Intensity Estimation with STAR: short version

Christophe Pouzat

Laboratoire de Physiologie Cérébrale, CNRS UMR 8118
UFR biomédicale de l'université Paris-Descartes
45, rue des Saints-Pères
75006 Paris, France

August 24, 2009

Contents

1 Introduction

What follows is a detailed description of how I presently estimate the *intensity function* (*IF*)[1] of spike trains recorded in the *spontaneous regime* using STAR functionalities.

This a "short" version of the vignette describing rather breifly the analysis of a single spike train. A longer,more comprehensive version can be found in the STAR web site[2].

1.1 Some jargon

Before entering into the details of the analysis some technical terms that are going to be used constantly in the sequel will be introduced. The notations follow mainly the ones of Brillinger [1988a, pp 190–191].

Definition 1 *For points* $\{t_j\}$ *randomly scattered along a line, the* counting process $N(t)$ *gives the number of points observed in the interval* $(0, t]$*:* *counting process* definition

$$N(t) = \sharp\{t_j \text{ with } 0 < t_j \leq t\} \tag{1}$$

where $\sharp$ *stands for the cardinality (number of elements) of a set.*

Brillinger [1988a] uses τ_j for our t_j.

Definition 2 *The* history*,* $\mathcal{H}_t$*, consists of the variates determined up to and including time* t *that are necessary to describe the evolution of the* counting process*.* *history* definition

The *history* is often called the *filtration* in the *counting process* literature. See Touboul and Faugeras [2007, p 93] for a rigorous definition of the concept, see also Andersen et al. [1993, pp 49–51].

Definition 3 *For the process* N *and* history $\mathcal{H}_t$*, the* intensity function *at time* t *is defined as:* *intensity function* definition

$$\lambda(t \mid \mathcal{H}_t) = \lim_{h \downarrow 0} \frac{\text{Prob}\{\text{event } \in (t, t+h] \mid \mathcal{H}_t\}}{h} \tag{2}$$

[1] "Our" *intensity function* is also often called the *conditional intensity function*, *e.g.*, Brillinger [1988a], Ogata [1988] or the *hazard function*, *e.g.*, Johnson [1996]. The *intensity function* should be called more properly the *intensity process*since it is in general a function of random variables Andersen et al. [1993, p 51]

[2] http://sites.google.com/site/spiketrainanalysiswithr/

For small h one has the interpretation:

$$\text{Prob}\{\text{event } \in (t, t+h] \mid \mathcal{H}_t\} \approx \lambda(t \mid \mathcal{H}_t)\, h \tag{3}$$

Notice that we are using symbol λ for the *intensity function*, following the now most usual convention Andersen et al. [1993, p 51], while Brillinger [1988a], Johnson [1996] use μ.

1.2 Loading STAR

We will start with the analysis of the discharge of `neuron 1` in data set: `e060824spont`. I assume that you have installed the last version of STAR from your favorite CRAN server. Then the first thing to do once R has been started is to load the package: library

```
> library(STAR)
```

Here some of the stuff printed upon loading the package has been removed.

2 Analysis of data from neuron 1 of e060824spont data set

2.1 Loading data

Fine, we now have to make the data set `e060824spont` available from our work space, and this is done with function data: data

```
> data(e060824spont)
```

2.2 Summarizing data

We start by getting a quick summary of neuron 1 spike train by applying the summary method to the spikeTrain object, `e060824spont[["neuron 1"]]`[3]: summary

```
> summary(e060824spont[["neuron 1"]])
```

```
A spike train with 505 events, starting at: 0.594 and ending at: 58.585 (s).
The mean ISI is: 0.115 and its SD is: 0.36 (s).
The mean log(ISI) is: -3.148 and its SD is: 1.044
The shortest interval is: 0.008
 and the longest is: 3.811 (s).
```

2.3 Automatic analysis

2.3.1 reportHTML

We next carry out an "automatic analysis" using method reportHTML: reportHTML

[3] As can be seen by looking at the documentation of the data set (`?e060824spont`), `e060824spont` is a list of two spikeTrain objects.

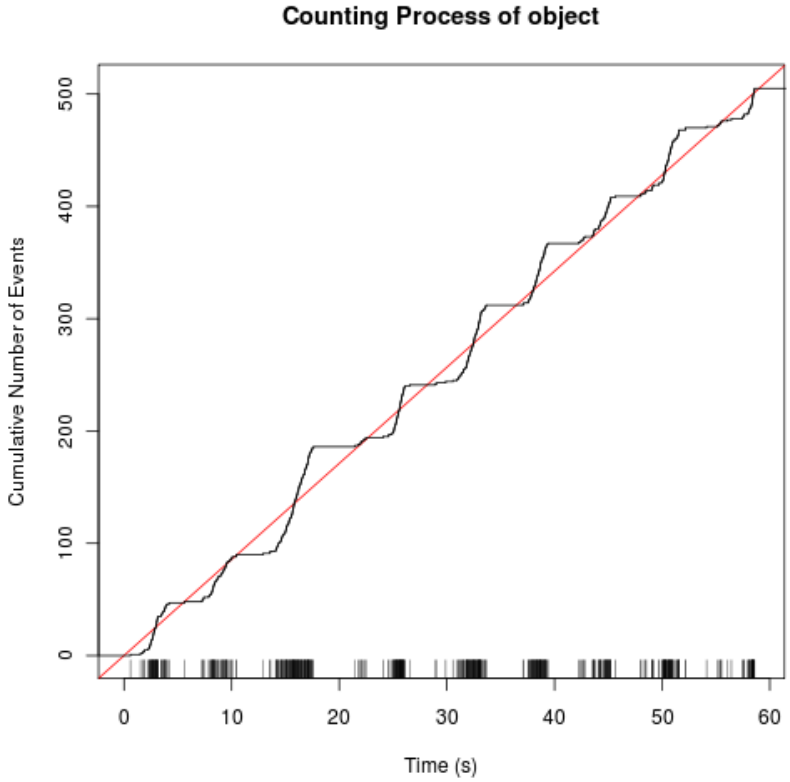

Figure 1: The spike train plot of neuron 1 of data set `e060824spont`.

```
> reportHTML(e060824spont[["neuron 1"]], filename = "e060824spont_1",
+     directory = "report", otherST = e060824spont[c(2)], maxiter = 100)
```

The result of this automatic analysis is a bunch of figures in `png` format and an `html` file named `e060824spont_1.html` and located in subdirectory `report`. The best way to visualize the `html` file is clearly to use your favorite web browser.

2.3.2 Spike train plot

The first figure appearing on the web page (`e060824spont_1.html`) is a spike train plot [Pouzat and Chaffiol, 2009] and is reproduced in Fig. 1. A striking staircase pattern can be seen on the realization of the *counting process* defined by Eq. (1). This pattern which translates into the non-uniform distribution of the ticks on the *raster plot* shown at the bottom of the graph rules out a model based on a *homogenous Poisson process* for this spike train.

Intensity function* of an *homogenous Poisson process The *IF* of an *homogenous Poisson process* is extremly simple. One has:

$$\lambda(t \mid \mathcal{H}_t) = \lambda_0 \tag{4}$$

That is, the *IF* is a constant.

Tip When dealing with spike train with a lot of events, say 1000 or more, the extra "visibility" provided by the spike train plot compared to the classical *raster plot*, can be defficient in the sense that important details of the discharge can end up being not discernible. It is then easy to use the subsetting method for spikeTrain objects which would give in the present case:

Subsetting spikeTrain objects

```
> e060824spont[[1]][10 <= e060824spont[[1]] & e060824spont[[1]] <
+      40]
```

The resulting spike train plot is not shown in this document, but a "zoom" of Fig. 1 between seconds 10 and 40 would pop-up.

2.3.3 renewalTestPlot

As explained in Pouzat and Chaffiol [2009, Sec. 2.4.3], the model "following" the *homogenous Poisson process* is the *homogenous renewal process*. A graphical plot of the suitability of such a model for empirical data is the second graph appearing on the web page, `e060824spont_1.html`, and is generated by function renewalTestPlot. We show it here on Fig. 2 from which it is clear that a *homogenous renewal process* model does not apply.

Intensity function* of a *homogenous renewal process The *IF* of a *homogenous renewal process* is still reasonably simple. One has:

$$\lambda(t \mid \mathcal{H}_t) = r(t - t_l) \tag{5}$$

where t_l is the occurence time of the last spike before t, formally, $t_l = \max\{t_j : t_j < t\}$. In other words, $t - t_l$ is the elapsed time since the last spike. It is as if the clock was reset at 0 everytime an event occurs. For a *homogenous renewal process* the *history* is simply made of all the spikes observed up to, but not including, t: $\{t_j : t_j < t\}$.

2.3.4 Cross correlation histograms and Cross-intensity plots

The web page shows next two plots which are relevant only when the *homogenous renewal process* applies. They are not reproduced here since a more sophisticated model is required as shown by Fig. 2. The last plots showing the *cross-correlation histogram* [Brillinger et al., 1976, Eq. (13), p 218] and its smooth version, the *cross-intensity plot*, is reproduced here on Fig. 3. Since this data sets contains only two neurons, only one such plot appears on the web page. With more neurons in the data set, more plots

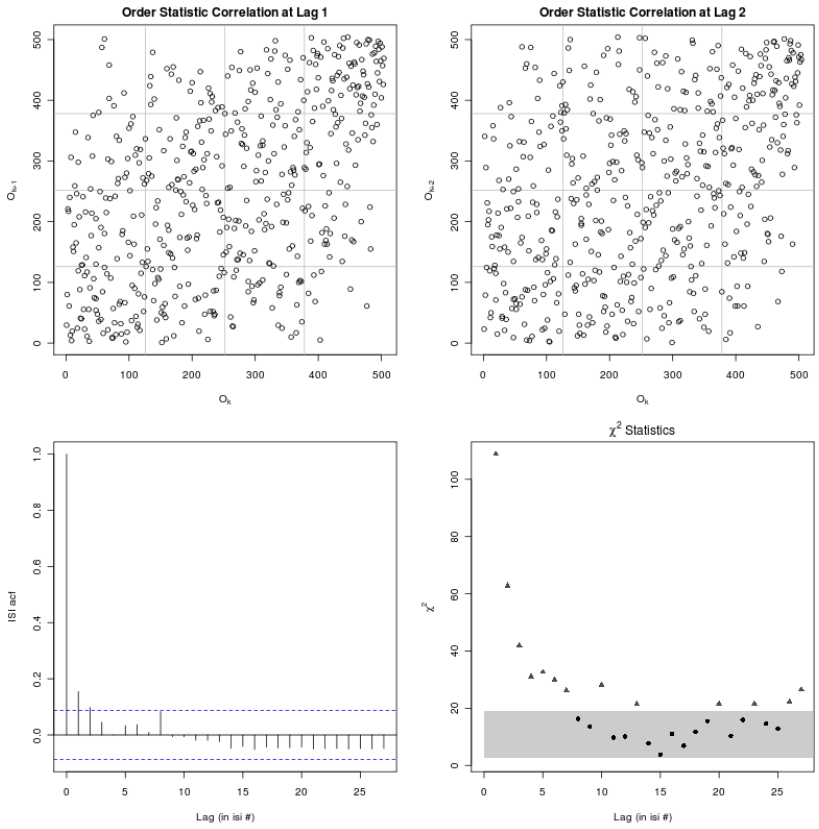

Figure 2: Renewal test plot of neuron 1 of data set `e060824spont`.

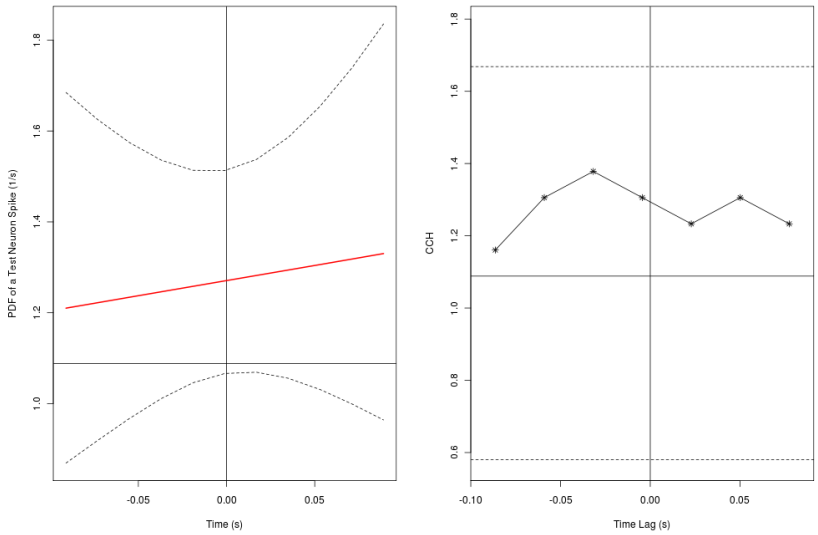

Figure 3: Cross-intensity plot and Cross correlation histogram between neuron 1 and 2 of data set `e060824spont`.

can be generated by setting properly argument `otherST` of method reportHTML. Since the black horizontal lines on Fig. 3 is entirely contained in the "confidence region", there is no ground to include an interaction term between the two recording neurons in our discharge model for neuron 1.

2.3.5 Conclusions of the automatic analysis

This automatic analysis leads us to conclude that our model needs to be more complex than a *homogenous renewal process* model (Fig. 2). The absence of significant cross-correlation (Fig. 3) suggests that an interaction term between neurons 1 and 2 of the data set is not required. Moreover neither the spike train plot (Fig. 1) nor the auto-correlation function plot (bottom left of Fig. 2) show clear signs of non stationnarity of the train. *At the present stage we do not have any method leading to unambiguously interpretable models with non stationnary data in the spontaneous regime.*

A model more complex than a *homogenous renewal process* model will necessarily lead us to a multivariate *IF*. Biophysics teaches us that every neuron exhibits a refractory period following a spike (ruling out the *homogenous Poisson process* as a "true" model) and that will lead us to always include the elapsed time since the last spike in our models; just as we did for the *homogenous renewal process* model of Eq. (5). Of course the bothering question at this stage is: What the extra variables in our *IF* model should be? A "natural" way to include interactions between neurons would be to add the elapsed time since the last spike of a "functionally" coupled neuron in our variables list. But as we just saw for the present data set such an additional variable does not seem necessary. We

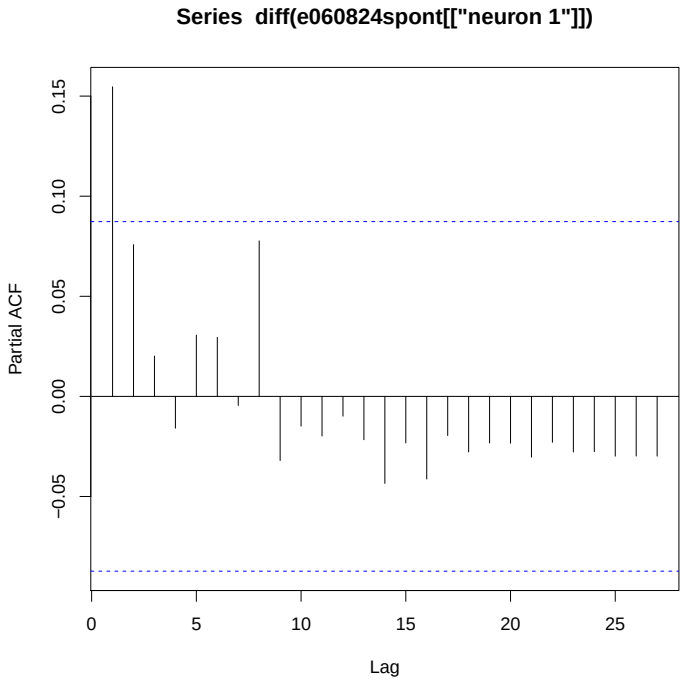

Figure 4: The *partial autocorrelation function* of neuron 1 of data set `e060824spont`.

are therefore left with the occurrence times of the other previous spikes, or equivalently, with the duration of the previous *inter spike interval*s (*isi*s). The question becomes then: how many previous *isi*s should we include in our variables list? The next section presents a tool providing us with a first guess.

2.4 Partial autocorrelation function

A practical guidance on how many past *isi*s should be included is provided by the *partial autocorrelation function* of the *isi*s [Kuhnert and Venables, 2005, pp 77–79]. A graph of this function for the present data set is shown on Fig. 4. It is obtained with command: `acf`

```
> acf(diff(e060824spont[["neuron 1"]]), type = "partial")
```

What we should look at here are the lags at which the function is out of 95% "confidence intervals", like lag 1 for this data set.

This initial analysis would lead us to a model like:

$$\lambda(t \mid \mathcal{H}_t) = f(t - t_l, i_1) \tag{6}$$

where i_1 is the duration of the last *isi*.

Tip In practice when the *homogenous renewal process* model does not apply, I always include the last *isi* in the model variables list even if *partial autocorrelation function* is not out of the confidence intervals at lag 1. I include in addition all the other *isi*s for which it is out.

2.5 Data frame for gss

We will follow the approach of Brillinger [1988a, p 191], where for computational convenience, a discretization of the spike train is performed. That is, we go from the "actual train", $\{t_1, \ldots, t_n\}$ where $0 < t_1, \ldots, t_n \leq T$, to a binary vector, *event*, whose jth element is the value of the sum over $\{t_1, \ldots, t_n\}$ of the indicator function I_j defined by:

time discretization and vector *event*

$$I_j(t) = \begin{cases} 1 & \text{if } (j-1)\delta < t \leq j\delta \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Where the "bin width", δ, is chosen small enough to have at most one spike per bin[4]. More explicitly we have:

$$event_j = \sum_{k=1}^{n} I_j(t_k) \tag{8}$$

When we work with this binary vector *event* we do not estimate $f(t - t_l, i_1)$ directly anymore but:

$$f_\delta(t - t_l, i_1) \equiv f\left((j - j_l)\delta, (j_l - j_{l-1})\delta\right)\delta \tag{9}$$

where j is be index of the bin containing time t, j_l is the index of the bin of the previous spike and j_{l-1} is the index of the second previous spike. f_δ should be a probability (if δ has indeed been set small enough), that is a number between 0 and 1. This is what Brillinger [1988a, Eq. (2.5), p 191] writes p_t (his t being our j).

Since Biophysics doesn't help us much beyond the presence of a refractory period, it is hard to guess what $f(t - t_l, i_1)$ of Eq. (6) or $f_\delta(t - t_l, i_1)$ of Eq. (9) should look like. We will therefore use a nonparametric approach where $f_\delta(t - t_l, i_1)$ will be estimated with a *penalized likelihood* method. The general features of this approach are describe briefly in Pouzat and Chaffiol [2009, Sec. 2.5.2] and in depth in Gu [2002].

The model estimation will moreover be performed by function gssanova of Chong Gu's package gss. The data "fed" to this function have to be in a data frame format. Function mkGLMdf of STAR will allow us to build a data frame from a spikeTrain object. Since our preliminary analysis lead us to rule out an interaction between the two neurons of our data set, we do not need to include a variable containing the ellapsed time since the last spike of neuron 2 in our data frame. Our initial summary taught us that the shortest *isi* was 8 ms long and that events were obeserved between 0 and 59 s. We will therefore use a bin width of 4 ms and create our data frame with:

mkGLMdf

[4] This type of discretization is referred to by Berman and Turner [1992, pp 33–34] as a *probabilistic approximation*, they propose an alternative *numerical approximation* where the bin width, δ, is allowed to change along the time axis. The quantity being approximated is the likelihood of intensity [Pouzat and Chaffiol, Sec. 2]. Chornoboy et al. [1988] present an approach similar to the one of Brillinger [1988a] albeit with a different motivation. Brillinger did moreover use this probabilistic approximation from the early eighties on as witnessed by his 1983 "Wald Memorial Lecture" [Brillinger, 1988b, p 34].

```
> DFA <- mkGLMdf(e060824spont[["neuron 1"]], 0.004, 0, 59)
```

We can get a quick view of the first elements of our data frame `DFA` with: head

```
> head(DFA)
```

```
    event  time neuron  lN.1
150     0 0.596      1 0.004
151     0 0.600      1 0.008
152     0 0.604      1 0.012
153     0 0.608      1 0.016
154     0 0.612      1 0.020
155     0 0.616      1 0.024
```

Here the variables are:

- `event` corresponds to our previous *event* vector, it contains the binary version of the spike train.
- `time` contains the time at the bin center (in s).
- `neuron` contains the number of the considered neuron in the data set (the one to which the spikes in the `event` variable belong). It wont be used here but it becomes useful when several neurons are present and when interactions between them have to be considered as we will later see.
- `lN.1` contains the elapsed time since the last spike of neuron 1, that is, $j - j_l$ in Eq. (9).

We can also get a quick view at the end of `DFA` with: tail

```
> tail(DFA)
```

```
      event   time neuron  lN.1
14746     0 58.980      1 0.396
14747     0 58.984      1 0.400
14748     0 58.988      1 0.404
14749     0 58.992      1 0.408
14750     0 58.996      1 0.412
14751     0 59.000      1 0.416
```

There is still one variable missing in our data frame in order to work with our candidate model: the last *isi* which can be obtained with function isi of STAR: isi

```
> DFA <- within(DFA, i1 <- isi(DFA, lag = 1))
```

Here we have just added variable `i1` to our data frame. As we can see by calling head on our modified `DFA`:

```
> head(DFA)
```

```
    event  time neuron  lN.1 i1
150     0 0.596      1 0.004 NA
151     0 0.600      1 0.008 NA
152     0 0.604      1 0.012 NA
153     0 0.608      1 0.016 NA
154     0 0.612      1 0.020 NA
155     0 0.616      1 0.024 NA
```

values of `i1` are not available for the first elements of the data frame. It makes sense since we do not know when was the last spike before the beginning of the acquisition. We are therefore going to remove the elements of `DFA` for which one of the variables is not available. We can do that efficiently with function complete.cases of R: complete.cases

```
> DFA <- DFA[complete.cases(DFA), ]
```

Calling head again, we can see that complete.cases did its job right:

```
> head(DFA)
```

```
    event  time neuron  lN.1    i1
436     0 1.740      1 0.004 0.216
437     0 1.744      1 0.008 0.216
438     0 1.748      1 0.012 0.216
439     0 1.752      1 0.016 0.216
440     0 1.756      1 0.020 0.216
441     0 1.760      1 0.024 0.216
```

We can moreover check that isi did its job correctly by looking at a well chosen part of `DFA`, like the one starting at index 14169:

```
      event   time neuron  lN.1    i1
14604     0 58.412      1 0.012 0.016
14605     1 58.416      1 0.016 0.016
14606     0 58.420      1 0.004 0.016
14607     1 58.424      1 0.008 0.016
14608     0 58.428      1 0.004 0.008
14609     0 58.432      1 0.008 0.008
14610     1 58.436      1 0.012 0.008
14611     0 58.440      1 0.004 0.012
```

2.6 Variables transformation

A crucial ingredient for efficient smoothing spline estimation is an a "reasonably" uniform distribution of the independent variables (or predictors). But as Fig. 5 shows our independent variables, `lN.1` and `i1` are not uniformly distributed.

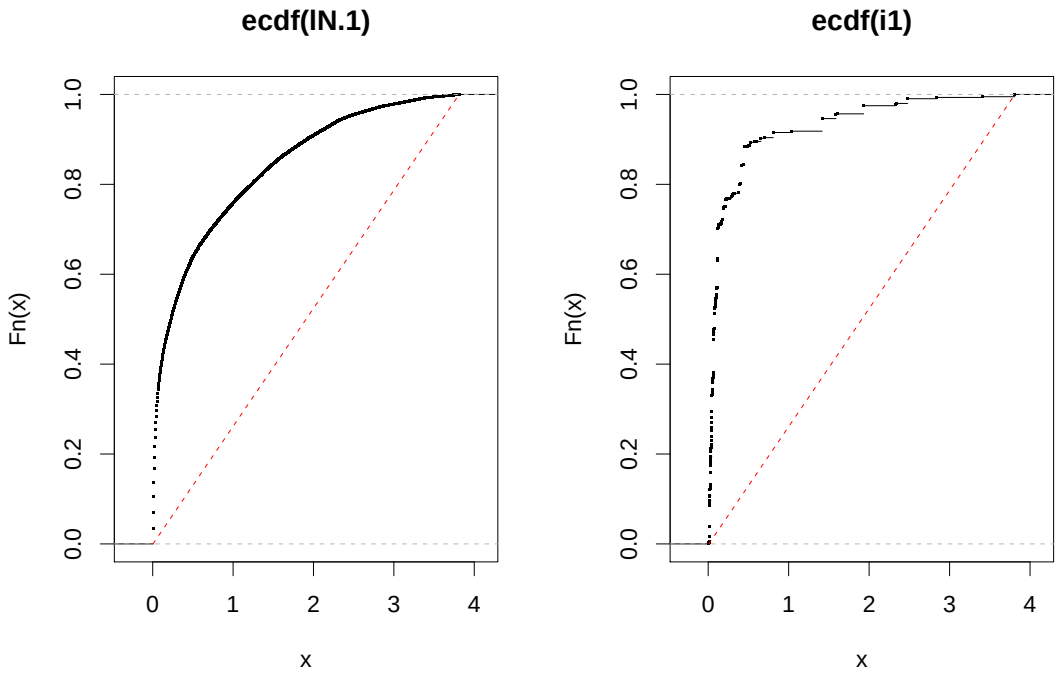

Figure 5: Empirical cumulative distribution function of `lN.1` and `i1`.

```
> with(DFA, plot(ecdf(lN.1), pch = "."))
> with(DFA, lines(range(lN.1), c(0, 1), col = 2, lty = 2))
> with(DFA, plot(ecdf(i1), pch = "."))
> with(DFA, lines(range(i1), c(0, 1), col = 2, lty = 2))
```

In the sequel we will adopt the slightly extrem "mapping to uniform" approach, that is, we are going to estimate a smooth version of the *cumulative distribution function* (*cdf*) and use it to transform our independent variables. The STAR function doing this job is mkM2U. It can be called on part of the data set in order to perform a mapping of the other part independent of the (mapped) data. For our two variables `lN.1` and `i1` we call:

mkM2U

```
> m2u1 <- mkM2U(DFA, "lN.1", 0, 28.5)
> m2ui <- mkM2U(DFA, "i1", 0, 28.5, maxiter = 200)
```

The results of these two commands are two mapping functions that we can now use to generate the "mapped to uniform" variables, `e1t` and `i1t`:

```
> DFA <- within(DFA, e1t <- m2u1(lN.1))
> DFA <- within(DFA, i1t <- m2ui(i1))
```

Fig. 6 shows us that our mapping worked properly.

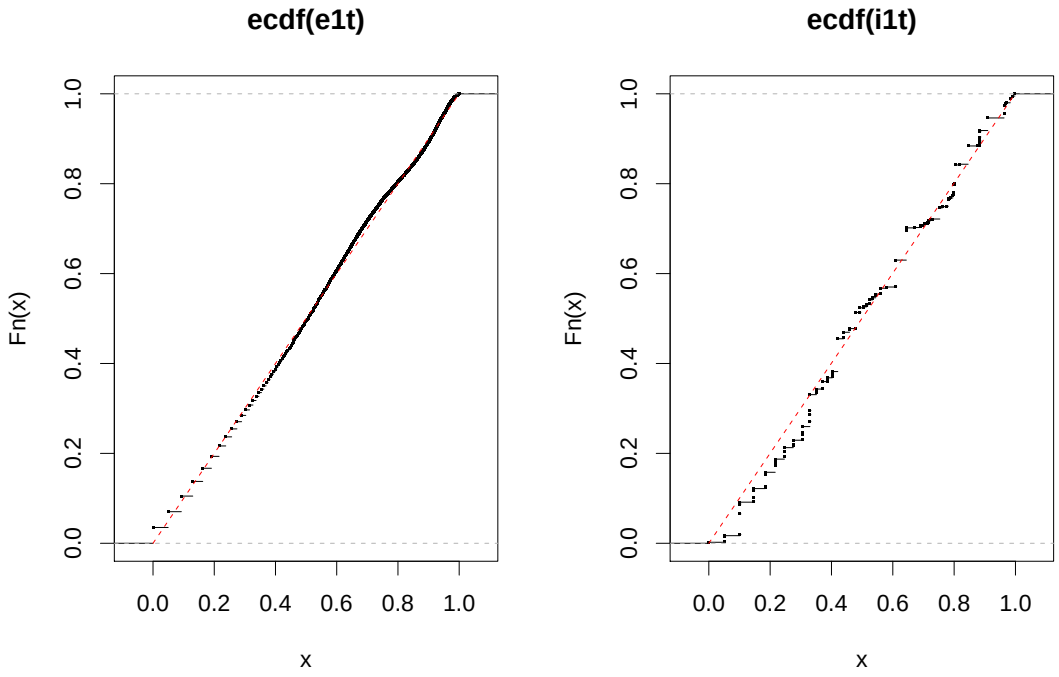

Figure 6: Empirical cumulative distribution function of `e1t` and `i1t`.

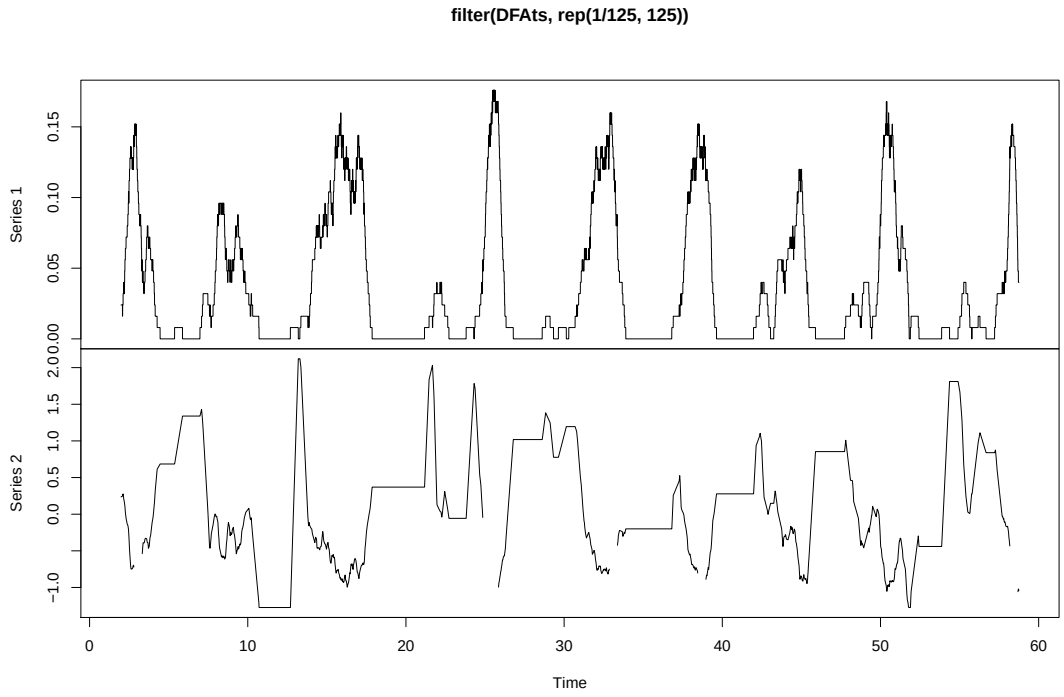

Figure 7: Time evolution of `event` and `i1t`. A box filter of 0.5 s has been applied

2.7 Variables evolution

Before going further it can be a good idea to look at the (time) evolution of the variables. In order to do that quickly we are going to use the default *time series objects* provided by R and created with function ts. Looking at `e1t` should not be too interesting since this variable is starting at zero following an event and increases linearly thereafter. We will therefore look at our `event` and `i1t` variables. In order to have a clearer picture we are going to box-filter the variables using function filter with a widow length of 0.5 s (that is, 125 indexes, since we used a bin width of 4 ms). We will also map `i1t` onto a normal random variable and we get Fig. 7 : ts filter

```
> DFAts <- ts(with(DFA, cbind(event, qnorm(i1t))), start = DFA$time[1],
+     delta = diff(DFA$time[1:2]))

> plot(filter(DFAts, rep(1/125, 125)))
```

Fig. 7 does not exhibit any clear trend in the graphed variables, confirming thereby our former stationary discharge conclusion. Depending on the data at hand it can clearly be a good idea to try out several filter window lengths.

2.8 Fitting and testing models

Since we are going to use nonparametric model estimation procedures and since we want to have meaningful goodness of fit tests we will systematically fit a given model to one half of the data and test it on the other half before switching the fit and test data parts and repeating the procedure.

2.8.1 Model fit: the straightforward approach

We are going to fit our models using function gssanova of package gss. The most straightforward way to fit the model of Eq. (6) to the first half of our data set is: gssanova

```
> GF1e <- gssanova(event ~ e1t * i1t, data = subset(DFA, time <=
+     29.5), family = "binomial", seed = 20061001)
```

The time needed to carry out this fit on the present machine is, 92.11 s[5]

2.8.2 Time transformation and goodness of fit

We will asses the quality of our model by evaluating the *intensity process* of the part of the data taht we did not use fro model estiamtion. This *intensity process* will then be used to performed a *time transformation* as proposed by Ogata [1988] after which a new *counting process* will be obtained. If our model is good this process should be the realization of a *homogenous Poisson process* with rate 1. The latter process is then the null hypothesis against which we are going to test Pouzat and Chaffiol. The time transformation is simply performed with function %tt% of STAR. It is called as follows: %tt%

```
> tt.GF1e <- GF1e %tt% subset(DFA, time > 29.5)
```

Object `tt.GF1e` is an object of class CountingProcessSamplePath for which a summary method exists providing a quick numeric summary of how appropriate the model is:

```
> tt.GF1e.summary <- summary(tt.GF1e)

> tt.GF1e.summary

 *** Test of uniformity on the time axis
     Prob. of the Kolmogorov statistic under H0: 0.25209
 *** Wiener process test
     Inside 95% domain: TRUE , inside 99% domain: TRUE
 *** Berman test
     Prob. of the Kolmogorov statistic under H0: 0.11301
 *** Renewal test
```

[5]For reference, it takes 88.23 s on an Intel Core2 Duo P9500 at 2.53 GHz with 4 GB of RAM, running Ubuntu 9.04, R-2.9.1 link to the ATLAS version of BLAS (version 3.8.3) everything being compiled with gcc 4.3.3.

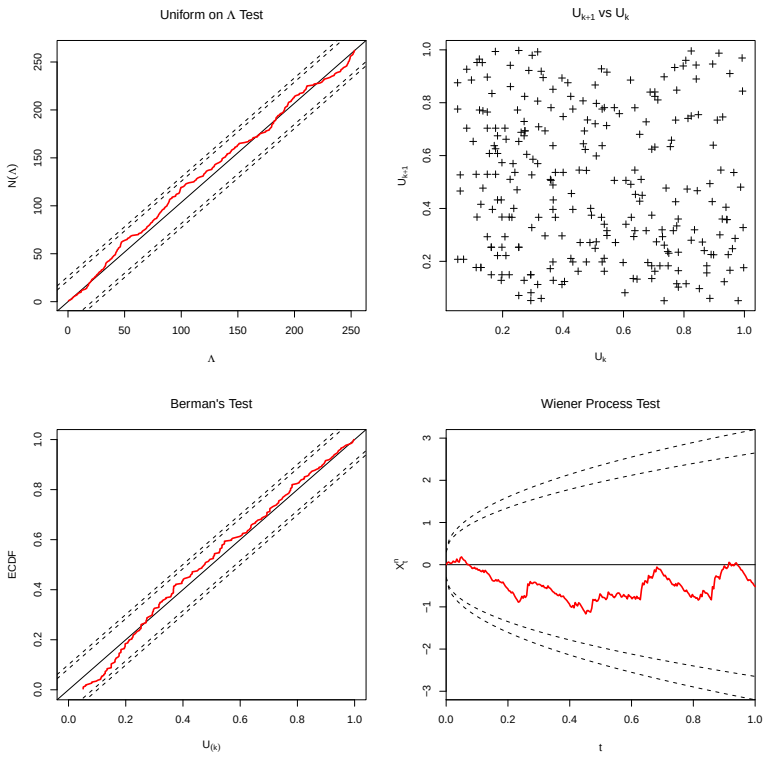

Figure 8: Ogata's tests battery applied to the time transformed second half of neuron 1 spike train (from data set: `e060824spont`) using model `GF1e` fitted on the first half. See Pouzat and Chaffiol for a description of the plots.

```
    Inside 95% domain: FALSE , inside 99% domain: TRUE
    Maximum lag: 24
*** Variance vs "time" with 5 time windows:
    1 window out at 95% level
    0 window out at 99% level
*** The object contains 262 events.
```

Notice that the last two commands could be combined in a single one by typing:

```
> (tt.GF1e.summary <- summary(tt.GF1e))
```

The quality of the model can also be assest by calling the plot method for CountingProcessSamplePath objects as shown on Fig. 8:

```
> plot(tt.GF1e.summary, which = c(1, 2, 4, 6))
```

Looking at Fig. 8 we would conclude that the model is satisfying.

2.8.3 Exchanging fitting and testing part

In order to fully validate our model we are going to exchange the fitting and testing part, that is, fit the same model as before to the last half of the data set before testing it on the first half:

```
> GF1l <- gssanova(event ~ e1t * i1t, data = subset(DFA, time >
+     29.5), family = "binomial", seed = 20061001)
```

The total time taken by our two fits is: 194.68 s. We now perform the same series of tests than before but this time on the early part of the data set:

```
> tt.GF1l <- GF1l %tt% subset(DFA, time <= 29.5)

> (tt.GF1l.summary <- summary(tt.GF1l))

 *** Test of uniformity on the time axis
     Prob. of the Kolmogorov statistic under H0: 0.44026
 *** Wiener process test
     Inside 95% domain: TRUE , inside 99% domain: TRUE
 *** Berman test
     Prob. of the Kolmogorov statistic under H0: 0.14104
 *** Renewal test
     Inside 95% domain: FALSE , inside 99% domain: TRUE
     Maximum lag: 24
 *** Variance vs "time" with 4 time windows:
     0 window out at 95% level
     0 window out at 99% level
 *** The object contains 240 events.

> plot(tt.GF1l.summary, which = c(1, 2, 4, 6))
```

Looking at Fig. 9 we would conclude again that the model is satisfying.

2.8.4 Doing two fits at once with a multi-core CPU

See the long version of the vignette on the STAR web site[6].

2.8.5 Trying a simpler model

We have just explored a model containing an "interaction" term between variable `e1t` and variable `i1t`. Since the latter gives good fits it is interesting to try simplifying it to see if a model without interaction would not give as good results we proceed as follows:

[6] http://sites.google.com/site/spiketrainanalysiswithr/

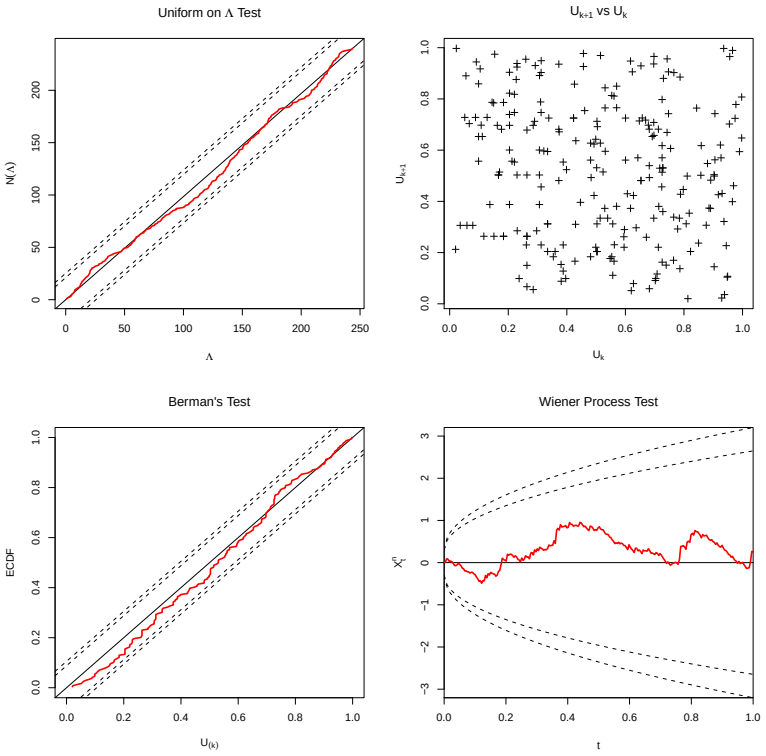

Figure 9: Ogata's tests battery applied to the time transformed first half of neuron 1 spike train (from data set: `e060824spont`) using model `GF11` fitted on the second half.

```
> GF2e <- gssanova(event ~ e1t + i1t, data = subset(DFA, time <=
+     29.5), family = "binomial", seed = 19731004)
> tt.GF2e <- GF2e %tt% subset(DFA, time > 29.5)
> (tt.GF2e.summary <- summary(tt.GF2e))

 *** Test of uniformity on the time axis
     Prob. of the Kolmogorov statistic under H0: 0.24169
 *** Wiener process test
     Inside 95% domain: TRUE , inside 99% domain: TRUE
 *** Berman test
     Prob. of the Kolmogorov statistic under H0: 0.07561
 *** Renewal test
     Inside 95% domain: TRUE , inside 99% domain: TRUE
     Maximum lag: 24
 *** Variance vs "time" with 5 time windows:
     2 windows out at 95% level
     1 window out at 99% level
 *** The object contains 262 events.

> GF2l <- gssanova(event ~ e1t + i1t, data = subset(DFA, time >
+     29.5), family = "binomial", seed = 19731004)
> tt.GF2l <- GF2l %tt% subset(DFA, time <= 29.5)
> (tt.GF2l.summary <- summary(tt.GF2l))

 *** Test of uniformity on the time axis
     Prob. of the Kolmogorov statistic under H0: 0.12505
 *** Wiener process test
     Inside 95% domain: TRUE , inside 99% domain: TRUE
 *** Berman test
     Prob. of the Kolmogorov statistic under H0: 0.05779
 *** Renewal test
     Inside 95% domain: TRUE , inside 99% domain: TRUE
     Maximum lag: 24
 *** Variance vs "time" with 4 time windows:
     1 window out at 95% level
     0 window out at 99% level
 *** The object contains 240 events.
```

The fit diagnostic plots are shown on Fig. 10. Since the simpler model also looks good, the question becomes: Which one should we choose?

2.9 Model selection

One way to compare two alternative models is to look at the probability they give to data *which were not the data used to fit them.* This can be done with function predictLogProb of

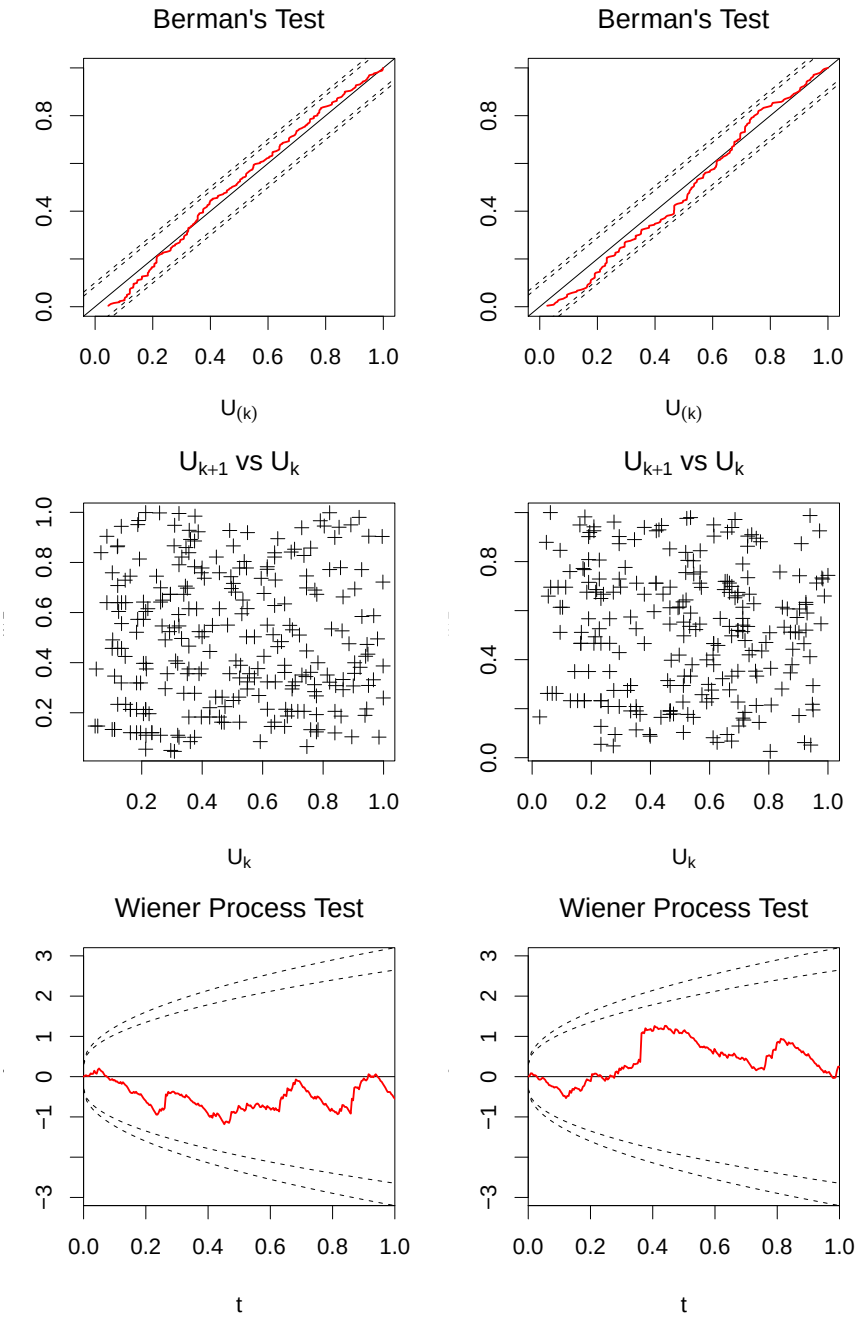

Figure 10: Ogata's tests battery applied to the time transformed neuron 1 spike train (from data set: `e060824spont`) using model `GF2`. Left column, fit first half, test second half (compare with Fig. 8). Right column, fit second half, test first (compare with Fig. 9). The first tests (upper right) of Fig. 8 and 9 are not shown here but are "passed" (*i.e.*, within the confidence bands).

STAR which returns the log probability of some data (passed as the second argument to the function) under some model (passed as the first argument). Here the log probability of our data using the simpler model is: predictLogProb

```
> (GF2.logProb <- predictLogProb(GF2e, subset(DFA, time > 29.5)) +
+     predictLogProb(GF2l, subset(DFA, time <= 29.5)))

[1] -1775.852

[1] -1775.852

> (GF1.logProb <- predictLogProb(GF1e, subset(DFA, time > 29.5)) +
+     predictLogProb(GF1l, subset(DFA, time <= 29.5)))

[1] -1763.006

[1] -1763.006
```

Since the most complex model (with interaction) gives a higher probability than the less complex one (without interaction) I would go ahead and keep the former.

2.10 Plotting results

2.10.1 Quick visualization of the model terms

Before looking at the model terms effect we would normaly refit our selected model to the full data set in orer to have better estimates with[7]:

```
> GF1f <- gssanova(event ~ e1t * i1t, data = DFA, family = "binomial",
+     seed = 20061001)
```

We use here the fit obtained from the first half of the data set (`GF1e`). A plot of the terms is then quickly generated with the plot method for gssanova objects:

```
> plot(GF1e, nr = 3, nc = 1)
```

2.10.2 Use of quickPredict and its associated methods

A finner control of the plots can be obtained with the quickPredict function of STAR and of its associated plot, contour, image and persp methods. The easiest way to fine tune a term effect plot with STAR is to generate a quickPredict object containing the term effect first. For the first two terms, `e1t` and `i1t` of our model this is done simply with: quickPredict

```
> term.e1t <- quickPredict(GF1e, "e1t")
```

or, using the binary operator version, %qp%: %qp%

[7] We do not do it in this short version but the results are presented in the "long" version of the STAR web site.

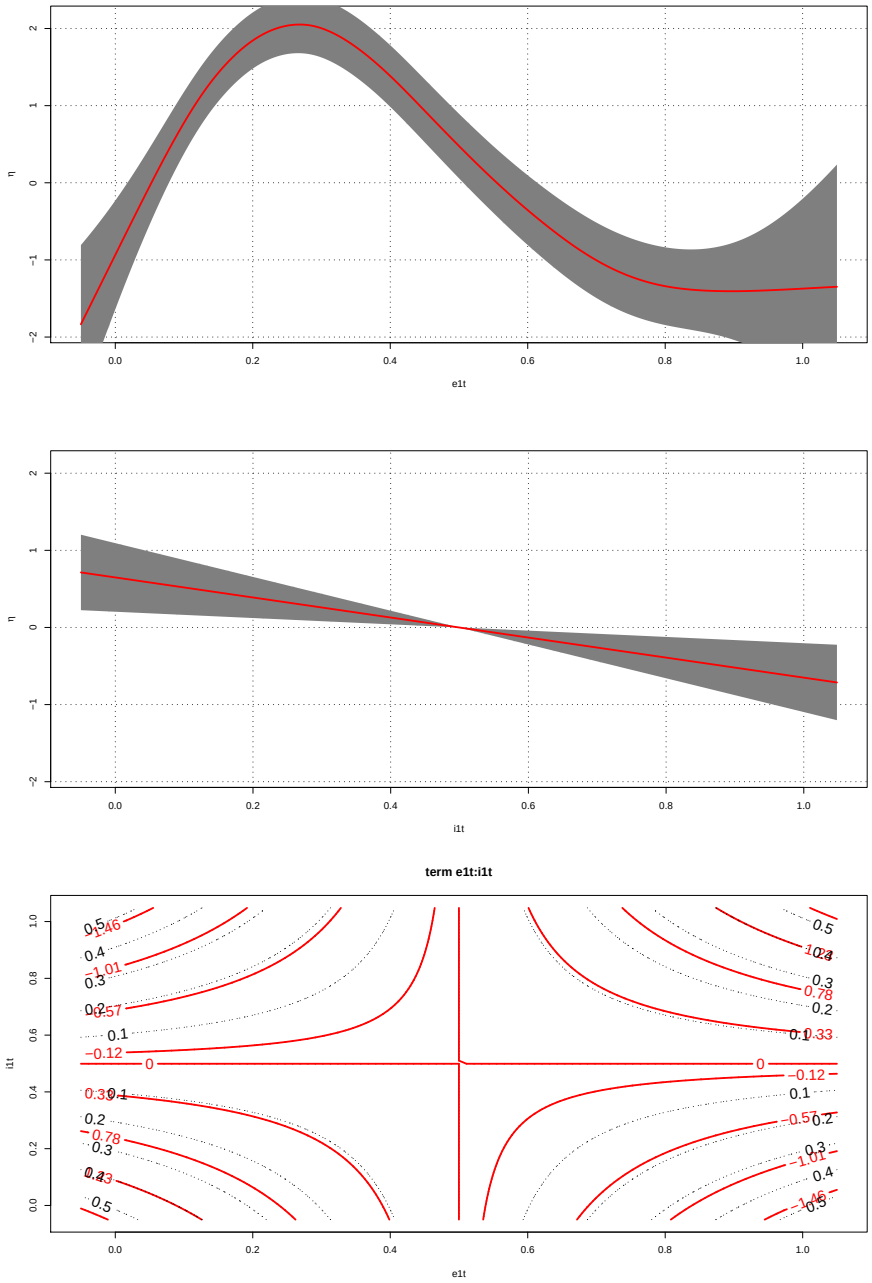

Figure 11: Effects of the 3 terms of the selected model for neuron 1 of data set `e060824spont`. The abscissa scale corresponds to the percentiles of the variables. The ordinate logit scales are directly comparable. On the third plot the estimated interaction term is displayed as red contours while the estimated standard error is displayed as dotted black contours.

```
> term.i1t <- GF1e %qp% "i1t"
```

We can then call the plot method for quickPredict objects and get basic plots. We can also pass additional arguments to these methods in order to fine tune the output. Another thing we can do is get a plot of the term effects on the "native" scale instead of the "probability" scale. To do that we can use the qFct attribute of our "mapping to uniform" functions (Sec. 2.6). What we have to transform is the xx element of our two quickPredict objects, `term.e1t` and `term.i1t`:

```
> term.e1 <- term.e1t
> term.e1$xx <- attr(m2u1, "qFct")(term.e1$xx)
> term.i1 <- term.i1t
> term.i1$xx <- attr(m2ui, "qFct")(term.i1$xx)
```

We can then use the plot method to get Fig. 12:

```
> plot(term.e1t, xlab = "Probability scale", ylab = expression(eta[1]),
+     main = "Elapsed time since last spike")
> plot(term.e1, xlab = "Time (s)", ylab = expression(eta[1]), panel.first = grid(col = 1),
+     main = "Elapsed time since last spike")
> plot(term.i1t, xlab = "Probability scale", ylab = expression(eta[i1]),
+     main = "Last ISI")
> plot(term.i1, xlab = "Time (s)", ylab = expression(eta[i1]),
+     main = "Last ISI", panel.first = grid(col = 1))
```

The quickPredict object corresponding to the interaction term of the model, `e1t:i1t`, is also easily obtained:

```
> term.e1ti1t <- GF1e %qp% "e1t:i1t"
```

We can call the image, contour persp methods on the resulting object. If we want to go to the native scale for the plot, the best way is to use the changeScale function of STAR: changeScale

```
> term.e1i1 <- changeScale(term.e1ti1t, attr(m2u1, "qFct"), attr(m2ui,
+     "qFct"))
```

The following commands: image, contour, persp

```
> image(term.e1ti1t)
> contour(term.e1ti1t, add = TRUE)
> contour(term.e1ti1t, levels = seq(-2, 2, 0.5), labcex = 1.5,
+     col = 2)
> contour(term.e1ti1t, what = "sd", levels = seq(-0.4, 0.4, 0.1),
+     col = 1, lty = 2, add = TRUE)
> persp(term.e1ti1t, theta = -10, phi = 30)
> persp(term.e1i1, theta = -25, phi = 30, xlab = "time since last (s)",
+     ylab = "last isi (s)", main = "")
```

lead to the plots shown on Fig. 13.

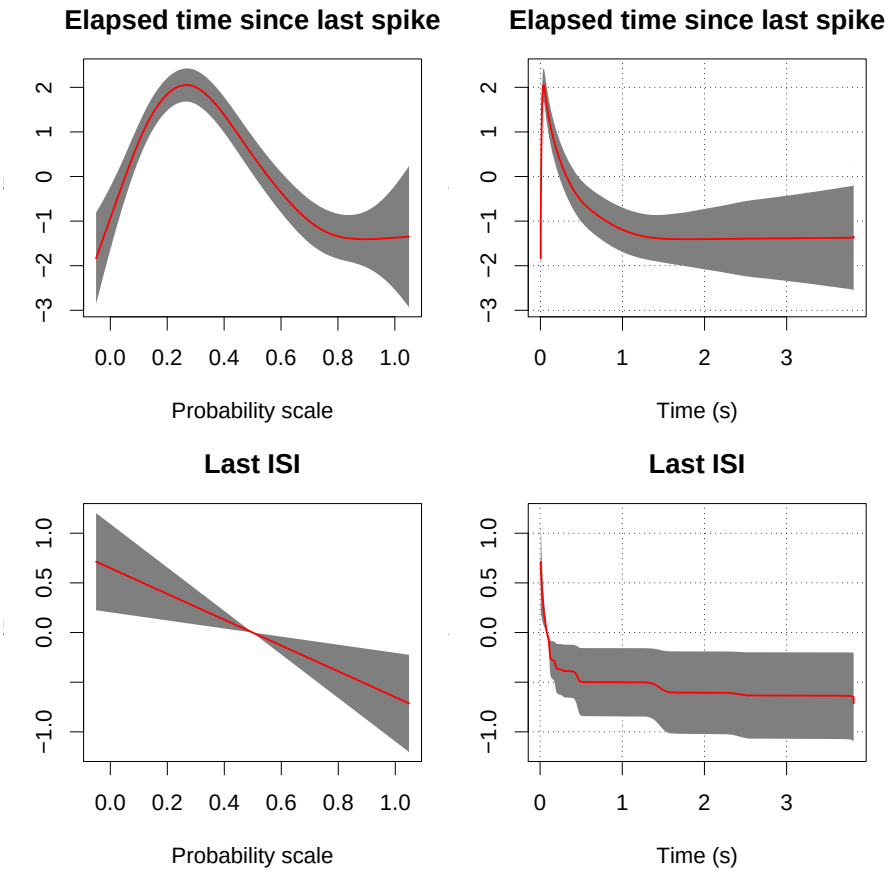

Figure 12: Terms `e1t` (upper row) and `i1t` (lower row) with a probability scale (left column) and with a native scale (right column).

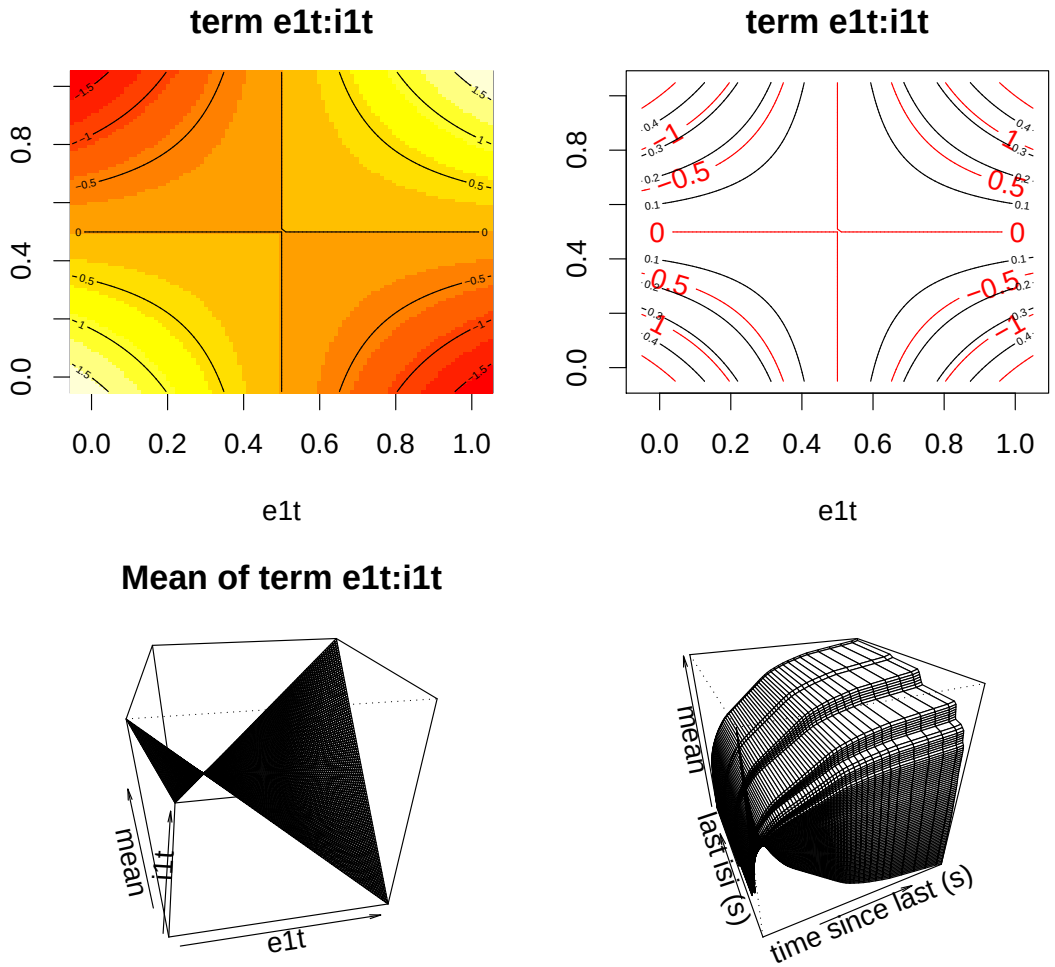

Figure 13: Examples of image (upper right), contour (upper row) and persp (bottom row) methods for quickPredict objects.

2.10.3 Looking at the *intensity process* of the two models

We conclude the analysis of the spike train of neuron 1 from the `e060824spont` data set by looking at the *intensity process* obtained with our two models (with and without interaction) on a small part of the spike train. We first get the predicted value of f_δ of Eq. (9) on the logit scale for the second half of the second half of the data set using the fit obtained from the first half:

```
> eta1.e <- predict(GF1e, newdata = subset(DFA, time > 29.5))
> eta2.e <- predict(GF2e, newdata = subset(DFA, time > 29.5))
```

We then convert `eta1.e` and `eta2.e` into proper frequencies:

```
> tigol <- function(x) exp(x)/(1 + exp(x))
> lambda1.e <- tigol(eta1.e)/0.004
> lambda2.e <- tigol(eta2.e)/0.004
```

Then a plot showing the *intensity process* of the two models is obtained with the following commands:

```
> with(subset(DFA, time > 29.5), plot(time, lambda1.e, xlim = c(30.5,
+     32), type = "l", col = 2, xlab = "Time (s)", ylab = expression(lambda ~
+     "(Hz)"), ylim = c(0, 50), lwd = 2))
> with(subset(DFA, time > 29.5), lines(time, lambda2.e, xlim = c(30.5,
+     32), col = 4, lty = 2, lwd = 2))
> with(subset(DFA, time > 29.5), rug(time[event == 1], lwd = 2))
> legend(30.5, 45, c("with interaction", "without interaction"),
+     col = c(2, 4), lty = c(1, 2), lwd = c(2, 2), bty = "n")
```

The results appears on Fig. 14. Notice that the *intensity process* of the "best model" (with interaction) is almost always larger than the one of the other model just before the spike while it tends to be smaller in between the spikes. In other words the best model predicts a lower event probability when there is actually no event and a larger probability when there are events.

2.11 Checking the necessity of variable transformations

See the long version of the vignette on the STAR web site[8].

3 Software versions used for this vignette

The versions of R and of the other packages used in this tutorial are obtained with function sessionInfo:

[8] http://sites.google.com/site/spiketrainanalysiswithr/

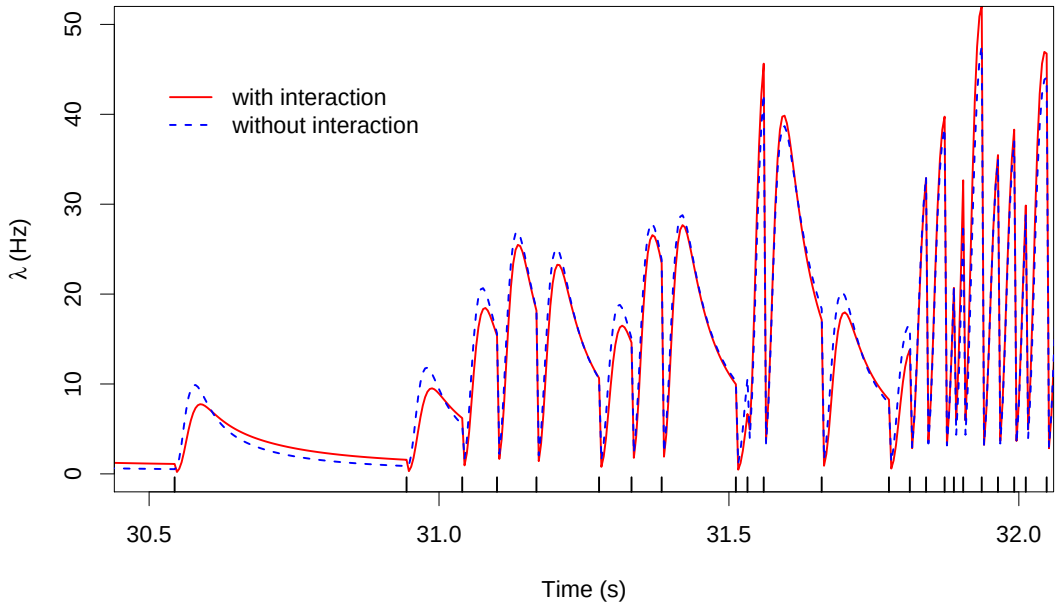

Figure 14: The *intensity process* of the two considered models, with (red, continuous) and without (blue, dashed) interactions between the elapsed time since the last spike and the last *isi*. The first half ($\leq$ 29.5 s) of the data set (`e060824spont`) was fitted with both models.

```
R version 2.9.2 (2009-08-24)
x86_64-unknown-linux-gnu

locale:
LC_CTYPE=fr_FR.UTF-8;LC_NUMERIC=C;LC_TIME=fr_FR.UTF-8;LC_COLLATE=fr_FR.UTF-8;LC_MONETARY=C;

attached base packages:
[1] splines   tools     stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
[1] STAR_0.3-2      gss_1.0-5       R2HTML_1.59-1   mgcv_1.5-5
[5] survival_2.35-4

loaded via a namespace (and not attached):
[1] grid_2.9.2      lattice_0.17-25 nlme_3.1-93
```

References

Per Kragh Andersen, Ørnulf Borgan, Richard D. Gill, and Niels Keiding. *Statistical Models Based on Counting Processes*. Springer Series in Statistics. Springer-Verlag, 1993. 2, 3

Mark Berman and T. Rolf Turner. Approximating Point Process Likelihoods with GLIM. *Applied Statistics*, 41(1):31–38, 1992. 9

D. R. Brillinger. Maximum likelihood analysis of spike trains of interacting nerve cells. *Biol Cybern*, 59(3):189–200, 1988a. 2, 3, 9

D. R. Brillinger, H. L. Bryant, and J. P. Segundo. Identification of synaptic interactions. *Biol Cybern*, 22(4):213–228, May 1976. 5

David R. Brillinger. Some Statistical Methods for Random Process Data from Seismology and Neurophysiology. *The Annals of Statistics*, 16(1):1–54, March 1988b. 9

E. S. Chornoboy, L. P. Schramm, and A. F. Karr. Maximum likelihood identification of neural point process systems. *Biol Cybern*, 59(4-5):265–275, 1988. 9

Chong Gu. *Smoothing Spline Anova Models*. Springer, 2002. 9

D.H. Johnson. Point process models of single-neuron discharges. *J. Computational Neuroscience*, 3(4):275–299, 1996. 2, 3

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. Available from: http://www.csiro.au/resources/Rcoursenotes.html. 8

Yosihiko Ogata. Statistical Models for Earthquake Occurrences and Residual Analysis for Point Processes. *Journal of the American Statistical Association*, 83(401):9–27, 1988. 2, 15

Christophe Pouzat and Antoine Chaffiol. Automatic Spike Train Analysis and Report Generation. An Implementation with R, R2HTML and STAR. *J Neurosci Methods*, in press, 2009. URL http://sites.google.com/site/spiketrainanalysiswithr/. Submitted manuscript. Pre-print distributed with the STAR package: http://cran.at.r-project.org/web/packages/STAR/index.html. 4, 5, 9

Christophe Pouzat and Antoine Chaffiol. On Goodness of Fit Tests For Models of Neuronal Spike Trains Considered as Counting Processes. URL http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/OnCPtests.html. 9, 15, 16, 30

Jonathan Touboul and Olivier Faugeras. The spikes trains probability distributions: A stochastic calculus approach. *Journal of Physiology, Paris*, 101(1–3):78–98, jun 2007. URL https://hal.inria.fr/inria-00156557. Preprint available at: https://hal.inria.fr/inria-00156557. 2

List of Figures

Index