of address to code
> codeURL <- "Code_folder/SpikeOMatic.R"
```

```
> ## source code in workspace
> source(paste(baseURL,"/",codeURL,sep=""))
```

Loading data

The data are freely available from the web[14]. They are made of a 20 s long tetrode recording from a locust, *Schistocerca americana*, antennal lobe. The data were collected in the absence of stimulation, filtered between 300 and 5000 Hz and sampled at 15 kHz[15]. They are stored as "native binary" with 64 bits per measurement (which is an overkill since the UIE DAQ board used had a 16 bits precision). The data are available in compressed format so we first download the 4 files corresponding to the 4 recording sites of the tetrode from the server to our working directory on our hard drive:

[14] `http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Data.html`.

[15] Negative potentials appear as upward deviations on these data (a remain of an intracellular physiologist's past).

```
> ## Define last part of address to data
> dataURL <- "Data_folder/"
> ## define dataNames
> dataNames <- paste("Locust_",1:4,".dat.gz",sep="")
```

```
> ## download files
> sapply(dataNames, function(dn)
         download.file(paste(baseURL,"/",dataURL,dn, sep=""),
                       destfile=dn,quiet=TRUE,mode="wb")
         )
```

```
> ## Check that the files are now in working directory
> list.files(pattern=".dat.gz")
[1] "Locust_1.dat.gz" "Locust_2.dat.gz" "Locust_3.dat.gz"
[4] "Locust_4.dat.gz"
```

We now load the data into R's workspace:

```
> locust <- sapply(dataNames,
                   function(dn) {
                     myCon <- gzfile(dn,open="rb")
                     x <- readBin(myCon,what="double",n=20*15000)
                     close(myCon)
                     x
                   }
                   )
```

The first 400 ms of data are shown next.

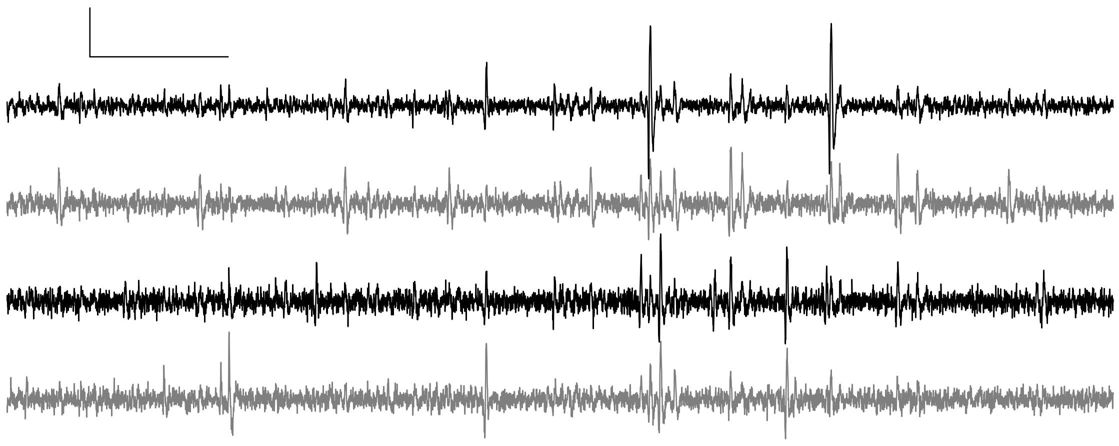

Figure 1: The first 400 ms of data on site 1, 2, 3 and 4 (from top to bottom) of one tetrode. Horizontal scale bar: 50 ms; vertical scale bar: 500 μV.

Preprocessing

We might want to explore the effect of filtering. A low-pass filter could be applied to eliminate part of the high frequency noise. Smoothing with cubic spline functions with a smoothing parameter set by minimizing the `generalized cross validation` criterion[16] could also be used.

[16] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996

For the data at hand neither filtering nor smoothing improves the sorting quality. Smoothing does in fact make it slightly worst due to an apparent extra peak position jitter resulting from the smoothing procedure. We will therefore focus next on the analysis carried out on the raw data. It is nevertheless straightforward to perform the same analysis on the filtered or on the smoothed data by changing one variable name in the `.Rnw` file.

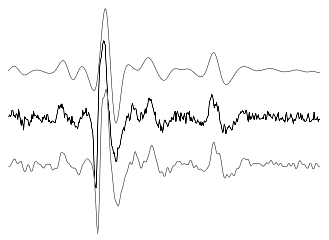

Figure 2: 25 ms from the first recording site (middle), smoothed with a cubic spline (top) and filter with a 3 kHz Bessel filter (bottom). The region shows the first "big" spike of figure 1.

Spike detection

Following the precise description given in the `SpikeOMatic` (new) tutorial we quickly come to the detection step. The detection threshold on each recording site is set as a multiple of the *medial absolute deviation* (MAD) on the corresponding site. Although `SpikeOMatic` contains utility functions like `summary` to help its user setting these thresholds this step is not well automatized requiring decisions to be made by the user. This is one step where the reproducible analysis approach is particularly useful since non-automatic actions get fully documented. For the present data, we decided to settle on a threshold of 4 times the MAD on each recording site. Only local maxima (peaks) were detected. The detection was moreover performed on a box-filtered version of the data[17]. With such a threshold we end up with 1723 events.

[17] Each measurement is replaced by the average of itself and its two nearest neighbors.

Cutting events

Once spikes are detected, events have to be "cut", that is, a small window is taken from the raw data, on each recording site, around the event's occurrence time. Then two parameters have to be chosen: the window length and the events reference time position within the window. For these data we chose windows of 3 ms placing the reference time (peak) at 1 ms. These two parameters are again chosen by the user, a choice fully documented in the vignette. A user / reader having access to this vignette (the `.Rnw` file) can easily change these parameters at will and see the effect on the final analysis.

From this point and for "simple" data like this locust antennal lobe recording, we get our classification of events in a two steps process. When events are defined by 3 ms long windows and when several neurons are active in the recording some events will in fact be the result of the near simultaneous occurrence of 2 or more action potentials. We would like to identify and classify events due to the superposition of two spikes as such. We will therefore build a "clean" sample by removing from our original sample the most obvious superpositions as shown on the figure bellow. This is done by building an event specific envelope with *a single* maximum and *at most two* minima. `SpikeOMatic` includes a function allowing users to set interactively the parameters of this envelope (see the tutorial) or to check, interactively again, how good the envelope parameters provided in a vignette are. With this data set, using the envelope parameters given in the vignette we start with a 1679 events[18] large sample and end up with 1472 clean events.

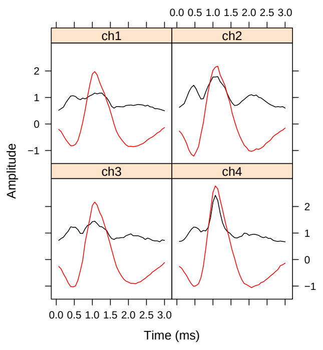

Figure 3: The median event (red) and its MAD (black). Sample size: 1723. Ordinate unit: 100 μV.

[18] When events get "cut", if two spikes are within a single window a single one is kept, the largest. This number is therefore smaller than the number of detected spikes.

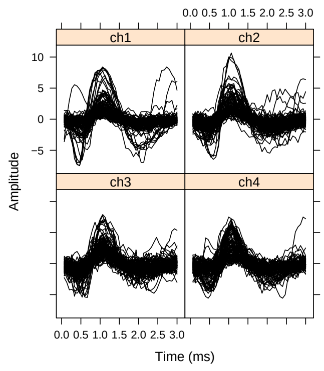
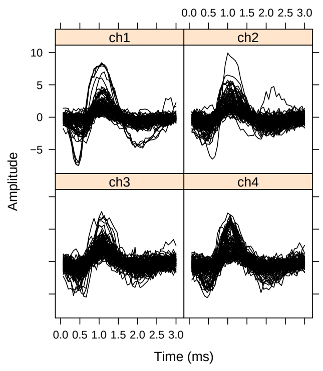
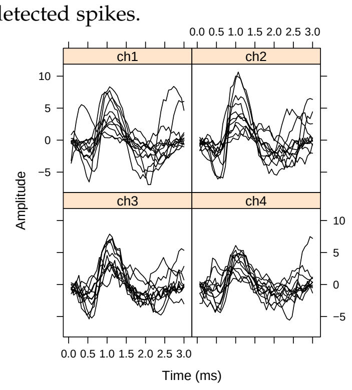

Figure 4: Left, the first 100 events; middle, the 89 "clean" events; right, the 11 "unclean" events.

Dimension reduction

We are presently using 3 ms long windows on 4 recording sites to represent our events. Since our sampling rate was 15 kHz our sample space has $4 \times 15 \times 10^3 \times 3 \times 10^{-3} = 180$ dimensions. Such a large number can become a serious problem if we try to estimate both location (mean value) and scale (covariance matrix) parameters of probability densities. Since this is precisely what some of our model-based clustering algorithms are going to do, *we have to* reduce the dimension of our sample space before going further. We use systematically *principal component analysis* (PCA) to do that. Clearly other methods like *independent components analysis* (ICA) could be used. Since they are implemented in `R` a sceptical reader having access to our vignette could very easily try out such an alternative method and see if any different classification would follow. In our experience PCA does a perfectly good job but we have to admit that we are not ICA experts.

Fine but even if you agree that PCA is satisfying there is still a decision to make: How many dimensions (principal components) should we keep. We don't believe that an automatic criterion like: keep enough components to account for 80% of the total variance, is suitable. The number of components to keep depends too strongly on the signal to noise ratio of the individual neurons spikes and on the number of neurons present in the data. In other words it is too sample dependent[19]. So we keep k components if the projection of the clean sample on the plane defined by components k and $k+1$ is "featureless", that is, looks like a bivariate Gaussian[20]. We point out at this stage that interactive multidimensional visualization software like `GGobi` can be extremly useful in this decision process. We see (bellow) that a "structure" is still apparent in the projection of the clean sample onto the plane defined by principal components 7 and 8 (left), there is a slight decrease of points density around -2.5 on the abscissa (also visible on the density estimate shown on the right). No such structure seems present in the projection onto the plane defined by principal components 8 and 9.

[19] But for a given preparation, *e.g.* the locust antennal lobe, we always end up with nearly the same number of components.

[20] See the discussion in Ripley (1996) p. 296.

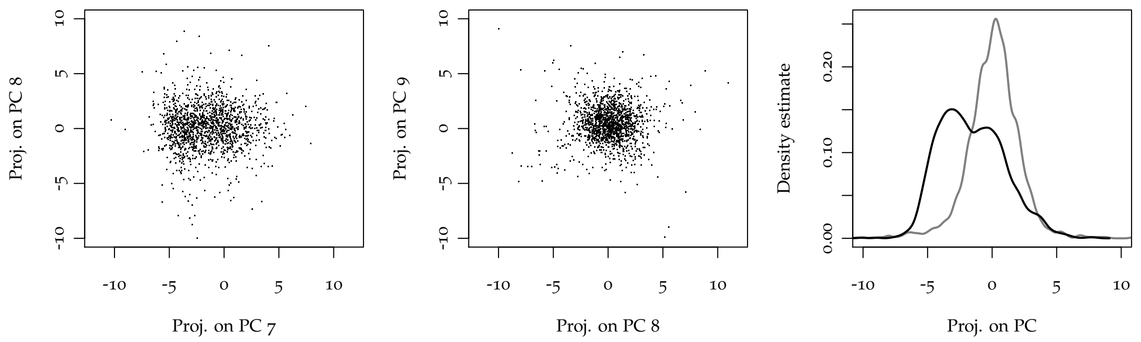

Figure 5: Left, clean sample projected on the plane defined by PC 7 and 8; middle, clean sample projected on the plane defined by PC 8 and 9; right, estimated densities of projections on PC 7 (black, smoothing bandwidth: 0.39) and on PC 8 (grey, smoothing bandwidth: 0.25). A Gaussian kernel was used for densities estimation. The bandwidth were chosen to slightly undersmooth the data.

Based on these observations we would decide to keep the first 7 principal components which happen to account for 75% of the total sample variance. The matrix of scatter plots whose elements are sample projection on the planes defined by every possible pair of principal components already shows nice clusters. We see for instance on figure 6 10 clusters on the projection onto the plane defined by principal components 2 and 3 (second row, third column), meaning that we should include at least 10 clusters / neurons in our models. Here again using a dynamic visualization software like `GGobi` can be very useful, since more often than not some structures which are lost on sequences of static projections start appearing clearly.

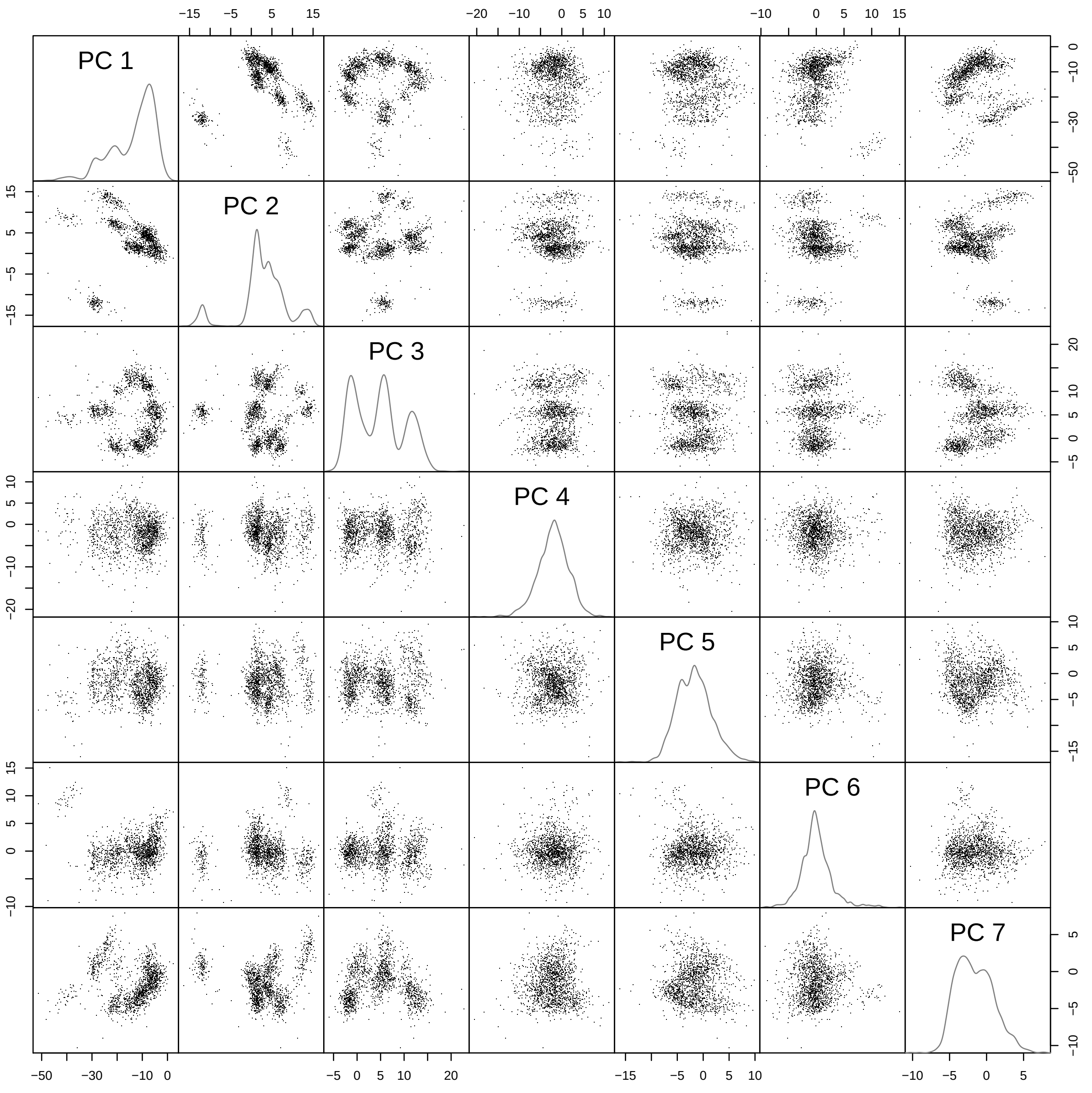

Figure 6: Scatter plot matrix of projections onto planes defined by pairs of the 7 principal components. Smooth density estimates of projections onto the corresponding principal component are shwon on the diagonal.

Clustering

Since our last graph shows 10 well separated clusters we can start with a very simple clustering method like `kmeans` and run it with 10 centers. Doing that we get the clusters shown on figure 7.

The median and medial absolute deviation of each of the 10 clusters are shown on figure 8. The medial absolute deviation curve

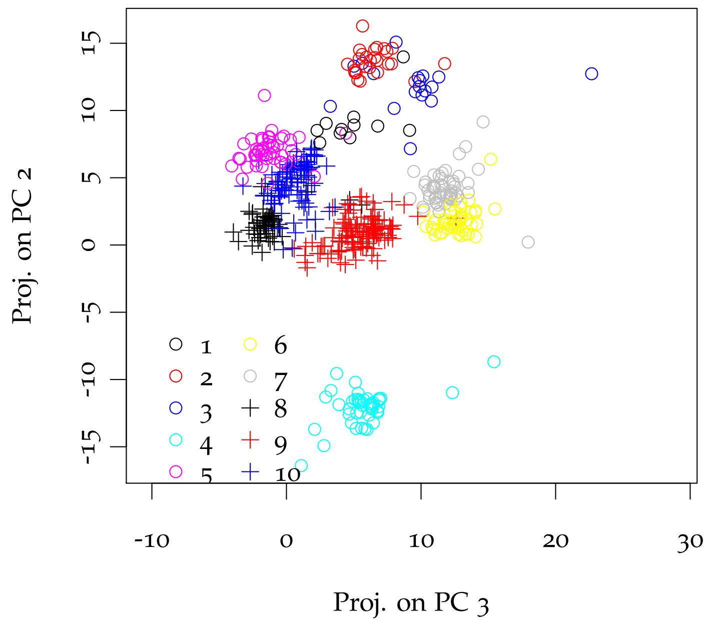

Figure 7: Results of kmeans clustering with 10 centers projected onto the plane defined by principle components 2 and 3. The first 500 clean events are shown.

are all nearly flat and a the same level on each recording site independently of the cluster. This is an indication of both good classification *and* of stability of the spike waveform of the individual neurons. In other words the intra-cluster variability is almost entirely due to the background noise. When a slight deviation from the horizontal appears on the MAD curves, like for the first and the fourth clusters, it occurs at the point of steepest slope of the mean waveform and is due to uncompensated sampling jitter[21]. The overall mean value of the medial absolute deviation is 65 μV. Inspection of the mean waveforms and comparison of their peak values with the MAD level suggests moreover that the last three clusters give rise to signals too small for reliable classification.

[21] C. Pouzat, O. Mazor, and G. Laurent. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. *J Neurosci Methods*, 122(1): 43–57, 2002. DOI: 10.1016/S0165-0270(02)00276-5. Pre-print available at: `http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Papers_folder/PouzatEtAl_2002.pdf`

A conservative approach would be to forget about these last three clusters in the subsequent analysis and consider that only 7 good clusters / neurons are present in the data set. Before going further one should of course look at the individual events on a cluster by cluster basis as shown on figure 9. This is easily done with `SpikeOMatic` and the required commands are present in the associated vignette. Using `GGobi` at this stage is also a good idea.

Model construction and complete sample classification proceed in a straightforward way as described in the tutorial and explicitly shown in the vignette. The superposition identification works moreover rather well on this "easy" data set as can be checked with the `plot` method for sorting results of `SpikeOMatic` (not shown but directly usable from the vignette).

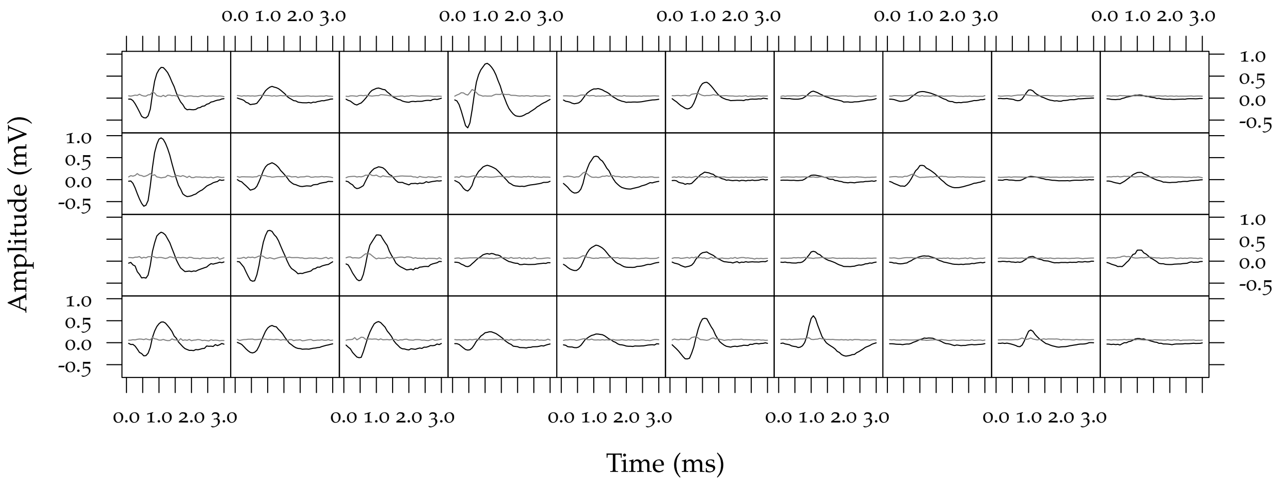

Figure 8: The median (black) and MAD (grey) of the 10 clusters obtained with kmeans. Cluster number increases from left to right and correspond to the legend of figure 7. Channel 1 is at the top, channel 4 at the bottom.

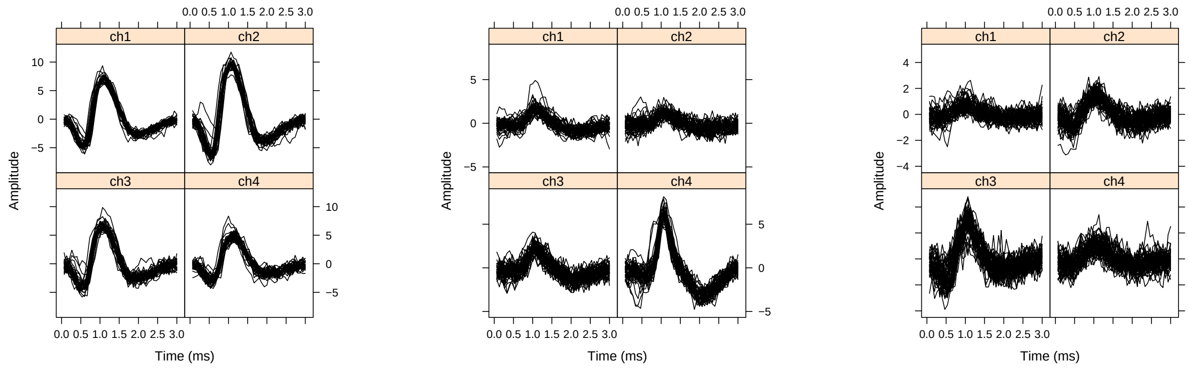

Figure 9: Left, the 36 events from cluster / neuron 1; middle, the first 50 of the 198 events from cluster / neuron 7; right, the first 50 of the 215 events from cluster / neuron 10.

Spike Train Analysis

Given the very short duration, 20 s, of the recording we cannot illustrate any serious spike train analysis endeavour. We will therefore stick to some very basic stuff.

	nb	freq	min	max	mean	sd
Neuron 1	90	4	22	2828	213	399
Neuron 2	89	4	5	1567	224	329
Neuron 3	77	4	8	1435	246	303
Neuron 4	169	8	15	1075	117	192
Neuron 5	132	7	10	1060	147	170
Neuron 6	167	8	3	1253	119	169
Neuron 7	222	11	0	690	90	123
Neuron 8	324	16	2	634	62	65
Neuron 9	442	22	1	255	45	44
Neuron 10	341	17	2	658	58	68

Table 1: Elementary spike train statistics of the 10 isolated units. nb, total number of spikes; freq, rounded frequency (Hz); min, shortest isi (ms); max, longest isi (ms); mean, mean isi (ms); sd, isis' SD (ms). Durations rounded to the nearest ms.

Bibliography

Petra Kuhnert and Bill Venables. An Introduction to R: Software for Statistical Modelling & Computing. Technical report, CSIRO Mathematical and Information Sciences, 2005. URL `http://www.csiro.au/resources/Rcoursenotes.html`.

Friedrich Leisch. Sweave and Beyond: Computations on Text Documents. In Kurt Hornik, Friedrich Leisch, and Achim Zeileis, editors, *Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria*, 2003. URL `http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/`. ISSN 1609-395X.

C. Pouzat, O. Mazor, and G. Laurent. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. *J Neurosci Methods*, 122(1):43–57, 2002. DOI: 10.1016/S0165-0270(02)00276-5. Pre-print available at: `http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Papers_folder/PouzatEtAl_2002.pdf`.

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

Software and Packages Versions

In order to get real reproducibility when performing an analysis requiring several user contributed packages, one must keep track of the `R` and user contributed packages version used.

- R version 2.11.0 (2010-04-22), `x86_64-unknown-linux-gnu`
- Base packages: base, datasets, graphics, grDevices, methods, splines, stats, utils
- Other packages: cacheSweave 0.4-4, class 7.3-2, codetools 0.2-2, filehash 2.1, getopt 1.15, gss 1.1-0, lattice 0.18-5, MASS 7.3-5, mgcv 1.6-2, pgfSweave 1.0.5, ppc 1.01, R2HTML 2.0.0, STAR 0.3-4, stashR 0.3-3, survival 2.35-8, tikzDevice 0.4.8, xtable 1.5-6
- Loaded via a namespace (and not attached): digest 0.4.2, grid 2.11.0, Matrix 0.999375-38, nlme 3.1-96, tools 2.11.0

Index