Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Processing and Statistical Analysis of Single-Neuron Recordings

Doctoral Thesis

Mgr. Tomáš Sieger

Prague, December 2013

PhD. Study Programme: P2612 - Electrotechnics and Informatics
Branch of Study: 3902V035 - Artificial Intelligence and Biocybernetics

Supervisor: **Prof. RNDr. Olga Štěpánková, CSc.**
Supervisor-Specialist: **doc. MUDr. Robert Jech, PhD.**

Abstract

The human brain is one of the most fascinating and complex systems in the universe, consisting of an enormous number of 10^{11} neurons and 10^{14} synapses. While recently the experimental research of the living human brain has been mostly limited to studying only the overall activity of large neuronal populations, new opportunities to record the activity of individual neurons of the living brain rise, being enabled by medical and technological advances. However, these advances need to be reflected in methods allowing to process and analyse the recorded data. The aim of this thesis is to contribute to bridging the gap from single-neuron recordings to clinical relevance.

The thesis consists of two parts. First, it strives to contribute to the state of the art methods of processing and analysis of single-neuron recordings. An original software framework was designed and implemented enabling automated processing and access to data. To ease exploration of the data, which are multidimensional by nature, a freely available general-purpose interactive data exploratory tool was implemented. Also, an independent evaluation of existing methods that detect the activity of individual neurons in single-neuron recordings was performed, and statistical methods that can be employed to study the activity of individual neurons were reviewed.

Second, the methods described were applied on data recorded from Parkinson's disease patients treated with deep brain stimulation. The goal was to find answers to important neuroscientific questions regarding the functioning of the human brain in general, and the mechanism of the deep brain stimulation in particular. While we have been the first to report the finding of basal ganglia neurons being connected to the human oculomotor system, we have also found basal ganglia neurons related to emotional and visuo-attentional processing.

Besides methodological contributions and contributions to the general neuroscientific knowledge, the results reached would also help to shape the future treatment of neurodegenerative diseases.

Keywords: single-neuron recordings, spike sorting, Parkinson's disease, deep brain stimulation, basal ganglia, subthalamic nucleus, automated data processing, interactive data exploration, bootstrap, generalized linear models, spike train models.

Acknowledgments

I would like to thank my advisor Prof. Olga Štěpánková for her generous support during my study, all her valuable comments, and her endless willingness to provide a helpful hand in all situations.

I express my thanks to my colleagues from the Dept. of Neurology and Center of Clinical Neuroscience, Charles University in Prague, and Dept. of Stereotactic and Radiation Neurosurgery, Na Homolce Hospital, Prague, Czech Republic: my co-advisor Robert Jech, who helped to shape my work not only from the neuroscientific, but also from the methodological perspective, Tereza Serranová, who kindly and constantly kept shedding light on my rambling through the neuroscientific jungle, Filip Růžička, Daniela Šťastná, and Dušan Urgošík for providing patients' data, Evžen Růžička for his support, and Martin Voleman and Markéta Fialová for their technical and administrative assistence.

I'm very grateful to my colleagues at the Dept. of Cybernetics: Daniel Novák, Jiří Wild, Eduard Bakštein, and Pavel Vostatek for their help, ideas, suggestions, and fruitful discussions.

I would also like to express my admiration to the patients participating in the scientific studies described for their courage and willingness to provide their data.

Last but not least I thank my family for supporting me during my study.

This work was supported by by the Czech Technical University in Prague: grant project SGS10/279/OHK3/3T/13, by the Czech Science Foundation: grant project 309/09/1145, by the Czech Ministry of Education: research project MŠM 0021620849, by the Charles University in Prague: research project PRVOUK-P26/LF1/4, the Hennerův nadační fond, and by the Google Summer of Code 2012.

I'm grateful to John Chambers, Ross Ihaka and Robert Gentleman, the fathers of the **R programming language**, and the community of volunteers contributing to the language. I found R to be an excellent tool for data analysis and visualization.

This thesis was written using **Tufte-LaTeX**, LaTeX classes for producing handouts and books according to the style of Edward R. Tufte[1] and Richard Feynman[2]. I'm grateful to Richard Feynman, Edward R. Tufte, Donald E. Knuth, and the developers of Tufte-LaTeX for inspiring/creating such a clear presentation style.

[1] Tufte E. R. *Beautiful Evidence*. Graphics Press, LLC, first edition, May 2006. ISBN 0-9613921-7-7; Tufte E. R. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 2001. ISBN 0-9613921-4-2; Tufte E. R. *Envisioning Information*. Graphics Press, Cheshire, Connecticut, 1990. ISBN 0-9613921-1-8; and Tufte E. R. *Visual Explanations*. Graphics Press, Cheshire, Connecticut, 1997. ISBN 0-9613921-2-6

[2] Feynman R, Leighton R. B, and Sands M. L. *The Feynman Lectures on Physics*. Addison-Wesley, 1963. 3 volumes

Contents

List of Figures

List of Tables

List of Abbreviations

AMI	Adjusted Mutual Information
API	Application Programming Interface
CDF	Cumulative Distribution Function
DAO	Data Access Object
DBS	Deep Brain Stimulation
EM	Eye Movements
EOG	Electrooculography
GLM	Generalized Linear Model
GP	Globus Pallidus
GUI	Graphic User Interface
HCA	Hierarchical Cluster Analysis
IAPS	International Affective Picture System
IFR	Instantaneous Firing Rate
ISI	Inter-Spike Interval
KS	Kolmogorov-Smirnov (plot)
LM	Linear Model
MER	Microelectrode Recording
PD	Parkinson's Disease
Q-Q	Quantile-quantile (plot)
SNr	Substantia Nigra, Pars Reticulata
STN	Subthalamic Nucleus

1
Introduction

The human brain is one of the most complex systems in the universe. It consists of a staggering number of 10^{11} neurons, each of which makes connections to about 10^3 others, on average. While recently the experimental research of the living human brain has been mostly limited to studying the overall activity of large neuronal populations, new opportunities to record the activity of individual neurons of the living brain rise, being enabled by medical and technological advances. However, these advances need to be reflected in methods allowing to process and analyse the recorded data. The aim of this thesis is to contribute to **bridging the gap from single-neuron recordings to clinical relevance**.

The aim of this thesis is twofold. First, it strives to contribute to the state of the art methods of processing and analysis of single-neuron recordings. Second, it applies the methods to data recorded from Parkinson's disease patients treated with deep brain stimulation in order to contribute to the understanding of the human brain in general, and the mechanism of the deep brain stimulation in particular.

This thesis spans three scientific disciplines: **applied statistics**, **computer science**, and **neuroscience**[1]. While applied statistics is fundamental to look at the data the right way - to focus on important aspects of the data and to make correct inference, computer science is needed to design and implement analytical and processing tools operating accurately and effectively. Finally, neuroscience sets the scene and specifies a set of challenging questions to answer.

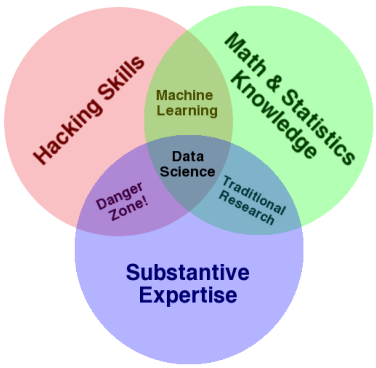

Figure 1.1: Data science as the intersection of computer science, applied statistics and domain knowledge. While traditional research applies statistics to small data only, applying hacking skills only to write computer programs dealing with data can be dangerous, as incorrect inference can be made ignoring the uncertainty in the data. Finally, combining hacking skills with statistical methods is valuable, but yet needs some data to be applied to. *(Adapted after Drew Conway.)*

[1] the joint use of applied statistics and computer programming in solving real-world problems is referred to as **data science**, and described in Fig. 1.1.

1.1 Goals of the Thesis

The methodological goals, spanning data processing and analysis, are:

- to devise and implement a means of **automated processing and access to** relatively large amount of **data** enabling to analyse the data in a straightforward and error-free way, in which the analytical applications would focus purely on the data of their interest, not on technical details as how to access or transform the data,
- to **explore the high-dimensional data** of extracellular single-neuron recordings, which can't be handled by classic tools,
- **to transform** raw data consisting of extracellular **single-neuron recordings** into directly analyzable traces of activity of individual neurons.

The neuroscientific questions aim at elucidating the role of the basal ganglia[2] in the human brain, with focus on understanding the function of the basal ganglia in Parkinson's disease[3] patients treated with deep brain stimulation[4]. New insights gained would help not only to optimize the treatment of neurodegenerative diseases, but to contribute to the general understanding of the human brain. In particular, the goals are:

[2] to learn what the basal ganglia are, see Section 2.3

[3] see Section 2.3

[4] see Section 2.4

- to identify basal ganglia neurons related to planning, execution and control of **eye movements**,
- to identify basal ganglia neurons related to processing or responding to distinct types of **emotional information**,
- to identify basal ganglia neurons activated during specific **visuo-attentional tasks**.

1.2 Structure of the Thesis

This thesis is structured as follows. **Chapter 2** gives a brief **neuroscientific background**. Section 2.1 describes the neuron, an elementary information-processing unit of our interest, and Section 2.2 introduces the means of recording the activity of individual neurons, yielding single-neuron recordings I further process and analyse. Section 2.3 characterizes the Parkinson's disease, and Section 2.4 depicts the deep brain stimulation as the treatment of advanced Parkinson's disease.

Chapter 3 includes **the state of the art**. In Section 3.1 I discuss the process of transforming single-neuron recordings into traces of activity of individual neurons. Section 3.2 presents the bootstrap, a general technique enabling to assess statistical uncertainty and to test

statistical hypotheses. Section 3.3 describes linear models as valuable general-purpose statistical models and the foundation of generalized linear models presented later in Section 3.4.

Chapter 4 comprises of the **original methodological results** - techniques developed to access, process, explore, and analyse data of our interest. Section 4.1 covers the analysis and design of a software component enabling automated processing and access to data. In Section 4.2 I present an original data exploration software tool that allows to study the internal structure of data, and demonstrate the use of the tool in exploration of single-neuron recordings. Section 4.3 provides a qualitative comparison of tools that transform single-neuron recordings into traces of activity of individual neurons.

Chapter 5 presents the **answers to the neuroscientific questions** raised, utilizing the methods developed in Chapter 4. Focusing on the basal ganglia, Section 5.1 describes the finding of neurons whose activity was related to eye movements. Section 5.1 then summarizes the finding of emotion-related neurons of the subthalamic nucleus, and Section 5.3 depicts subthalamic neurons whose activity was specific to visuo-attentional tasks performed.

Finally, **Chapter 6 concludes** the thesis and highlights the achievements of the thesis.

2
Problem Context

All data used in this thesis originate from recordings of activity of individual neurons in Parkinson's disease patients treated with deep brain stimulation. I open the thesis by introducing the neuroscientific background of my work. I briefly describe neurons and their functions in Section 2.1, explain the way of recording the activity of individual neurons in Section 2.2, depict Parkinson's disease in Section 2.3, and illustrate the neurosurgical treatment of neurodegenerative diseases using deep brain stimulation in Section 2.4.

2.1 *The Neuron*

The *neuron* is a cell that processes and transmits information. A typical neuron has three components (Fig. 2.1): the body, dendrites, and an axon.

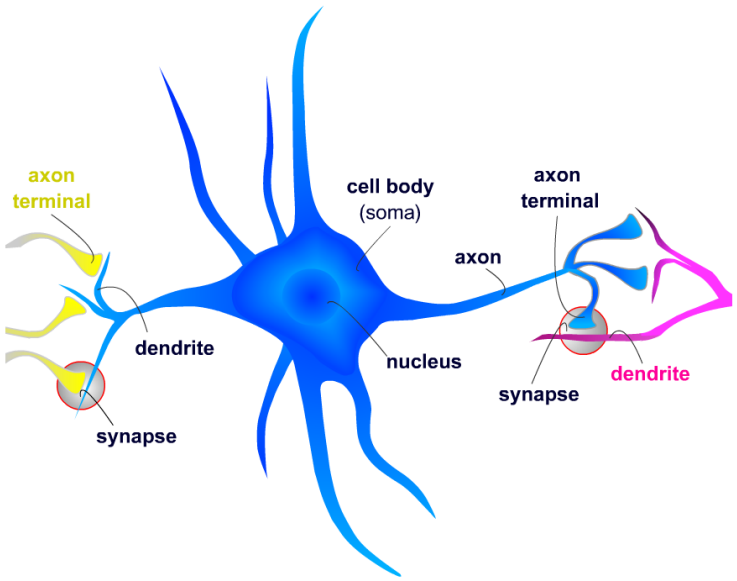

Figure 2.1: Example of a neuron (in blue), consisting of body surrounded by dendrites, and an axon with axon terminals that connect to another neuron (purple) at synapses. *(Adapted after The Mind Project.)*

The body takes part in control and metabolism of the cell. The *dendrites* pose places through which input signals from other neurons are typically received. The dendrites form a dendritic tree, enabling the neuron to receive signals from many other neurons. The *axon* extends from the body and transmits signals to other neurons. There is at most[1] one axon in each neuron. In contrast to relatively short dendrites, axons can traverse large distances (over a meter in humans, and even further in other species).

dendrites

axon

[1] there are neurons that have no axons and transmit information through their dendrites

Signals typically flow from one neuron to another through *synapses* located on dendrites. The number of synaptic inputs received by each human neuron varies from 1 to about 100,000. Synapses are of two types: chemical and electrical (Fig. 2.2). *Chemical synapses* are formed by *axon terminals* connected to dendrites of another neuron. The axon terminals contain chemical compounds called neurotransmitters. When a signal arrives in the presynaptic axon terminal, the neurotransmitter gets released from the axon terminal and diffuses to the *synaptic cleft* between the axon terminal and the postsynaptic neuron and binds to specific receptor proteins causing generation of electrical or chemical signals (e.g. the neurotransmitter opens ion channels and the influx of ions causes the change in the electrical potential of the postsynaptic neuron). *Electrical synapses* are formed by physical connections of two neurons to each other. The connection permits changes in the electrical potential of one neuron to effect the other, and vice versa. Unlike chemical synapses, electrical synapses have no *gain*, i.e. they can pass the same (or smaller) electrical potential to the target neuron (by ion channel opening), but cannot amplify the signal. In chemical synapses, however, a relatively small electrical signal releasing the neurotransmitter can result in much larger electrical potential generated in the target neuron.

synapses

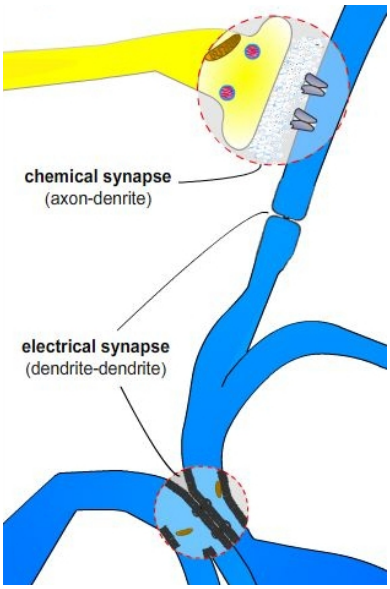

Figure 2.2: Chemical and electrical synapses connecting different neurons. Axon (in yellow) connects to a dendrite of another neuron using a chemical synapse (top). Dendrites (in blue) of three different neurons connect to each other using electrical synapses. *(Adapted after The Mind Project.)*

Under resting condition, the electrical potential of a neuron (the potential inside the cell relative to that of the surrounding environment) is -70 mV. The neuron is polarized. The influx of positive charge into the neuron makes the inside of the neuron more positive (hence, *depolarized*). When this depolarization reaches a threshold, a large electrical signal called an *action potential (or spike)* is generated (Fig. 2.3). The action potential, which is a roughly 100 mV fluctuation in the electrical potential across the cell membrane and lasts for about 1 ms, travels rapidly[2] along the axon, and propagates to other neurons through synaptic connections. At chemical synapses, it results in the release of a neurotransmitter, at electrical synapses, the output is the electrical signal itself.

action potential (spike)

[2] at rates up to 150 meters per second

Action potential generation depends on whether the neuron has fired recently. For one or a few milliseconds just after an action potential has been fired, it may be virtually impossible to fire another spike

refractory period

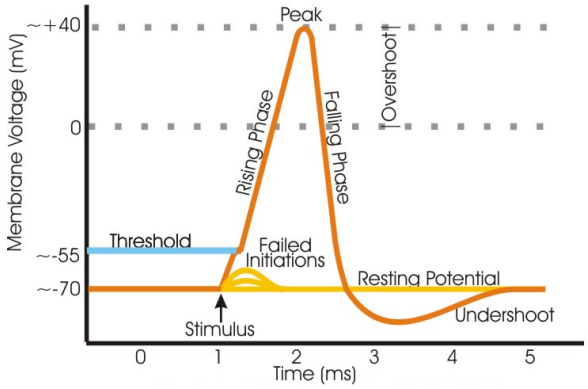

Figure 2.3: A schematic view of an action potential. *(Figure reproduced from Wikimedia Commons.)*

for biophysical reasons. This is called the *absolute refractory period*. For a longer interval of up to tens of milliseconds after spike generation, the probability of firing another action potential is lowered. This is called the *relative refractory period*.

Note that the output of a neuron is discrete: the neuron either fires an action potential, or it does not. The action potential is always the same: different stimuli cannot produce different action potentials. However, stimuli of greater intensity can (but do not necessarily need to) produce a higher frequency of firing. The frequency, timing, and the pattern of spikes constitues the *neural code* used by neurons to transfer information from one location to another. What neural code do neurons use is a topic of intense debate (for reference, see e.g. Section 1.5 in Dayan and Abbott (2001)[3]). In reality, ruling out and ruling in neural codes is not easy, but possible both theoretically[4] and practically[5].

neural code

Neurons are of several types. Sensory neurons respond to changes in the inner and outer environment, motor neurons mediate muscle contractions, and interneurons connect neurons to other neurons within the same region.

Neurons connect to each other to form neural networks. There are about 100 billion (10^{11}) neurons and 100 trillion (10^{14}) synapses in the human brain.

Detailed facts about neurons can be found e.g. in Purves (2008)[6]. A simplified, but visually appealing explanation can be found at The Mind Project [7].

[3] Dayan P and Abbott L. F. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 12-01 2001. ISBN 0262041995

[4] Simoncelli E. P and Olshausen B. A. Natural image statistics and neural representation. *Annu. Rev. Neurosci.*, 24: 1193–1216, 2001

[5] Jacobs A. L, Fridman G, Douglas R. M, Alam N. M, Latham P. E, Prusky G. T, and Nirenberg S. Ruling out and ruling in neural codes. *Proc. Natl. Acad. Sci. U.S.A.*, 106(14):5936–5941, Apr 2009

[6] Purves D. *Neuroscience*. Sinauer, 4th edition, July 2008. ISBN 9780878936977

[7] The Mind Project . URL `http://www.mind.ilstu.edu/`

2.2 *Recording Neuronal Activity*

Neuronal activity can be recorded at different scales using many different techniques. For instance, at the macroscopic scale of large neuronal populations, the direct method of electroencephalography as well as

the indirect methods of functional magnetic resonance imaging and positron emission tomography can be used. At the scale of individual neurons, there are basically two ways how to record the neuronal activity: intracellular and extracellular recordings. *Intracellular recordings* are made by electrodes inserted into a neuron, such that the phenomena internal to the neuron can be seen in these recordings, such as subthreshold membrane potentials[8]. Intracellular recordings are typically used for in vitro experiments on slices of neural tissue. In contrast, electrodes located in the bath surrounding neurons can give rise to *extracellular recordings*. Such recordings cannot capture the phenomena internal to the neuron, such as membrane potentials that do not reach the threshold to fire a spike, but only action potentials fired by the neuron. Extracellular recordings are typically used for in vivo experiments in behaving animals and I will focus on those solely in this text.

intracellular recordings

[8] the "Failed Initiations" in Fig. 2.3

extracellular recordings

An electrode inserted into a neural tissue will detect the electrical activity generated by the neurons adjacent to the tip of the electrode. Depending on the size of the electrode, the activity of a small or large number of neurons can be detected on the electrode.

On a microelectrode with a tip size of the order of 1 μm the activity of one or a few neurons will be detected - the microelectrode will "listen" only to the activity in the area close to its tip, where none, one, or a few neurons can be situated (Fig. 2.4). Recordings of this type are called "single-unit" or "single-neuron" recordings (Fig. 3.1). As such single-neuron recordings contain the aggregated activity of possibly more than just one neuron, it is usually desirable to restore the activity of individual neurons from the recordings. This process is called *spike detection and sorting* and will be discussed in Section 3.1 in detail.

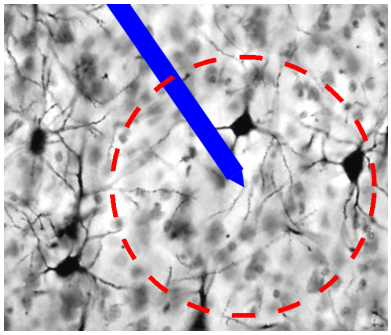

Figure 2.4: An illustration of single-neuron recording. While there are five neurons (black) visible in the figure, the electrode (blue) can effectively record the activity just from the two neurons that are close to the tip of the electrode (red dashed circle).

If the electrode is larger, it can no longer distinguish the activity of individual neurons, but it will record the *local field potential* generated by the activity of a large number of cells close to the electrode.

2.3 *Parkinson's Disease*

Parkinson's disease (PD) is the second most common[9] neurodegenerative disease, first described by an English apothecary James Parkinson[10] in 1817. Besides motor symptoms, which include movement slowness, muscle rigidity, and resting tremor, PD patients manifest also cognitive, affective, sleep, and autonomous disturbances (Fig. 2.5).

The impairment in motor function is due to the progressive loss of dopaminergic neurons in the substantia nigra, which affects the function of the *basal ganglia*, a group of nuclei including the caudate nucleus, the putamen, the globus pallidus, the substantia nigra, and the

[9] after Alzheimer's disease

[10] Parkinson J. An Essay on the Shaking Palsy. *J Neuropsychiatry Clin Neurosci*, 14(2):223–236, May 2002. DOI: 10.1176/appi.neuropsych.14.2.223

Figure 2.5: Front and side views of a man portrayed to be suffering from Parkinson's disease. *(Reproduced from Wikipedia.org.)*

subthalamic nucleus (STN). The dysfucntion of the basal ganglia, in turn, affects cortical functions and leads to the symptoms described.

The etiology of the progressive deterioration of the dopaminergic neurons is mostly unknown. However, in rare cases, genetic investigations are providing clues to the etiology and pathogenesis of PD.

At present, PD can be treated symptomatically, but not cured. While gene therapy and approaches using stem cell seem to be promising[11], in practice, the most widely used treatment of PD is dopaminergic substitution (such as L-DOPA medication). In some people, however, such a treatment is not effective or cannot be used due to its side effects. Thus, surgical means of treatment can become helpful, including the deep brain stimulation of the STN.

[11] Purves D. *Neuroscience*. Sinauer, 4th edition, July 2008. ISBN 9780878936977

2.4 *Deep Brain Stimulation*

Deep brain stimulation (DBS) is a surgical treatment of diseases of the central nervous system, which include mainly movement disorders (such as Parkinson's disease, tremor, and dystonia), chronic pain, and more recently also some psychiatric disorders.

DBS involves the implantation of a neurostimulator that sends electrical pulses into electrodes inserted into specific parts of the brain (Fig. 2.6).

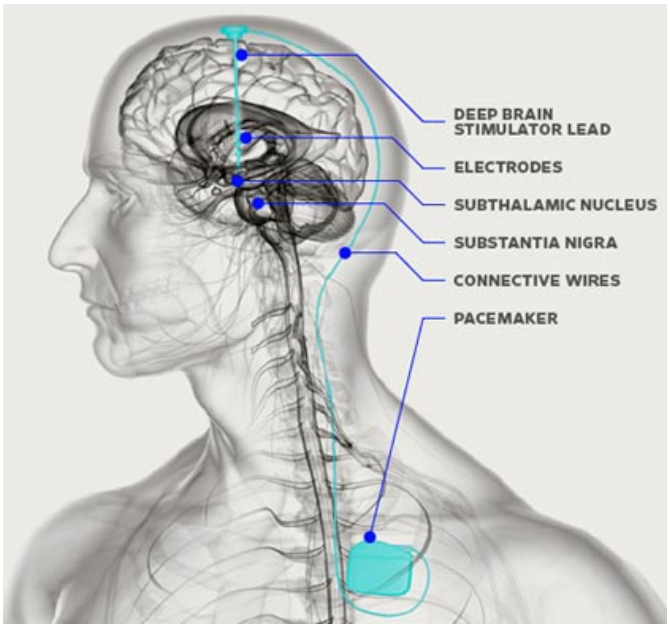

Figure 2.6: Illustration of deep brain stimulation.

DBS has been shown to provide remarkable therapeutic benefits for otherwise treatment-resistant longstanding conditions. Compared to irreversible lesion-based therapy, the major advantage of the DBS is

that it alters the brain activity in a controlled manner and its effects are mostly reversible. For example, if new treatment strategy becomes available, the DBS can be turned off (or the neurostimulator can even be explanted), leaving the target brain structure almost in the state prior the implantation of the DBS electrodes. On the other hand, the DBS comes with the disadvantage of being more expensive and requiring maintenance.

It shall also be noted that the DBS can be associated with negative side effects[12]. This is mainly because the principles and mechanisms underlying the DBS are still unclear. Therefore, one of the aims of this thesis is to shed some light on the the mechanism of the STN DBS by gaining some evidence of what is the function of neurons being stimulated by the STN DBS. Indeed, as we found both eye movement-related[13] and emotion-related[14] neurons in the STN, we can suggest that the observed phenomena of altered eye movements (e.g. paresis or diplopia) or emotion-related side effects of the STN DBS (e.g. emotional lability, depression) are caused by the stimulation of the neurons of the STN.

[12] Wolters E. C. h. Deep brain stimulation and continuous dopaminergic stimulation in advanced Parkinson's disease. *Parkinsonism Relat. Disord.*, 13 Suppl:18–23, Sep 2007

[13] see Section 5.1

[14] see Section 5.2

3
State of the Art

In this chapter I summarize the methods used to preprocess and analyse microelectrode recordings and data based on them. In Section 3.1, I present means of converting single-neuron recordings into traces of activity of individual neurons. Section 3.2 introduces bootstrap, a statistical technique to assess uncertainty. Finally, I describe several statistical models: linear models in Section 3.3, generalized linear models in Section 3.4, and models of spiking data in Section 3.5.

3.1 Spike Detection and Sorting

Single-neuron microelectrode recordings (MER)[1] consist of traces of the electrical potential in the tissue surrounding the recording microelectrode (Fig. 3.1). There could be none, one or a few neurons located in the tissue close to the microelectrode (Fig. 2.4), such that the MER could contain traces of none, one, or several neurons. Therefore, in order to study not the aggregated activity of the neuronal ensemble, but the behaviour of individual neurons, we need to untangle the aggregated neuronal activity and obtain the activity of the individual neurons in the ensemble.

[1] see Section 2.2

The discrete nature of neuronal activity, i.e. the fact that at any given time point each neuron either fires an action potential[2] or stays silent, reduces the problem significantly. Thus, all we need is to *detect* action potentials (spikes) in a single-neuron recording, and *sort* them according to what neuron emitted them. This process is called *spike detection and sorting*. It results in a sequence of spikes, called *the spike train*, for each neuron detected in the single-neuron recording.

[2] often called *the spike* in technical literature

spike detection and sorting

spike train

There have been many methods designed to detect and sort spikes. While some authors described the spike-emitting system of neurons using a probabilistic generative model and tried to restore the parameters of that model[3], others proceeded literally in two steps by first detecting spikes, and sorting them subsequently[4]. Different approaches to spike detection and sorting can also be taken according

[3] Herbst et al. (2008); Sahani (1999)

[4] Quiroga et al. (2004); Rutishauser et al. (2006); Harris et al. (2000); Takahashi et al. (2003b,a)

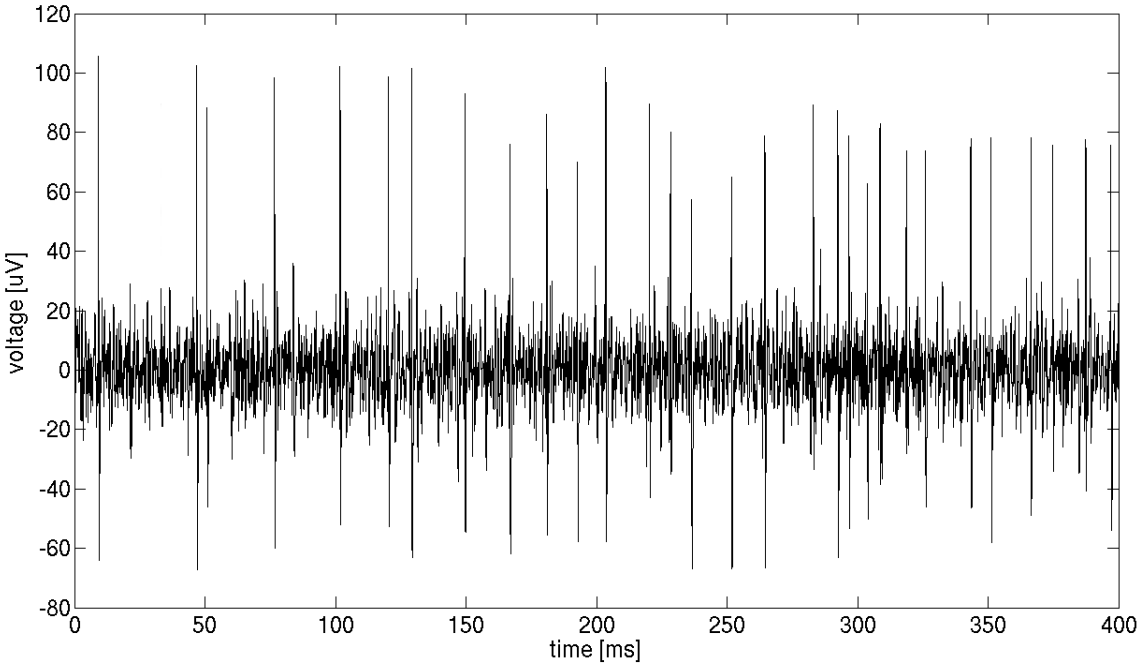

Figure 3.1: Single-neuron microelectrode recording expressed as a voltage trace, in which both action potentials (spikes, sharp peaks in the signal) and lower-amplitude noise background activity are present.

to whether they deal with *single-channel recordings*, in which the aggregated activity of the neuronal ensemble is recorded by a single microelectrode[5], or with *multi-channel recordings*, in which the activity of the neuronal ensemble is recorded by multiple microelectrodes located close to each other[6]. Naturally, spike trains can be isolated more accurately from multi-channel recordings, as there is much more information available: even though the activity recorded by all the microelectrodes refers to the very same neuronal ensemble, the differences in the relative positions and orientations of the individual neurons in respect to the recording microelectrodes make the contribution of each neuron unique[7], simplifying the identification of the spikes they emit.

In the following text, I restrict my attention to two-step single-channel spike sorting methods, because our recordings are single-channel, and all the spike sorting methods readily available follow the two-step approach.

Spike detection and sorting methods can be illustrated as follows. The input signal consisting of the aggregated activity of the neuronal ensemble surrounding the tip of the recording microelectrode is shown in Fig. 3.1. First, the signal gets thresholded (Fig. 3.2, top), and spikes exceeding the threshold get extracted from the signal (Fig. 3.2, bottom left). Then, the detected spikes get sorted into several classes of putative neurons according to the shape of the spikes they had produced (Fig. 3.2, bottom right). In the example given, there were spikes of two

[5] Quiroga et al. (2004); Rutishauser et al. (2006); Herbst et al. (2008)

[6] Takahashi et al. (2003b,a)

[7] Gold C, Henze D. A, Koch C, and Buzsaki G. On the origin of the extracellular action potential waveform: A modeling study. *Journal of neurophysiology*, 95 (5):3113–3128, May 2006

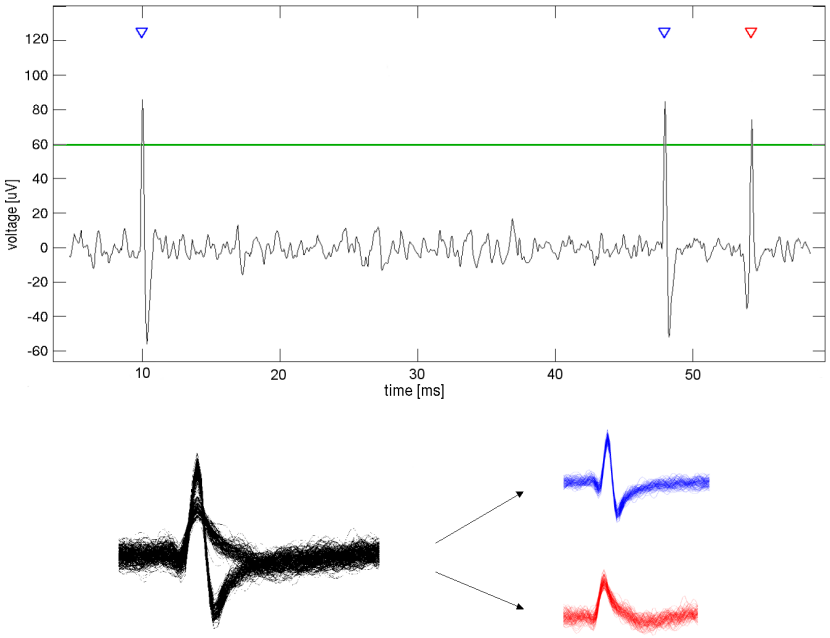

Figure 3.2: Detection and sorting of spikes in single-neuron microelectrode recording. **Top**: illustration of three spikes exceeding the threshold (green horizontal line) in a portion of microelectrode recording (MER). **Bottom**: spikes exceeding the threshold in the whole MER signal (black) were sorted according to their shape into two groups representing two putative neurons (blue and red).

neurons detected in the MER and sorted correctly.

amplitude spike detection

To detect spikes, I employed the *amplitude spike detection*[8] that recognized spikes once the absolute value of their amplitude reached a given threshold computed as

$$threshold = C \cdot \hat{\sigma}_{noise}, \qquad \hat{\sigma}_{noise} = median\left(\frac{|MER_{BF}|}{.6745}\right) \tag{3.1}$$

where C was a user-supplied parameter set to 3, MER_{BF} was the single-neuron MER signal band-pass-filtered in the range from 300 to 3000 Hz, and $\hat{\sigma}_{noise}$ was the estimate of the standard deviation of the background noise.

[8] Quiroga R. Q, Nadasdy Z, and Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004

In general, there is no single optimal spike sorting method applicable to any MER data, as the performance of spike sorting methods can depend on many factors (the number of neurons present in the MER, the signal-to-noise ratio, the shape of spikes, etc.) Thus, our task was to pick the optimal spike sorting method to be applied to our single-neuron MER data, and to choose the optimal parameters for the method selected. The evaluation of spike sorting methods is presented in Section 4.3.

evaluation of spike sorting methods

In addition, I developed an exploratory tool capable of interactive inspection of spikes in single-neuron recordings. The tool can also be used to assess the validity of the result of a spike sorting algorithm. The tool is described in Section 4.2, including a case study demonstrating how the tool can be used to explore spikes data (Section 4.2.7).

exploration of spikes

3.2 *The Bootstrap*

In applied statistic, we often cope with realizations of random variables following distributions that are not fully known, but which are of scientific interest (typically, we need to learn some function of the distribution, such as its mean, variance, quantile ranges, etc.). If we knew the parametric form of the distribution (e.g. normal, or Student t-distribution), all we needed would be to estimate the parameters of the distribution, which would fully determine the distribution. However, what can we do if we do not know the distribution at all? This section provides a solution to this problem.

3.2.1 *Problem Formulation*

Formally, the problem is as follows. Given a realization $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ of random variable $\boldsymbol{X} = (X_1, X_2, ..., X_n)$[9] drawn from some unknown probability distribution F, and some random variable $T = T(\boldsymbol{X})$ depending on the random variables $\boldsymbol{X}$, we want to estimate the *sampling distribution* of T, i.e. the distribution of estimates of the true value of T.

[9] I follow the practice of using small letters to represent observed data, and capital letters to represent random variables

3.2.2 *The Bootstrap Principle*

The solution to the aforementioned problem, as proposed by Efron[10], is surprisingly simple (at least from the theoretical point of view). The core idea here is to simulate replications needed to assess uncertainty.

[10] Efron B. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, January 1979

We proceed in three steps. First, we aproximate the unknown probability distribution F by constructing the sample distribution $\hat{F}$ by putting mass $1/n$ to each of $x_1, x_2, ..., x_n$. Subsequently, we draw random samples of size n (with replacement) from $\hat{F}$, obtaining so-called *bootstrap samples* $\tilde{\boldsymbol{x}}$ of random variables $\tilde{\boldsymbol{X}}$:

simulated replications

$$\tilde{X}_i \sim \hat{F}, \quad \tilde{x}_i = \tilde{X}_i \tag{3.2}$$

Finally, we approximate the sampling distribution of $T(\boldsymbol{X})$ by the bootstrap distribution of

$$\tilde{T} = T(\tilde{\boldsymbol{X}}) \tag{3.3}$$

If $F = \hat{F}$, then the bootstrap distribution of $\tilde{T}$ equals the desired sampling distribution of T.

Once we have obtained the estimate of the sampling distribution of T, we can assess the accuracy of our (point) estimate $\hat{t} = T(\boldsymbol{x})$[11]. For example, we can estimate the variance of the estimate $\hat{t}$ by

[11] Shalizi C. R. *Advanced Data Analysis from an Elementary Point of View*. 2012. URL `http://www.stat.cmu.edu/~cshalizi/uADA/12/`

$$\widehat{var}(\hat{t}) = var(\tilde{t}) \tag{3.4}$$

assuming that the simulated bootstrap samples $\tilde{\boldsymbol{x}}$ have about the same distribution as the real data $\boldsymbol{x}$, such that computing $T(\tilde{\boldsymbol{X}})$ over the simulated data gives the sampling distribution of $T(\boldsymbol{X})$.

It should be noted that the procedure outlined above is sometimes referred to as the *nonparametric bootstrap*, as the bootstrap samples are generated directly from the original observations. In contrast, in *parametric bootstrap*, there is a (parametric) model fitted on the observations and bootstrap samples are drawn from that model[12].

[12] we will see an example of parametric bootstrap in Section 5.3

3.2.3 *Simulating Bootstrap Samples*

So far, I have described the bootstrap principle from the theoretical point of view, but have not specified how to simulate the bootstrap samples. There are at least three methods to do so[13]:

[13] Efron B. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, January 1979

1. Direct theoretical derivation and calculation.
2. Monte-Carlo aproximation to the sampling distribution. This aproximation consists of repeated sampling of B bootstrap samples using computers. The accuracy of such aproximation depends on the number of simulated samples B.
3. Taylor series expansions can be used to aproximate the mean and variance of the sampling distribution $\tilde{T}$.

In the following text, I will focus on Monte-Carlo simulations, as they are widely applicable in real-world settings (in which theoretical derivations can not be conducted) and yield accurate results, when used properly.

3.2.4 *Hypothesis Testing*

The bootstrap is a powerful technique enabling testing complex statistical hypotheses, for which the sampling distributions of their corresponding test statistics are unknown.

Consider a test based on some test statistic $T(\boldsymbol{X})$, observed data $\boldsymbol{x}$, and considering some null and alternative hypotheses. The p-value of such test is the probability that the test statistic $T(\boldsymbol{X})$ computed under the null is equally or more extreme compared to the test statistic $T(\boldsymbol{x})$ computed from the observed data. To compute p-value using the bootstrap principle, we perform the following steps:

p-value definition

bootstrap p-value

1. compute the test statistic $t = T(\boldsymbol{x})$ from the observed data
2. under the null hypothesis, simulate B bootstrap samples $\tilde{\boldsymbol{x}}_i$ and compute the test statistic $\tilde{t}_i = T(\tilde{\boldsymbol{x}}_i)$ from each of them; $\hat{D}$, the distribution of the simulated $\tilde{t}_i$ statistics approximates the sampling distribution of $T(\boldsymbol{X})$

3. calculate the p-value

P-value can be calculated by determining the percentage of bootstrap samples for which the corresponding test statistic $\tilde{t}_i$ is equally or more extreme compared to the test statistic t computed from the observed data:

$$p = \frac{\#\{T(\tilde{\boldsymbol{x}}_i) \geq T(\boldsymbol{x})\}}{B}, \tag{3.5}$$

where $\#\{condition\}$ is the number of cases for which the *condition* is satisfied.

However, the p-value calculated by formula 3.5 comes with the disadvantage of possibly being equal to zero. It is the case if none of the simulated samples gives rise to test statistic $T(\tilde{\boldsymbol{x}}_i)$ equally or more extreme compared to the test statistic $T(\boldsymbol{x})$ computed from the observed data, which can happen for two reasons: either, $T(\boldsymbol{x})$ can be too far from the sampling distribution of the $T(\boldsymbol{X})$ simulated under the null hypothesis, or the number of samples B is too low, such that no simulated sample $\tilde{\boldsymbol{x}}_i$ gave rise to an extreme value of $T(\tilde{\boldsymbol{x}}_i)$ by chance alone. In the latter case, the p-value of zero clearly represents a sampling artifact.

The p-value of zero poses a problem, as it can give a false impression of extreme evidence against the null, while, in fact, the p-value perhaps represents only a sampling artifact. For example, consider the p-value of 0 is subject to multiple comparison correction[14], in which the p-value gets multiplied by some correction factor. A p-value of 0 can never increase being subjected to such a correction, such that the p-value can never become insignificant, i.e., it can represent an uncorrectable false positive finding.

[14] Benjamini Y and Hochberg Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57 (1):289–300, 1995. ISSN 00359246. DOI: 10.2307/2346101

p-value formula

The recommended way of p-value calculation is thus the following formula:

$$p = \frac{\#\{T(\tilde{\boldsymbol{x}}_i) \geq T(\boldsymbol{x})\} + 1}{B + 1} \tag{3.6}$$

in which the p-value can never be smaller than $1/(B+1)$. This lower bound guarantees that none of the issues raised above exists any more: no sampling artifact can occur (because small B results in inaccurate, but a large p-value), and p-values can safely be subject to multiple comparison correction. The idea behind formula 3.6 is that the observed data $\boldsymbol{x}$ should be considered to be one of the samples observable under the null hypothesis, which makes the p-value slightly more conservative[15] compared to formula 3.5. However, the conservativeness of formula 3.5 is not a big issue, as both the formulas result in the same p-value when B approaches infinity, and we can set the value of B to any value, namely a value for which both the formulas 3.5 and 3.6 yield p-values that are equal for practical purposes.

[15] i.e. favoring the null hypothesis

An example R[16] code computing bootstrap p-values follows:

[16] R Core Team . *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL http://www.R-project.org/. ISBN 3-900051-07-0

```
# x .. data sample
# T .. function computing the test statistic
# B .. number of bootstrap samples to draw

n <- length(x)
t <- T(x) # the observed test statistic

tbs <- rep(NA, B) # create a vector of 'B' bootstrap statistics
for (i in 1:B) { # repeat 'B' times
    xb <- sample(x, n, replace = TRUE) # bootstrap resampling
    tbs[i] <- T(xb) # compute the i-th bootstrap test statistic
}

# p-value of a two-sided test:
p.value.two.sided <- (1 + sum(abs(tbs) >= t)) / (B + 1)

# p-value of one-sided test (assuming 't' is greater than zero):
p.value.one.sided <- (1 + sum(tbs >= t)) / (B + 1)
```

In practice, the choice of B can be made according to the purpose of the bootstrap test. Roughly speaking, if a single test is to be performed at the significance level of 0.05, a value of $B = 999$ would suffice (resulting in the smallest possible p-value of 0.001). If, however, multiple tests are to be performed, subjects to multiple comparison corrections, much larger values of B may be needed to achieve higher p-value accuracy. In all cases, the formula 3.4 can help to estimate the value of B to meet some prespecified accuracy of the test statistic estimates.

3.3 *Linear Models*

In this section I briefly describe a very powerful technique to model data: (general) linear models, which can be used to model Gaussian random variables (conditionally on a linear combination of some covariates).

Care must be taken not to confuse (general) linear models with generaliz**ed** linear models, which will be described in Section 3.4.

The theory of linear models can be found in almost any statistical textbook, see e.g. Faraway (2002)[17], which takes a practical view in the context of R[18], a language and environment for statistical computing. In Czech, there is an excellent book[19] written by Zvára.

[17] Faraway J. J. *Practical Regression and Anova using R*. 2002

[18] R Core Team . *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL `http://www.R-project.org/`. ISBN 3-900051-07-0

[19] Zvára K. *Regrese*. Martfyzpress, 2008

3.3.1 *Definition*

In brief, given a vector $\boldsymbol{Y}$ of n responses[20] Y_i, which represent measurements on n independent units, and a full rank matrix of nxp (possibly random) regressors[21] X_{ij} ($\boldsymbol{X_i}$ in matrix notation), the *linear model* explains the expected value of the response Y_i (conditionally on the regressors, if they are random) in terms of a linear combination of the regressors:

$$E(Y_i|\boldsymbol{X_i}) = (\boldsymbol{X_i})^T\boldsymbol{\beta}, \tag{3.7}$$

linear model

[20] called also outputs, or dependent variables

[21] called also predictors, covariates, or exploratory or independent variables

where $\boldsymbol{\beta}$ is a vector of p unknown parameters to be estimated from the data.

The i-th response can be written as:

$$Y_i = (\boldsymbol{X_i})^T\boldsymbol{\beta} + \epsilon_i, \tag{3.8}$$

where $\epsilon_1, ..., \epsilon_n$ are error terms[22], which are considered to be conditionally independent (given the regressors), centered at zero and homoscedastic:

[22] also called noise

$$E(\epsilon_i|\boldsymbol{X_i}) = 0 \tag{3.9}$$

$$var(\epsilon_i|\boldsymbol{X_i}) = \sigma^2. \tag{3.10}$$

Under these assumptions, it can be shown that the unbiased estimate of the $\boldsymbol{\beta}$ parameter is

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}. \tag{3.11}$$

If, in addition, we assume normality of the error terms (which is equivalent to normality of the reponse given the regressors), i.e., if we assume

$$Y_i \sim N(\boldsymbol{X_i}^T\boldsymbol{\beta}, \sigma^2), \tag{3.12}$$

normal linear model

we get the *normal linear model*. In the normal linear model, we can easily test hypotheses regarding individual β_j parameters as well as compare the full model[23] against a submodel[24] asking whether the full model explains the reponse significantly better compared to the submodel. These hypotheses tests can be found in textbooks on linear models.

[23] model explaining the response using all the regressors

[24] model explaining the response using a subset of regressors

3.3.2 *Examples*

generality of linear models

Linear models, however simple, are quite a general means of modeling data, which can be illustrated by noticing that many apparently diverse statistical techniques are, in fact, only a special case of a linear model. Indeed, linear models span, for example:

- simple linear regression, in which each response is modeled in terms of a single regressor (plus an implicit intercept term),
- multiple linear regression, in which the expected value of the response is modeled not by one, but multiple regressors,
- one sample (or paired) t-test,
- two-sample t-test (assuming both samples share the same variance),
- analysis of variance (ANOVA), a generalization of the two-sample t-test to more than two groups,
- analysis of covariance (ANCOVA), which can be considered a generalization of ANOVA adjusting for a shared covariate.

non-linearity of linear models

Also note that even though the term ***linear** model* can evoke the impression that these models are unable to grasp non-linearity in data, it is not necessarily true, as *linear models* are linear in their regressors, which, however, can be non-linear in respect to some variable of interest, even to one of the regressors. To realize this, consider modeling temperature (measured over a whole day long) in terms of the time of the day. The relation between a particular temperature $Temperature_i$ measured at time $time_i$ can be written as:

$$Temperature_i = \beta_0 + \beta_1 time_i + \epsilon_i, \tag{3.13}$$

where β_0 stands for the temperature at time 0 (i.e., the intercept) and β_{time} represents the mean increase in temperature over unit time increment. While such a simple model can express only a linear relationship between the temperature and the day time, adding the square of the day time as an additional regressor[25] leads to a model which is linear in regressors, but clearly non-linear in the day time:

$$Temperature_i = \beta_0 + \beta_1 time_i + \beta_2 time_i^2 + \epsilon_i. \tag{3.14}$$

[25] This is a toy example only, as the model with the quadratic time would suffer from the multicolinearity issue, i.e. the matrix of regressors could be rank-deficient. This could be solved e.g. by modeling the quadratic effects using orthogonal polynomials of the day time. See (Faraway, 2002), section 9.4 to learn more.

Non-linear dependencies can enter the model not only in terms of polynomials, but also through *splines* (see Faraway (2002), section 8.3), which makes linear models even more powerful.

3.3.3 Notes on Terminology

Linear models are also called *regression models* for history reasons[26]. Sometimes, people tend to use the term ***general*** *linear model*[27] instead of a *linear model*, even though these terms are equivalent. Perhaps, the term ***general*** *linear model* should evoke the generality of linear models, namely the ability to use more than a single regressor in the model. However, the preferred and more standard way of calling a model with several regressors is *multiple linear regression*. In contrast, the term *multivariate linear regression* is used when modeling multivariate responses, i.e. when the response measured on individual units is multidimensional.

[26] Galton, Francis . Family Likeness in Stature. *Proceedings of the Royal Society of London*, 40:42–83, Jan 1886. DOI: 10.1098/rspl.1886.0009; and Galton F. Regression Towards Mediocrity in Hereditary Stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263, 1886. ISSN 09595295. DOI: 10.2307/2841583

[27] Friston K. J, Holmes A. P, Worsley K. J, Poline J. P, Frith C. D, and Frackowiak R. S. J. Statistical parametric maps in functional imaging: A general linear approach. *Hum. Brain Mapp.*, 2(4): 189–210, 1994. ISSN 1097-0193. DOI: 10.1002/hbm.460020402

3.4 Generalized Linear Models

Linear models provide a flexible means of modeling responses following normal (Gaussian) distribution, but they can't be directly used to model other types of responses, e.g. binomial, or count (Poisson) data. However, as the concept of explaining the response using a linear combination of regressors turned out to be highly useful in linear models, it has been extended by John Nelder and Robert Wedderburn[28] to be able to model non-Gaussian responses as well.

[28] Nelder J. A and Wedderburn R. W. M. Generalized linear models. *Journal of the Royal Statistical Society, Series A, General*, 135:370–384, 1972

3.4.1 Definition

The generalization of the *generalized linear models (GLM)* lies in two concepts: the generalization of the distribution of the random response, and the generalization of the way how linear predictor constructed from the regressors drives the random process modeling the response. In linear models, the response distribution is Gaussian. In generalized linear models, response following any distribution from the family of exponential distributions (see below for definition) can be modeled. While in linear models the expected value of the response is equal to the linear predictor, in GLM the expected value of the response is set equal to some function of the linear predictor.

Formally, independent and identically distributed (i.i.d.) random scalar variables Y_i can be modeled using a GLM, if the following conditions are met:

definition of GLM

1. the conditional distribution of Y_i, given the regressors, is an exponential family with probability density:

$$f(y_i|\theta_i, \varphi) = \exp\left(\frac{y_i\theta_i - b(\theta_i)}{\varphi} + c(y_i, \varphi)\right), \tag{3.15}$$

   where θ_i are response-specific real location parameters that depend on regressors via a linear predictor, and φ is a positive dispersion parameter of the distribution; the functions $b(.)$ and $c(.)$ are known and determine which member of the exponential distribution family is used,

2. the response can be modeled in terms of a function of the location parameter θ_i:

$$E(Y_i|\boldsymbol{X_i}) = \mu_i = \mu_i(\theta_i), \tag{3.16}$$

3. the conditional expected value of the response $\mu_i(\theta_i)$ depends on the linear predictor $\eta_i = \boldsymbol{X_i}^T\beta$, and this dependency can be described in terms of a *link function* g:

link function

$$\eta_i = g(\mu_i), \tag{3.17}$$

where g must be strictly increasing and twice differentiable. Note that η_i, the linear combination of regressors, can in general take any real value. The purpose of the link function is therefore to map μ_i, the (conditional) expected value of the response[29] to a real number. The other way around, the inverse of the link function transforms the linear combination of regressors onto the scale of the expected value of the response. The transforms of the θ_i, μ_i, and η_i parameters are depicted in Fig. 3.3.

[29] whose range can be limited

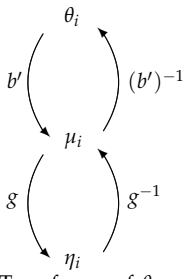

Figure 3.3: Transforms of θ_i, μ_i, and η_i parameters in GLM.

It can be shown that the function $b(.)$ is connected to both the expected value and the variance of the response:

$$E(Y_i|\boldsymbol{X_i}) = \mu_i = b'(\theta_i) \tag{3.18}$$

$$var(Y_i|\boldsymbol{X_i}) = \varphi V(\mu_i) = \varphi b''(\theta_i) \tag{3.19}$$

where $V(.)$ is the *variance function* connecting the variance of Y_i to its mean value:

variance function

$$V(\mu_i) = b''(\theta_i) = b''\left((b')^{-1}(\mu_i)\right) \tag{3.20}$$

In fact, having known the $b(.)$ function (and, thus, the variance function), the distribution is determined, up to the scale parameter φ, by its mean.

The variance function of each distribution in the exponential family is unique and determines the distribution. However, we can not restore the function $b(.)$ from $V(.)$ in general.

GLM parameter estimation

An important feature of GLM is that the likelihood function, i.e. the probability density function (eq. 3.15) taken as a function of the θ_i parameters, is convex. This means that it is easy to find the value of the θ_i parameters by finding the global optimum of the likelihood function. The parameters in generalized linear models are usually estimated using maximum likelihood estimation procedures numerically, by the iterative weighted least squares algorithm[30].

[30] McCullagh P and Nelder J. A. *Generalized linear models (Second edition)*. London: Chapman & Hall, 1989

3.4.2 *Examples*

To illustrate the notation described, let's consider a few examples. First, we express the linear regression model (which, of course, is a special case of GLM) using the notation defined above to realize it indeed collapses to a linear model. Second, we describe the logistic regression model, and finally focus on Poisson regression, which we will further discuss in upcoming sections. Interested reader can found full details in McCullagh and Nelder (1989)[31].

[31] McCullagh P and Nelder J. A. *Generalized linear models (Second edition)*. London: Chapman & Hall, 1989

linear regression

In a linear regression model, the i-th response is drawn from a

normal distribution with mean μ_i and variance σ^2:

$$Y_i \sim N(\mu_i, \sigma^2) \tag{3.21}$$

The standard form of the normal probability density

$$f(y_i; \mu_i, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i-\mu_i)^2}{2\sigma^2}} \tag{3.22}$$

can be rewritten as the density of an exponential family distribution as

$$f(y_i; \mu_i, \sigma^2) = \exp\left(\frac{y_i\mu_i - \frac{\mu_i^2}{2}}{\sigma^2} - \left(\frac{y_i^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)\right)\right)$$

such that

$$\theta_i = \mu_i, \quad \varphi = \sigma^2, \quad b(\theta_i) = \frac{\theta_i^2}{2} = \frac{\mu_i^2}{2},$$
$$c(y_i, \varphi) = -\left(\frac{y_i^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)\right)$$

yielding

$$E(Y_i|\boldsymbol{X_i}) = b'(\theta_i) = \theta_i = \mu_i$$
$$var(Y_i|\boldsymbol{X_i}) = \varphi\, b''(\theta_i) = \varphi = \sigma^2.$$

Note that the variance function $V(\theta_i)$ does not depend on μ_i and is

$$V(\mu_i) = 1.$$

In fact, the normal distribution is the only exponential family distribution in which the variance does not depend on the mean.

As the link function used in the linear model is the identity:

$$g(\mu_i) = \mu_i,$$

the conditional mean can be expressed as

$$E(Y_i|\boldsymbol{X_i}) = \mu_i = g^{-1}(\eta_i) = \eta_i = \boldsymbol{X_i}^T\beta,$$

i.e. the response is modeled directly in terms of the linear predictor, the linear combination of the regressors $\boldsymbol{X_i}$.

logistic regression

In case of the logistic regression, we model responses drawn from the alternative distribution of successes and failures:

$$Y_i \sim Alt(p_i), \tag{3.23}$$

where the success in i-th attempt comes with the probability $p_i \equiv \mu_i$ (i.e., the probability is the location parameter μ_i). The probability density function

$$f(y_i; p_i) = p_i^{y_i}(1-p_i)^{1-y_i} \tag{3.24}$$

can be expressed as the density of an exponential family distribution as

$$\begin{aligned} f(y_i; p_i) &= \exp\left\{\log(p_i^{y_i}) + \log\left((1-p_i)^{1-y_i}\right)\right\} \\ &= \exp\left\{y_i \log p_i + (1-y_i)\log(1-p_i)\right\} \\ &= \exp\left\{y_i\left(\log p_i - \log(1-p_i)\right) - (-\log(1-p_i))\right\} \\ &= \exp\left\{y_i \log\left(\frac{p_i}{1-p_i}\right) - (-\log(1-p_i))\right\}, \end{aligned}$$

such that

$$\theta_i = \log\left(\frac{p_i}{1-p_i}\right), \quad p_i = \frac{e^{\theta_i}}{1+e^{\theta_i}}, \quad 1-p_i = \frac{1}{1+e^{\theta_i}}, \quad \varphi = 1$$

and

$$b(\theta_i) = -\log(1-p_i) = \log(1+e^{\theta_i}), \quad c(y_i, \varphi) = 0$$

yielding

$$\begin{aligned} E(Y_i|\boldsymbol{X_i}) &= b'(\theta_i) = \frac{e^{\theta_i}}{1+e^{\theta_i}} = p_i \\ var(Y_i|\boldsymbol{X_i}) &= \varphi\, b''(\theta_i) = b''(\theta_i) = \frac{e^{\theta_i}}{1+e^{\theta_i}} - \frac{e^{2\theta_i}}{(1+e^{\theta_i})^2} = \\ &= \frac{e^{\theta_i}(1+e^{\theta_i}) - e^{2\theta_i}}{(1+e^{\theta_i})^2} = \frac{e^{\theta_i}}{(1+e^{\theta_i})^2} = \frac{e^{\theta_i}}{(1+e^{\theta_i})} \cdot \frac{1}{(1+e^{\theta_i})} = \\ &= p_i(1-p_i), \end{aligned}$$

as expected in the alternative distribution. The variance function, in contrast to the linear model, depends on the location parameter $p_i \equiv \mu_i$ and is

$$V(\mu_i) = \mu_i(1-\mu_i).$$

The appropriate link function turns out to be the "logit" function

$$\eta_i = g(\mu_i) = log\left(\frac{\mu_i}{1-\mu_i}\right), \tag{3.25}$$

such that its inverse is the "expit" function (Fig. 3.4)

$$\mu_i = g^{-1}(\eta_i) = \frac{e^{\eta_i}}{1+e^{\eta_i}}, \tag{3.26}$$

which leads to the average response to be expressed in terms of the linear combination of regressors as:

$$E(Y_i|\boldsymbol{X_i}) = \mu_i = g^{-1}(\eta_i) = g^{-1}(\boldsymbol{X_i}^T\beta) = \frac{e^{\boldsymbol{X_i}^T\beta}}{1+e^{\boldsymbol{X_i}^T\beta}}, \tag{3.27}$$

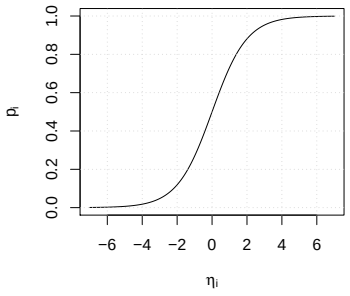

Figure 3.4: The *expit* function transforming the linear predictor η_i into probability p_i.

i.e., the average response, the expected probability of success, is bounded in the $(0, 1)$ range.

Poisson regression

Poisson regression can be used to model count responses, i.e. observations drawn from the Poisson distribution. The Poisson distribution is parametrized with $\lambda_i \equiv \mu_i$ parameters representing the mean number of events occurring in unit time. The probability density of the Poisson distribution is

$$f(y_i; \lambda_i) = \frac{\exp(-\lambda_i)\lambda_i^{y_i}}{y_i!} \tag{3.28}$$

and can be expressed as the density of an exponential family distribution as

$$f(y_i; \lambda_i) = \exp\{y_i \log(\lambda_i) - \lambda_i - \log(y_i!)\} \tag{3.29}$$

such that

$$\theta_i = \log(\lambda_i), \quad \varphi = 1$$

and

$$b(\theta_i) = \lambda_i = e^{\theta_i}, \quad c(y_i, \varphi) = -\log(y_i!)$$

yielding

$$\begin{aligned} E(Y_i|\boldsymbol{X_i}) &= b'(\theta_i) = e^{\theta_i} = \lambda_i \\ var(Y_i|\boldsymbol{X_i}) &= \varphi\, b''(\theta_i) = b''(\theta_i) = e^{\theta_i} = \lambda_i. \end{aligned}$$

The appropriate link function turns out to be the $\log(.)$ function here:

$$g(\lambda_i) = \log(\lambda_i), \tag{3.30}$$

such that the inverse of the link is the $\exp(.)$ function:

$$g^{-1}(\eta_i) = \exp(\eta_i) \tag{3.31}$$

and the expected value of the response is

$$E(Y_i|\boldsymbol{X_i}) = \lambda_i = g^{-1}(\eta_i) = g^{-1}(\boldsymbol{X_i}^T\beta) = e^{\boldsymbol{X_i}^T\beta}, \tag{3.32}$$

such that we model the logarithm of the response in terms of the linear combination of the regressors.

3.4.3 Notes on Terminology

Poisson regression models are often referred to as *loglinear models*, due to the fact that they relate the logarithm of the response to a linear combination of regressors.

*Generaliz**ed** linear models* are often confused with *general linear models*, although they are quite distinct. John Nelder, who had co-proposed generalized linear models, commented this in an interview with Stephen Senn[32] in this way:

> **Senn:** I must confess to having some confusion when I was a young statistician between general linear models and generalized linear models. Do you regret the terminology?
> **Nelder:** I think probably I do. I suspect we should have found some more fancy name for it that would have stuck and not been confused with the general linear model, although general and generalized are not quite the same. I can see why it might have been better to have thought of something else.

[32] Senn S. A conversation with John Nelder. *Statist. Sci.*, 18(1):118–131, 2003. ISSN 0883-4237

3.5 Modeling Spike Train Data

spike train

The activity of a single neuron can be abstracted by a *spike train*, a sequence $S_1, S_2, \ldots$ of times at which the neuron fires a spike. Of course, such an abstraction is reasonable provided we are interested in studying the information the neuron produces, not the intrinsic neuronal processes resulting in firing the spikes (such as the synapses, the membrane potential etc.[33]). This section gives an overview of building probabilistic models of spike trains, and fitting these models to real data using the GLM framework.

[33] see Section 2.1

3.5.1 Spike Train as a Stochastic Process

point process

The spike train, the sequence $S_1, S_2, \ldots$ of times at which given neuron fires a spike, can be considered to be the outcome of a stochastic process. We call such a process the *point process*. The process is *stochastic*, in general, because we do not (and can not) measure all the variables that determine the neuronal firing. All we can do is to express the **probability** that a neuron fires a spike in a given narrow time interval $(t, t + \Delta t]$. In the simplified case in which the occurrence of individual spikes is independent on each other, the spiking probability can be expressed unconditionally as $P(\text{spike in } (t, t + \Delta t])$, and described by a special case of the point process, the *Poisson process*. The Poisson process lacks memory, meaning that the probability that a neuron fires a spike at any given time is independent on the previous spiking.

Poisson process

The probability of firing a spike can be constant over time (in case of the *homogeneous* Poisson process), or can depend on time (in case of the *inhomogeneous* Poisson process). However, although simple and mathematically appealing, the Poisson process can't be used to model real spike trains. For example, it can't capture the refractoriness of neurons, i.e., it can't express the highly decreased probability of firing a spike immediately following the past one. (This inability is dictated by the biophysical properties of neuronal ion channels, which, after firing a spike, can recover only after a given time interval called the *refractory period*[34].)

[34] see Section 2.1 for details

A more realistic model of spiking must enable the present activity to be influenced by the past spiking activity. Had the present activity been influenced only by the immediately preceding spike, we would have used the *renewal process*. In general, however, the probability of firing a spike in a given narrow time interval $(t, t + \Delta t]$ becomes conditional not only on the immediate history (e.g. the last spike), but the whole history. The probability of firing a spike in a given time interval then can be expressed conditioning on history as $P(\text{spike in } (t, t + \Delta t] | H_t)$, where H_t stands for the whole spiking his-

renewal process

tory prior time t. The fact that, in general, the probability depends on the whole history means that the corresponding process is called the (general) point process, which has memory.

The conditional probability depends on three variables: t, Δt, and H_t. For simplicity reasons, we omit the Δt parameter and characterize the corresponding point process equivalently by its *conditional intensity function*[35], $\lambda(t|H_t)$, defined as follows:

conditional intensity function

$$\lambda(t|H_t) = \lim_{\Delta t \to 0} \frac{P(\text{spike in } (t, t+\Delta t]|H_t)}{\Delta t}, \tag{3.33}$$

[35] Daley D. J and Vere-Jones D. *An Introduction to the Theory of Point Processes, Volume 1 (2nd ed.)*. Springer, New York, June 2003. ISBN 0387955410

which can be expressed for small Δt as

$$P(\text{spike in } (t, t+\Delta t]|H_t) \approx \lambda(t|H_t)\Delta t, \tag{3.34}$$

providing an intuitive meaning to the conditional intensity function: the conditional intensity function multiplied by a small Δt yields the probability that a neuron fires in a small time interval Δt after time t. Thus, the conditional intensity function can be though of as the instantaneous firing frequency.

To express the fact that the stochastic firing with the intensity $\lambda(t|H_t)$ depends on another stochastic process, the history H_t, the corresponding process is sometimes called the *doubly stochastic point process*.

doubly stochastic point process

Note that when the conditional intensity function $\lambda(t|H_t)$ does not depend on history H_t, the general point process reduces back to the Poisson process.

3.5.2 *Discrete Representation of Spike Trains*

So far, we've been modeling spike trains using continuous-time point processes. In that case, we considered Δt to be infinitesimally small. To fit real data, however, it will be convenient to make Δt small, but finite. Thus, we divide the time interval $(0, T]$, in which all the spikes in the spike train $S_1, S_2, \ldots, S_n$ reside, into $K = T/\Delta t$ short bins $(t_k, t_{k+1}]$ of length Δt, such that at most one spike occurs in each bin. This way, we approximate the spike train consisting of events $S_1, S_2, \ldots, S_n$ by D_k, $k = 1 \ldots K$, a sequence of 0s and 1s of length K, in which all 1s represent bins in which a spike occurred, and 0s represent bins of no spike (Fig. 3.5).

binning a spike train

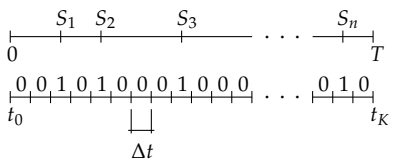

Figure 3.5: Approximation of a continuous-time spike train $S_1, S_2, \ldots, S_n$ by a discrete sequence of Bernoulli events D_k, $k = 1 \ldots K$.

Equivalently, we can describe the discretized version of the spike train by the counting process N_k, $k = 0 \ldots K$, in which each N_k holds the cumulative number of spikes fired up to the k-th bin. The relation between D_k and N_k is thus:

$$N_k = \sum_{i=1}^{k} D_i,$$
$$D_k = N_k - N_{k-1} \stackrel{def}{=} \Delta N_k.$$

Under this representation, the spike train can be analysed as a Bernoulli process of conditionally independent events.

The probability of a spike in time interval $(t_k, t_{k+1}]$ is the following:

$$P(\text{spike in } (t_k, t_{k+1}]|H_t) \approx \lambda(t|H_t)\Delta t, \tag{3.35}$$

and the probability of no spike is:

$$P(\text{no spike in } (t_k, t_{k+1}|H_t) \approx 1 - \lambda(t|H_t)\Delta t. \tag{3.36}$$

The probability of observing exactly n spikes in bins in which $D_k = 1$ is then

$$\begin{aligned} P(\text{spikes at } D_k = 1) &\approx \prod_{k=1}^{K} [\lambda(t_k|H_k)\Delta t]^{D_k}[1 - \lambda(t_k|H_k)\Delta t]^{1-D_k} = \\ &= \prod_{k=1}^{K} [\lambda(t_k|H_k)\Delta t]^{D_k}[1 - \lambda(t_k|H_k)\Delta t]^{-D_k} \prod_{k=1}^{K} [1 - \lambda(t_k|H_k)\Delta t] = \\ &= \prod_{k=1}^{K} \left[\frac{\lambda(t_k|H_k)\Delta t}{1 - \lambda(t_k|H_k)\Delta t}\right]^{D_k} \prod_{k=1}^{K} [1 - \lambda(t_k|H_k)\Delta t], \end{aligned} \tag{3.37}$$

where H_k stands for the whole spiking history up to bin $k-1$.

For small Δt, $\log\{\lambda(t_k)\Delta t[1 - \lambda(t_k)\Delta t]^{-1}\} \approx \log(\lambda(t_k)\Delta t)$ and $[1 - \lambda(t_k)\Delta t] \approx \exp(-\lambda(t_k)\Delta t)$, therefore we obtain

$$P(\text{spikes at } D_k = 1) \approx \exp\left\{\sum_{k=1}^{K} \log[\lambda(t_k|H_k)\Delta t]D_k - \sum_{k=1}^{K} \lambda(t_k|H_k)\Delta t\right\} \tag{3.38}$$

Going back to the continuous-time domain (remembering that $D_k \stackrel{def}{=} \Delta N_k$, and turning N_k into continuous $N(t)$), the probability of observing exactly n spikes in $(0, T]$ is

$$
\begin{aligned}
P(\text{spikes at } S_1, S_2, \ldots, S_n) &= \lim_{\Delta t \to 0} \frac{\exp\left\{\sum_{k=1}^{K} \log[\lambda(t_k|H_k)\Delta t]\Delta N_k - \sum_{k=1}^{K} \lambda(t_k|H_k)\Delta t\right\}}{(\Delta t)^n} \\
&= \lim_{\Delta t \to 0} \frac{\exp\left\{\sum_{k=1}^{K} \log[\lambda(t_k|H_k)]\Delta N_k - \sum_{k=1}^{K} \lambda(t_k|H_k)\Delta t\right\}(\Delta t)^n}{(\Delta t)^n} \\
&= \exp\left\{\int_0^T \log(\lambda(t|H_t))dN(t) - \int_0^T \lambda(t|H_t)dt\right\},
\end{aligned}
\tag{3.39}
$$

which is the joint probability density of the spike train-modeling point process[36].

We obtained the probabilistic representation of a spike train, namely the conditional intensity function $\lambda(t|H_t)$, which defines the spike train in terms of eq. 3.33. Note that the the form of the probability density in eq. 3.39 is of the same type as the probability density of the Poisson random variable shown in eq. 3.29. Thus, we can use GLM to fit the spike train data, making use of all the beauty of GLM, namely the robustness and computation efficiency of the GLM framework. Also, the diagnostics and analytical tools developed for GLM can be introduced in the context of spike train data modeling.

[36] Daley D. J and Vere-Jones D. *An Introduction to the Theory of Point Processes, Volume 1 (2nd ed.)*. Springer, New York, June 2003. ISBN 0387955410

using GLM to model spike train data

3.5.3 *Examples*

To illustrate[37] modeling spike train data by specifying the conditional intensity function, consider a simple example of a neuron firing spikes at a constant intensity. However, note that neurons are not able to fire an action potential just after the previous one has been fired - the neuron needs some time (the *refractory period*) to recover. Therefore, the conditional intensity function of this neuron needs to depend on the previous firing, as the intensity must be decreased temporarily after each spike has been fired.

[37] after Eden (2008)

Let's approximate the spike train by a sequence of conditionally independent Bernoulli events by binning the $(0, T]$ time interval into bins of length 1 ms and define the conditional intensity function as

$$
\lambda(t_k|H_k) = \exp\left\{\alpha_0 + \sum_{i=1}^{4} \alpha_i \Delta t D_{k-i}\right\}, \tag{3.40}
$$

where t_k is the time representing the bin k, and H_k is the spiking history up to bin $k-1$. The conditional intensity function $\lambda(t_k|H_k)$ does not depend on the whole history H_k, but on the last four bins only $(D_{k-1}, D_{k-2}, D_{k-3}, D_{k-4})$, i.e., we can model the refractory period of up to 4 ms. This model has four covariates (spiking history D_{k-i}) and five coefficients $(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4)$. Consider we take $\alpha_0 = log(10)$, $\alpha_1 = -100$, $\alpha_2 = -2$, $\alpha_3 = -0.5$, and $\alpha_4 = -.01$ (Fig. 3.6).

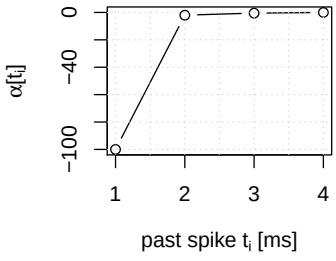

Figure 3.6: Coefficients corresponding to history covariates in the model of conditional intensity function of a hypothetical neuron exhibiting refractory period.

How does the conditional intensity function encode the (instantaneous) firing frequency? If at any point in time no spikes have occurred in the past 4 ms (i.e., calculating the intensity in the time bin k, there have been no spikes in the $D_{k-1}, D_{k-2}, D_{k-3}, D_{k-4}$ bins), the firing intensity is $\lambda(t_k|H_k) = \exp\{\alpha_0\} = \exp(\log(10)) = 10$ spikes per second. If a spike has occurred in the past millisecond (and it has been the only spike in the past 4 ms), then the firing intensity drops to $\lambda(t_k|H_k) = \exp(\log(10) - 100) \approx \exp(-97.7)$, which is negligibly small. It means that it is virtually impossible for this neuron to fire a spike within 1 ms of a previous spike. If a spike occurred 2 ms previously, and no other spike had been fired in the 4 ms history, then the firing intensity is $\lambda(t_k|H_k) = \exp(\log(10) - 2) \approx 1.35$ spikes per second, which is a substantial drop from the baseline firing rate of 10 spikes per second, but not as small as it was immediately after a spike. It illustrates the fact that inhibitory effect of a spike diminishes as it recedes into the past. The absolute refractory period of this neuron is 1 ms (it cannot fire), and the relative refractory period is 4 ms (the probability of firing is decreased). If, however, we had one spike 2 ms in the past and another spike 4 ms in the past, then the inhibitory effects of each past spike combine and we get $\lambda(t_k|H_k) = \exp(\log(10) - 2 - .1) \approx 1.22$ spikes per second, which is less than the intensity caused by either past spike individually. This simple example demonstrates that the precise timing of the previous spiking activity can alter current spiking propensity, and that this can be modeled with a conditional intensity function.

Note that the curve depicted in Fig. 3.6 is typical for real neurons exhibiting a refractory period: the coefficients corresponding to immediately preceding spikes are highly negative, and then increase reaching slightly negative values for lags equal to the refractory period of the studied neuron. Thus, the coefficients driving the contribution of the past spikes to the intensity function can serve as a means of learning the properties of the neuron.

On the other hand, in the hypothetical case of a neuron exhibiting no refractory period, the coefficients corresponding to immediately preceding spikes would have been zero (Fig. 3.7). Such an hypothetical neuron would have been modeled by a simple memory-less Poisson process, without the need to consider the past spiking activity.

To demonstrate the ability of the conditional intensity function to model a regularly spiking neuron, consider a hypothetical neuron that has the tendency to fire an action potential every 10 ms, i.e. at the frequency of 100 spikes per second. (Fig. 3.8). In this example, the coefficients corresponding to the history covariates are negative for small lags (this captures the refractory period), but then they increase to reach a positive peak of $\log(10)$ for the lag of 10 ms. In case an

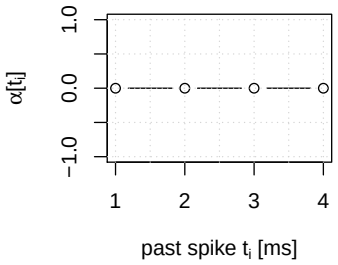

Figure 3.7: Coefficients corresponding to the history covariates in the model of conditional intensity function of a hypothetical neuron exhibiting no refractory period. Such neuron could have been modeled by the memory-less Poisson process.

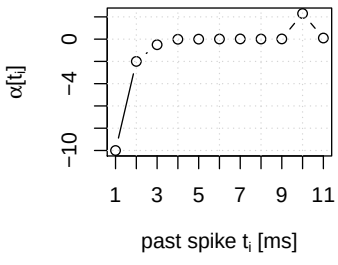

Figure 3.8: Coefficients corresponding to the history covariates in the model of conditional intensity function of a hypothetical regularly firing neuron.

action potential has been fired 10 ms ago, the firing intensity $\lambda(t_k|H_k)$ is increased from the baseline intensity of $\exp(\log(10)) = 10$ spikes per second to $\exp(\log(10) + \log(10)) = 100$ spikes per second. On average, the neuron fires quite regularly at the frequency of 100 spikes per second, but, however, as the process driving the neuronal firing is still stochastic, the neuron fires also casual action potentials.

Note that the covariates that appear in the conditional intensity function describing the instantaneous firing intensity of a neuron are not limited to the past spiking activity of that neuron. We can incorporate any other relevant information in the conditional intensity function of a behaving neurons, such as concurrent ensemble activity, stimuli, or animal's behavior[38].

The framework of GLM has been successfully used in modeling real spiking data. Eden et al. studied the spiking dynamics of the STN in Parkinson's disease[39]. Using the point process GLM, Sarma et al. revealed a difference in spiking activity in the STN of Parkinson's disease and a healthy primate[40]. Pedoto et al. showed that GLM is able to reveal anatomical non-uniform distribution of spiking activity across the STN in Parkinson's disease[41].

[38] Truccolo W, Eden U. T, Fellows M. R, Donoghue J. P, and Brown E. N. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *J. Neurophysiol.*, 93(2):1074–1089, Feb 2005

[39] Eden U. T, Gale J. T, Amirnovin R, and Eskandar E. N. Characterizing the spiking dynamics of subthalamic nucleus neurons in Parkinson's disease using generalized linear models. *Front Integr Neurosci*, 6:28, 2012

[40] Sarma S. V, Eden U. T, Cheng M. L, Williams Z. M, Hu R, Eskandar E, and Brown E. N. Using point process models to compare neural spiking activity in the subthalamic nucleus of Parkinson's patients and a healthy primate. *IEEE Trans Biomed Eng*, 57(6):1297–1305, Jun 2010

[41] Pedoto G, Santaniello S, Fiengo G, Glielmo L, Hallett M, Zhuang P, and Sarma S. V. Point process modeling reveals anatomical non-uniform distribution across the subthalamic nucleus in Parkinson's disease. *Conf Proc IEEE Eng Med Biol Soc*, 2012:2539–2542, 2012

3.5.4 Diagnostics

Data modeling consists not only of designing a model for given data and fitting the model to the data. Of enormous importance is to diagnose how well the model fits the data. The standard way of assessing the fit of a model is to study the residuals of the model, and ensure that the residuals are negligibly small and that there are no non-random patterns left in them. Any non-random structure left in what the model has not fitted would imply that the model has failed to capture the structure of the data.

model residuals

In case of linear models, in which the response is continuous, it is relatively easy to come up with residual diagnostics. However, for generalized linear models, in which the response is not continuous, in general, such direct approaches cannot be applied. One possible

way to go would be to transform data to the continuous domain, and apply some diagnostics developed for continuous responses. The most widely used technique in the context of spike train data is to apply the *time rescaling theorem* on the point process data, and employ goodness-of-fit tests on the results of this transform.

time rescaling theorem

The time rescaling theorem states that the inter-spike intervals (ISIs) of any continuous-time point process can be transformed (rescaled) such that the resulting process is Poisson with unit rate. Informally, the idea is as follows. While the conditional intensity function $\lambda(t|H_t)$, i.e. the instantaneous firing frequency, can be considered to be the (instantaneous) rescaling factor converting a stochastic Poisson spiking with unit rate into the modeled point process, the conditional intensity function can also be used the other way round - to *rescale* the ISIs of the modeled point process back to the Poisson process with unit rate. Formally, the time rescaling theorem states that ISIs of any continuous-time point process with conditional intensity function $\lambda(t|H_t)$ and events occurring at $S_1, S_2, \ldots$ can be transformed (rescaled), such that the transformed process is homogeneous Poisson with unit rate, whose ISIs are, by definition, independent and exponentially distributed. The transform is defined by:

$$\tau_1 = \int_0^{S_1} \lambda(t|H_t)dt, \text{ and} \tag{3.41}$$

$$\tau_i = \int_{S_{i-1}}^{S_i} \lambda(t|H_t)dt, \text{ for } i > 1, \tag{3.42}$$

where the resulting $\tau_1, \tau_2, \ldots$ ISIs are independent and identically distributed (i.i.d.) $Exp(1)$ random variables. The intuition behind the equation is that the intensity $\lambda(t|H_t)$ accumulates over the intervals between spikes $S_1, S_2, \ldots$. If the time interval between two spikes is long, the intensity should be low, on average (as the intensity is faithful to the process generating the spikes) and the accumulated value would be similar to the value accumulated over a much shorter time interval, in which, however, the corresponding intensity function should be high. The fact that long ISIs are shrunk and short ISIs are stretched, i.e. the time gets rescaled, gave rise to the name of the transform. The proof of the time rescaling theorem can be found e.g. in Brown et al. (2002)[42].

[42] Brown E. N, Barbieri R, Ventura V, Kass R. E, and Frank L. M. The time-rescaling theorem and its application to neural spike train data analysis. *Neural Comput.*, 14(2):325–346, February 2002. DOI: 10.1162/08997660252741149

The time rescaling theorem can be used as follows. If a model is faithful to the process it tries to cover, its intensity function can be used to rescale the ISIs of the process, which should result in i.i.d. $Exp(1)$ ISIs. As the time rescaling transform (Eq. 3.42) is one-to-one, the rescaled ISIs are i.i.d. $Exp(1)$ if and only if the conditional intensity function (and the model) captures the modeled process well. Thus, the faithfulness (quality) of the model can be assessed indirectly by assessing whether the rescaled ISIs are i.i.d. $Exp(1)$.

It is not easy to assess whether the rescaled ISIs are of Poisson process with unit rate, i.e. whether they are i.i.d. $Exp(1)$. The problem is that there are many ways in which the "i.i.d. $Exp(1)$" assumption can be violated. If there were a statistical test that would have been used to reject the null hypothesis of "i.i.d. $Exp(1)$" in favor of a general alternative, its power could have been too low[43] to be practical. On the other hand, there are special tests guarding against simple alternatives[44], but the problem is which one to perform: there is a trade-off between the number of such special tests and the power of such battery of tests. The reason is that the more tests we perform, the higher the probability of detecting a true violation of the null hypothesis in the data. At the same time, however, the more tests we perform, the higher the probability of making at least one false discovery, while any attempts to keep the probability of making at least one false discovery at some predefined level (e.g., 5%) would again decrease the power of such a battery of tests.

I describe two widely used methods to assess the distribution assumption of the rescaled ISIs. Other methods can be found in literature [45].

Tests of independence can be found in literature on time series[46].

[43] often, it would not have rejected the null even if the null had been false

[44] such that their power is high

[45] Ogata Y. Statistical Models for Earthquake Occurrences and Residual Analysis for Point Processes. *Journal of the American Statistical Association*, 83 (401):9+, March 1988. ISSN 01621459. DOI: 10.2307/2288914; and Pouzat C and Chaffiol A. On Goodness of Fit Tests For Models of Neuronal Spike Trains Considered as Counting Processes. 2008. URL `https://sites.google.com/site/spiketrainanalysiswithr/`

[46] Brockwell P. J and Davis R. A. *Time Series: Theory and Methods (Springer Series in Statistics)*. Springer, 2nd ed. 1991. 2nd printing 2009 edition, April 2009. ISBN 1441903194

To assess how close the observed rescaled ISIs are to $Exp(1)$ distribution, we can make use of the visual means of quantile-quantile plots and Kolmogorov-Smirnov goodness-of-fit plots.

quantile-quantile (Q-Q) plot

The quantile-quantile (Q-Q) plot can be constructed by plotting the quantiles of the observed rescaled ISIs against the theoretical quantiles of the $Exp(1)$ distribution. In practice, we sort the rescaled ISIs $\tau_1, \tau_2, \ldots, \tau_n$ and plot them against the quantiles of the $Exp(1)$ distribution $e_1, e_2, \ldots, e_n$ computed by transforming the quantiles of the uniform distribution $U(0,1)$ using the inverse $Exp(1)$ cumulative distribution function (CDF) as:

$$e_i = CDF^{-1}_{Exp(1)}(u_i) = -ln(1-u_i), \tag{3.43}$$

where u_i are quantiles of the uniform $U(0,1)$ distribution set to

$$u_i = \frac{i - \frac{1}{2}}{n},\ i = 1 \ldots n. \tag{3.44}$$

If the model is correct, i.e. the transformed ISIs follow the $Exp(1)$ distribution, the points in the plot should lie on a straight diagonal (45-degree) line, as the observed quantiles should equal the theoretical ones. Any significant departures from the diagonal line would imply that the transformed ISIs do not follow $Exp(1)$ and that the model does not fit the data well. The magnitude of any departures from

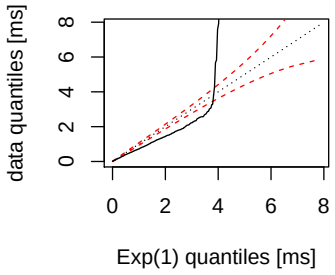

Figure 3.9: Q-Q plot showing the discrepancy between the theoretical quantiles and the quantiles of the rescaled ISIs (black solid line). The ideal agreement is depicted as a diagonal line (dotted). The 95% point-wise confidence bounds are plotted as dashed red curves. See text for details.

the straight line can be assessed using point-wise confidence bounds [47]. These bounds can be constructed by transforming the quantiles of a Beta distribution using the inverse $Exp(1)$ CDF. The confidence interval around each point (e_i, e_i) lying on the diagonal line is

$$e_i \pm CDF^{-1}_{Exp(1)}(\beta_i) = e_i \pm -ln(1-\beta_i), \tag{3.45}$$

where β_i is the quantile of the Beta distribution with parameters i and $n-i+1$. To construct the 95% confidence interval, we take the 2.5% and 97.5% Beta quantiles. An example of a Q-Q plot is given in Fig. 3.9, in which the goodness-of-fit of a spike train model is assessed. The plot shows that the corresponding model does not fit the data well, as demonstrated by the excursion of the points in the Q-Q outside the 95% confidence bounds. We can infer that the model fails to capture well ISIs longer than about 1 ms (note that the data quantiles corresponding to theoretical quantiles from 1 to 4 ms lie below the 95% confidence bounds).

[47] Brown E. N, Barbieri R, Ventura V, Kass R. E, and Frank L. M. The time-rescaling theorem and its application to neural spike train data analysis. *Neural Comput.*, 14(2):325–346, February 2002. DOI: 10.1162/08997660252741149

Kolmogorov-Smirnov (KS) plot

The Kolmogorov-Smirnov (KS) plot can be constructed by transforming the quantiles used in the Q-Q plots by the CDF of the $Exp(1)$ distribution, yielding values in the range of $(0,1)$. In practice, we transform the $\tau_1, \tau_2, \ldots, \tau_n$ by the $Exp(1)$ CDF yielding values $p_i = CDF_{Exp(1)}(\tau_i) = 1 - exp(-\tau_i)$ and plot them against the values of the CDF of the uniform $U(0,1)$ distribution $u_i = \frac{i-\frac{1}{2}}{n}$, $i = 1 \ldots n$. Similarly to the Q-Q plot, the points in the KS plot should lie along the diagonal line if the model is correct. For moderate to large sample sizes, the magnitude of any departures from the diagonal can be assessed by the simultaneous 95% confidence bounds constructed as:

$$u_i \pm \frac{1.36}{\sqrt{n}}, \tag{3.46}$$

or by the simultaneous 99% confidence bounds constructed as

$$u_i \pm \frac{1.63}{\sqrt{n}}, \tag{3.47}$$

see Brown et al. (2002).

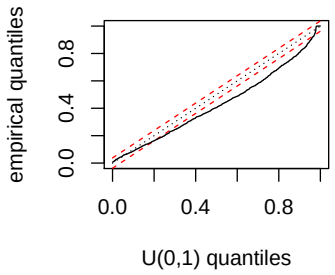

Figure 3.10: KS plot showing the correspondence between the theoretical $Exp(1)$ distribution and the data (black solid curve). As the curve is far from the diagonal line (dotted) depicting the ideal agreement and exceeds the 95% simultaneous confidence bounds (dashed red lines), the corresponding model does not fit the data well. See text for details.

An example of a KS plot is given in Fig. 3.10, in which the goodness-of-fit of the spike train model studied also in Fig. 3.9 is assessed. The plot shows the discrepancy between the theoretically expected $Exp(1)$ distribution and the rescaled ISIs. On the horizontal axis come the quantiles of the uniform $U(0,1)$ distribution (in fact, $Exp(1)$ quantiles transformed by the $Exp(1)$ CDF), on the vertical axis come the $Exp(1)$-CDF-transformed quantiles of the rescaled ISIs. The correspondence between the $Exp(1)$ and data is shown as a black solid line. The diagonal line (dotted) depicts the ideal agreement. The 95% simultaneous

confidence bounds are plotted as dashed red lines. The plot demonstrates that the corresponding model does not fit the data well: it fails to model about 80% of the ISIs (the solid black curve exceeds the confidence bands about 80% of the time). The plot is based on the same data as Fig. 3.9, from which we can see that the model fails to capture ISIs from 1 to 4 ms.

Note that the KS plot simultaneous confidence bounds are wider compared to the point-wise Q-Q confidence bounds. The KS 95% bounds consider the maximum discrepancy from the diagonal line (for all quantiles simultaneously). These bounds can detect discrepancies 5% of the time by chance if the plotted data were truly $Exp(1)$-distributed. Contrary, the Q-Q plot 95% confidence bounds are point-wise, i.e. they consider the maximum discrepancy from the the diagonal line for each quantile separately. If the data were truly $Exp(1)$-distributed, the data would have exceed the 95% confidence band for some fixed quantile 5% of the time. (Note that the 95% confidence bands in *any* quantile would have been exceeded much more often.)

4
Methodological Results

In this chapter I present the original methodological achievements reached. Section 4.1 presents a software framework that enables automated data access and processing. Section 4.2 describes an interactive data exploratory tool that can be used e.g. to study the shapes of spikes detected in single-neuron microelectrode recordings and to informally assess the quality of spike sorting methods. Section 4.3 evaluates existing spike sorting methods.

4.1 Data Access Objects - An Automated Way of Dealing with Data

Processing and analysis of large experimental data involves designing a simple, yet flexible way of accessing it. This section introduces the general concept of a special software component that gives access to data at multiple processing stages in a simple and automated way, and describes the implementation of this concept in the context of neurophysiological data.

4.1.1 Problem Definition

The analysis of large experimental data can be associated with few problems, which can make analytical applications hard to write and difficult to managed. I identify some of the problems here and will propose a solution to them below.

Diversity in data location. Experimental data can be stored in diverse locations: they can span e.g. different files, folders, disks, or databases. The problem stands out vividly once there are data that should be dealt with in an uniform way (e.g. data recorded during several runs of the same experiment), but are stored in multiple places. A straightforward way of processing such data would be to hardcode the physical locations of the data in an application, or, worse, writing several

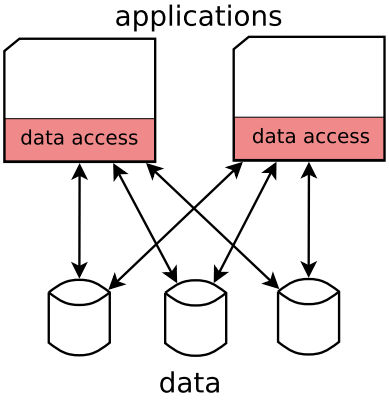

Figure 4.1: Applications that access data directly need to be aware of all the low-level details of data location and format, and the code accessing data must be duplicated.

applications, each of them working with data residing in different location. Badly enough, any change to data location would make the application(s) to be updated accordingly (Fig. 4.1)

Diversity in data format. Data, however semantically equivalent, can be stored using different data formats, e.g. in binary/text form or using different encodings. Applications accessing such data directly would need to be able to handle each and every data format separately, even though once they read the data in, they process all of them in the very same way.

Preprocessing required. Raw data, as provided by biomedical experts or acquired from a recording machine, can require some preprocessing to be performed before analysis can take place. For example, the data can require cleaning, filtering, aggregating, or conversion to be performed. Even though preprocessing should be done prior analysis, analytical applications can often be found being obfuscated performing preprocessing, which makes them clumsy.

In addition, there comes the problem of where to store the preprocessed data: shall the application store the data locally, such that it can be accessed later by the authoring application, or shall the data be propagated to some shared location and reused later by other applications as well?

Diversity in analytical applications. Typically, data gets analysed using not only one, but more applications, which study different aspects of the same data. Provided there are many applications operating over the same data, the problems described above would lead to an explosion of complexity, as each and every application would need to address the same problems separately. They all would need to be aware of where the data is, what format it is in, and how to preprocess it. Any change to the low-level details of where or how the data is stored would result in a need to rewrite all the applications. Moreover, it can lead to "copy&paste" errors, because programmers can be tempted to adapt one application and reuse (copy & paste) the change in another one, overwriting some application-specific code in the second application by code from the first one.

Also, a multiple exclusion problem can arise when multiple applications access the same data at the same time. Any attempts to alter the data simultaneously by several programs can, if not secured, result in data inconsistency or damage. If the data access is secured, but not correctly, deadlocks[1] can occur.

Also, preprocessing the same data by several applications simultaneously can waste computing resources.

[1] the situation in which two or more applications are waiting for each other to finish indefinitely, e.g. when application A1 holds write lock on file F1 and tries to acquire write lock on file F2, while application A2 holds the write lock on file F2 and tries to acquire the write lock on file F1

4.1.2 *Problem Solution: Making Data Access Abstract*

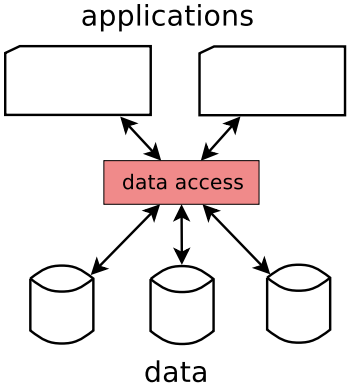

Figure 4.2: Applications that access data using a shared data-access component are simpler and the data access is more efficient and secure.

The solution to the data-access problems described above is to separate the code accessing data from the code operating over the data, and automate the data-accessing code as much as possible (Fig. 4.2). Such a separation comes with several advantages.

First, the code accessing data can be written once, and all the low-level data-specific details can be put together, which makes such code adaptable easily. Data access synchronization is also much simpler when performed from within a single piece of code.

Second, preprocessing can be performed automatically, on demand, at the moment when preprocessed data are requested. The preprocessed form of data can be cached easily for later reuse, such that any subsequent requests for the preprocessed data issued by any application would result in fetching the cached data without a need to perform preprocessing again. Synchronization and wasting of computing power is no longer an issue following this setup.

Third, applications become simpler, as they delegate data access to the specialized code. Applications do not need to be rewritten once data moves or its format changes.

4.1.3 *Identifying Data*

Once we delegated data access to a specialized component, analytical applications no longer need to be aware of all the low-level details of how and where the data is stored, but they still need to identify the data, i.e. to tell what data shall the data-access component provide. The identification of data should be simple, short, intuitive, and free of technical details. Good data identification should describe *what data* to get, not *where to find the data* or *how to access the data*.

declarative data identification

The idea of describing *what data to get* instead of explicitly programming the way *how to get the data* can resemble the distinction between logic and procedural programming: in logic programming, one describes *what* he/she seeks to achieve, whereas in procedural programming they explicitly describe *how* to reach a specific goal.

In the following paragraphs I describe the general design concept of the Data Access Object (DAO), a data-access component with declarative data identification, and illustrate this concept using implementation of the DAO in the MATLAB programming language in the context of neurophysiological data.

The DAO is a software component with extremely simple application programming interface (API) consisting of only one data getting method: `daoGet`. Even though there are also methods enabling to delegate data access to other components, and compute and cache data on the fly, those are internal (used by programmers implementing the

DAO API overview

DAO itself). First, I describe the DAO from the user's point of view: I describe the `daoGet` operation. Next, I portray the other methods, and more advanced usage of the DAO: stacking DAOs in a hierarchy, and data iterators.

4.1.4 Basic Use of the DAO: Getting Data

The typical usage of the DAO consists of asking the DAO to retrieve some data by calling the `daoGet` operation. A user supplies a `daoDataSelector`, a data-identifying record, and the DAO returns the data requested accompanied with a `daoDataDescriptor`, a data-describing record (Fig. 4.3).

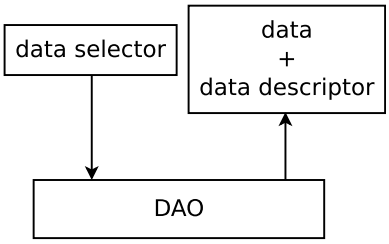

Figure 4.3: Basic use of the DAO.

Because data are identified in a declarative, descriptive way, the identification (of what data to get) and the description (of the data retrieved) overlap, such that `daoDataSelector` and `daoDataDescriptor` share the same fields. The `daoDataDescriptor` has usually all the fields set on return from `daoGet`, such that the data returned are described in a comprehensive way. In contrast, there is no need to set all fields in the `daoDataSelector` when asking for data - fields that uniquely identify the data are sufficient. In fact, the `daoDataSelector` may not even support all the fields of the `daoDataDescriptor`.

To illustrate the concept of the declarative data identification and description, below are listed fields of `daoDataSelector` and `daoDataDescriptor` as designed to identify and describe single-neuron recordings obtained using several microelectrodes in several brain areas from several patients during affective stimulation in terms of the scanning eye movement task[2]. In brief, the task consisted of several *experiment parts*, each of whose was composed of several *time frames*. Fields marked with the * tag are not supported in `daoDataSelector`.

[2] see page 71 for details

- `patientId` - numeric unique patient ID
- `patientName`* - string description of patient
- `side` - brain hemisphere (`dex` (right)/`sin` (left))
- `position` - recording position (`p1`, `p2`, `p3`, etc.)
- `area`* - brain area (`STN` (Subthalamic Nucleus), `SNr` (Substantia Nigra, pars Reticulata), `Hyp` (Hypothalamus), etc.) from which data was recorded
- `electrode` - electrode used to record given data
- `experiment` - scanning eye movement experiment presented
- `experimentPart`[3] - experiment part, usually a number between 1 and 24

[3] this is incompatible with the `category` field

- `timeFrame` - time frame of experiment part (`baseline`, `stimulation`, or `post`; or `whole` standing for the union of all the individual time frames)
- `category`*[4] - IAPS[5] picture category (this field can be used to filter specific experiment parts)

[4] this is incompatible with the `experimentPart` field

[5] International Affective Picture System

Data-retrieval MATLAB code typically looks like the following:

```
% create an empty data selector
dataSel = daoDataSelector();

% fill-in the fields of the data selector
% dataSel.XXX=xxx;
% dataSel.YYY=yyy;

% get the data
data = daoGet(dataSel, dataType);
```

For example, to retrieve *spikes* data recorded using the 3^{rd} electrode in the 2^{nd} recording position in patient 1, one can write the following:

```
% get access to global configuration
global dbs;

% create an empty  data selector
dataSel = daoDataSelector();

% request data for IAPS patient of ID 1, ...
dataSel.patientId=dbs.iapsPatients{1}.id;
% ... the 2nd position ...
dataSel.position=dbs.iapsPatients{1}.positions{2};
% ... the 3rd electrode available
dataSel.electrode=dbs.iapsPatients{1}.electrodes{2}(3);

% get the spikes data
data = daoGet(dataSel, spikesDataType);

% The "data" is a single daoDataDescriptor (or an array
% of daoDataDescriptor-s in general) containing,
% in addition to regular data descriptor fields,
% the "data" field holding the requested spikes data.
```

4.1.5 *The DAO API*

To a regular user, the DAO seems to offer a single operation only: `daoGet`. Indeed, this is the only public DAO operation you need to get your data. However, internally, the `daoGet` operation is broken into several steps, which make the DAO design-efficient, configurable, easily manageable and stackable. Thus, the DAO API is much richer

(Fig. 4.4) and consists of the following operations:

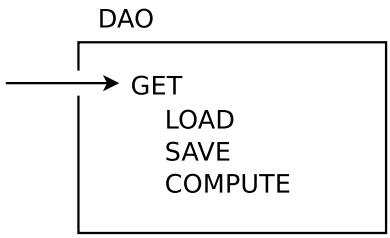

Figure 4.4: The DAO API offering one public method (`daoGet`) and a few internal functions.

- `daoGet` - get data. `daoGet` is the only operation visible to a regular user. However, it is implemented in terms of the other operations.
- `daoLoad` - load data, if the data is available. This operation gets called automatically by the `daoGet` operation in order to try to obtain already cached data, if any.
- `daoCompute` - compute data. This operation gets called by the `daoGet` operation if the `daoLoad` operation returned no data.
- `daoSave` - save data. This operation gets called by `daoGet` to save the data computed by the `daoCompute` operation.

The implementation of the `daoGet` operation first tries to *load* the requested data by calling the `daoLoad` operation. If the data is available, it gets returned to the caller. Otherwise, `daoGet` calls the `daoCompute` operation to *compute* the data, and *saves* the computed data using the `daoSave` operation (Fig. 4.5).

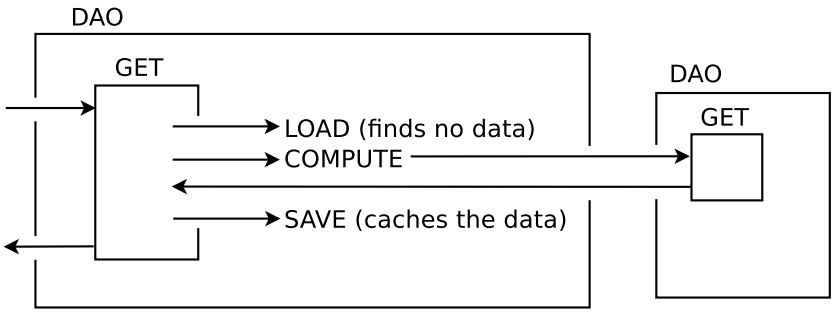

Figure 4.5: A call to the `daoGet` operation when the date requested has not been cached already. `daoGet` calls the `daoLoad` operation, which does not find the data. `daoGet` then calls `daoCompute` to compute the data on the fly. Typically, `daoCompute` delegates the call to another DAO (right) and transforms the obtained data into the requested form. The DAO then automatically saves the data using `daoSave` and returns it.

As shown in Fig. 4.5, the `daoCompute` can typically be implemented in terms of calling the `daoGet` operation of another DAO and processing the data to appear in the requested form, for example converting, transforming, or summarizing the data. This mechanism enables automated (pre)processing of data, in which programmer only needs to supply the implementation of the `daoCompute` operation and the DAO itself takes responsibility of automated caching of the data.

A subsequent request for the same data results in finding the data computed and saved previously (Fig. 4.6).

data storage abstraction

Note that developers implementing the `daoLoad` and `daoSave` operations can choose any data storage mechanism (a disk file, a database, or even a cloud) and any data format they like, because neither users nor the DAO access the data directly, but only through the use of the `daoLoad` and `daoSave` operation. It is, for example, possible to migrate the data without any change to users' code.

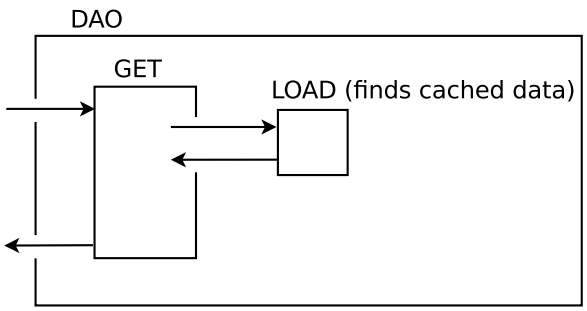

Figure 4.6: A call to the `daoGet` operation when the date requested has been cached already. `daoGet` calls the `daoLoad` operation, which finds the data, returns it, and `daoGet` returns it to the user.

automated data preprocessing

Automated preprocessing of data is enabled thanks to the workflow implemented by the `daoGet` operation. If the data requested is not available, the `daoCompute` operation gets called to compute it, and the resulting data gets cached automatically by the DAO. In the following paragraphs we will see how this feature can be used to ease processing of medical recordings.

4.1.6 Automated Access to Data at Different Processing Stages

processing stages

In medical sciences, researchers often need to handle the same data from a number of different views. They need to access raw data to check for their consistency. They also need to preprocess the data to make it accessible for standard analytical tools. Often they also need to summarize the data. What do these tasks share in common? They all make use of the same data, but at different processing stages. If the process of transforming the data between the stages can be automated, the DAO can ease access to data at all the stages significantly.

In the context of single-neuron microelectrode recordings[6], programs typically operate upon both the raw recordings and spike trains derived from the raw data. In addition, summary statistics can also be useful. Since the identification of the data to be operated upon is the same regardless of the data type (e.g. data from a particular brain area in the second position recorded in patient 1), the DAO enables to access any type of data via a single `daoGet` operation taking two parameters: `daoDataSelector` and `dataType`. While `daoDataSelector` can be shared across all the calls to `daoGet`, user can request particular data types simply by supplying different `dataTypes` to `daoGet`.

[6] see Section2.2

In the context of single-neuron recordings, the DAO implementation supports four different data types:

- `rawSignalTextDataType` - raw data stored in text form
- `rawSignalDataType` - raw data stored in binary form
- `spikesDataType` - spike train data

- `statisticsDataType` - summary statistics

Initially, raw signals (as exported from the recording machine internal format) comes in an inefficient text form: the data is large to store and slow to load. A conversion to a more compact and faster-to-load binary representation is desired. The DAO can help to perform the conversion automatically. To do so, we define two DAO data types carrying raw data: `rawSignalTextDataType` and `rawSignalDataType`, which represent raw data in text, and binary form, respectively. The conversion is trivial, implemented by only a single line of code of the `daoCompute` operation in the DAO handling data of type `rawSignalDataType`. When data of type `rawSignalDataType` is requested for the first time (Fig. 4.5 left), it can not be found by `daoLoad`, and `daoGet` calls the `daoCompute` operation to compute the data. `daoCompute` simply delegates the call to the DAO handling data of the `rawSignalTextDataType` type (Fig. 4.5 right) and returns it, as there is no need to perform any data transformation, because the memory representation of the raw data of both types are the same (they differ only in terms of the storage format). Finally, the `rawSignalDataType` DAO saves the data in binary format and returns it to the user. The conversion is thus performed on the fly, automatically, and happens as a side-effect of getting data of type `rawSignalDataType` for the first time.

text to binary raw data conversion

Single-neuron recordings hold the aggregated activity of a few neurons near the recording electrode[7]. To study the behaviour of individual neurons, there is a need to unmix the aggregated activity by extracting spike trains, the traces of activity of individual neurons, from it. Here, I describe this process from data-centric point of view[8]. I define the `spikesDataType` data type and implement the `daoCompute` operation which will perform the extraction of spike trains from the raw single-neuron recordings. Similarly to the conversion of the single-neuron recordings from text to binary form, here the conversion of the raw signals to spikes is merely a side effect of getting data of type `spikesDataType`, which gets automatically delegated to getting raw data and calling the transform from within the `daoCompute` operation. Again, the extracted spike trains (data of type `spikesDataType`) get cached automatically by the DAO for later use.

[7] see Section 2.2

[8] Section 3.1 discusses the extraction process in detail

conversion of raw data into spikes

Last, I define the `statisticsDataType` data type, which represents a brief summary view of the spikes data. Again, the first request for data of the `statisticsDataType` type results in getting data of the `spikesDataType` type, computing the summary statistics, and saving them for later use automatically.

The data types described above form a hierarchy of DAOs (tab. 4.1). Each layer of the hierarchy handles particular type of data and builds upon the data type of the underlying DAO. This stacking of DAOs

```
rawSignalTextDataType
        ↑
rawSignalDataType
        ↑
spikesDataType
        ↑
statisticsDataType
```

Table 4.1: The DAO data type hierarchy.

enables seamless automated conversion between data types, which comes with the advantages of making users' code simpler, shorter, and less fragile. This is because data access is limited to calling `daoGet` operations requesting particular data types, without any need to perform conversions explicitly, or to be aware where and in which format the data of different types is stored.

4.1.7 *Data Iterators*

Analysis of patients' single-neuron recordings typically involves requesting data recorded from individual patients in several recording positions using several electrodes, and possibly traversing individual parts of the experiment executed there. The following pseudocode illustrates such an analysis, which programmers can found to be too demanding and unnecessary, since, from their perspective, the only important operation is the `doSomething` function taking some useful actions, while the rest is just burden necessary to get the desired data:

```
dataSel = daoDataSelector();

for patientIdx = 1:length(dbs.iapsPatients)
  patient = dbs.iapsPatients{patientIdx};
  dataSel.patientId = patient.id;

  positions = patient.positions
  for posIdx = 1:length(positions)
    dataSel.position = positions{posIdx};

    electrodes = patient.electrodes{posIdx}
    for electrodeIdx = 1:length(electrodes)
      dataSel.electrode = electrodes(electrodeIdx)

      experimentParts = patient.experimentParts(posIdx)
      for experimentPart = 1:experimentParts

        % do something with data identified by "dataSel":
        data = daoGet(dataSel, spikesDataType);
        ...

      end
    end
  end
end
```

To overcome the burden and simplify the analysis, the DAO features so called `dbsDataIterator`, which iterates over all (or some of) data and invokes a specified user-supplied callback function on each iteration. The callback function stands for the `doSomething` function shown in the code listing above, at the inner-most level. The code shown above then simplifies to defining the callback function and data

data iterator

over which to iterate, as the DAO takes responsibility for automatically executing the callback over the data requested.

`dbsDataIterator` can iterate over:

- **P**atients
- **P**ositions
- **E**lectrodes
- **N**eurons
- **E**xperiment parts
- filtered **e**xperiment parts

The bold letters (concatenated) define the desired data to iterate over. For example, to iterate over over patients, positions, electrodes and episodes, you can use the **PPEE** pattern.

The code shown above thus simplifies to the following single line of code:

```
1 dbsDataIterator('PPEE', @doSomething);
```

callback-specific data

Usually, there is a need to share some data between the caller and the callback function, such that the callback can aggregate some data over all its invocations and return it to the caller. Thus, the `dbsDataIterator` takes an optional argument - the data to share with the callback. `dbsDataIterator` passes the data to the callback, which, in turn, is expected to alter the data and return it back to `dbsDataIterator`, which passes it to the callback again on the next iteration. Finally, `dbsDataIterator` returns the data back to the caller. An illustration of this technique is shown below.

conditional data iterator

`dbsDataIterator` also comes with a helper that enables processing of just data fulfilling some condition in an easy-to-use way, via a `dataSelector` data filter. An example of `dbsDataIterator` iterating over data recorded from the STN using a specific electrode follows. It also demonstrates how to return data from the callback and how to share callback-specific state between individual executions of the callback using callback-specific data.

```
% prepare data filter (data selector template)
dataSelTmpl = daoDataSelector();
% request data recorded using the central electrode only
dataSelTmpl.electrode = dbs.electrodeCentral;
% request STN area only
dataSelTmpl.area = dbs.areaFilterSTNFamily;

% prepare callback-specific data
cbData = [];

% Iterate over Patients, Positions and Electrodes.
% the cbData (passed through all the callback invocations)
% gets returned from the iterator.
cbData = dbsDataIterator('PPE', @callback, cbData, dataSelTmpl);
% "cbData" now holds the number of neurons detected
% on the central electrode in the STN across all patients.

% This is the callback called from within
% the dbsDataIterator. It receives two parameters:
%   cbData: the callback-specific data,
%   dataSel: daoDataSelector identifying patient's data.
% The callback operates upon the data, persists its
% internal state into "cbData" and returns "cbData",
% which will be passed back to the callback on the next
% invocation of the callback and, finally, returned
% from the dbsDataIterator to the caller.
function cbData = callback(cbData, dataSel)
    % get spike train from a specific patient and position
    % recorded on the central electrode in the STN
    spikes = daoGet(dataSel, spikesDataType());
    % compute the number of neurons in the spike train
    neuronCount = computeNeuronCount(spikes);
    % accumulate the number of neurons into "cbData"
    cbData = cbData + neuronCount;
end
```

To ease processing, the DAO also features `dbsDataIteratorOverEpisodes` and `dbsDataIteratorOverEpisodes2DS`[9] iterators enabling automated processing of data recorded during (some of) the episodes of a given single experiment.

[9] this iterator accepts two data selectors to ease processing of paired data

4.2 *Interactive Dendrograms as a Means of Exploratory Data Analysis*

In this section I describe an interactive data exploratory tool allowing to study the internal structure of data by inspecting the dendrogram resulting from the hierarchical cluster analysis of the data. The tool is implemented as an R[10] package called `idendro`. The tool enables the user to inspect dendrograms interactively: to select and color clusters, to zoom and pan the dendrogram, and to visualize the clustered data not only in a built-in heat map, but also in any interactive plot implemented in the `cranvas` package. The ability to inspect large dendrograms (consisting of tens of thousands of branches and more) interactively makes the tool an ideal candidate for data exploration in many diverse disciplines.

[10] R Core Team . *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL `http://www.R-project.org/`. ISBN 3-900051-07-0

The `idendro` R package is freely available at `https://github.com/tsieger/idendro`. This section is based on the documentation (R vignette) to the package, which has been submitted to the Journal of Statistical Software.

`idendro` has been used to explore high-dimensional data of single-neuron recordings, and also to guide informal evaluation of the accuracy of spike sorting methods. Irrelevantly to the thesis, `idendro` has also proven to be useful in exploratory analysis of flow-cytometry and spectroscopic data, as shown in the documentation.

The following text is structured as follows. Section 4.2.1 provides a brief introduction to hierarchical cluster analysis. Installation is covered in Section 4.2.2. Section 4.2.3 describes *idendro* invocation by way of simple examples. The graphic user interface (GUI) of `idendro` is described in Section 4.2.4 and its interactivity in Section 4.2.5. Section 4.2.6 is more technical, and discusses the data structures enabling the interaction between `idendro`, `cranvas`[11], and the user's code. Finally, Section 4.2.7 provides a case study demonstrating how `idendro` can be used to explore the wave forms of spikes detected in single-neuron recordings.

[11] Xie Y, Hofmann H, Cook D, Cheng X, Schloerke B, Vendettuoli M, Yin T, Wickham H, and Lawrence M. *cranvas: Interactive Statistical Graphics Based on Qt*, 2013. URL `http://github.com/ggobi/cranvas`. R package version 0.8.2

4.2.1 *Introduction*

Modern experiments often produce moderate- or high-dimensional data. These data can be challenging to explore, as one usually needs to consider information from all dimensions simultaneously. Hierarchical cluster analysis (HCA) (for an overview, see Hastie et al. (2009)[12], section 14.3.12) is a valuable tool that can give an insight into the structure of such data. In brief, HCA tries to reveal the structure of data by modeling it in terms of a hierarchy of clusters of observations. HCA can follow an agglomerative (bottom up) approach, or a divisive (top

hierarchical cluster analysis

[12] Hastie T, Tibshirani R, and Friedman J. *The Elements of Statistical Learning (2nd ed.)*. New York: Springer-Verlag, 2009. ISBN 0-387-84857-6

down) approach. In the agglomerative approach, HCA initially considers each observation to form an elementary cluster, and then builds a hierarchy of clusters by iteratively merging the two most similar clusters into a new one, until there is just a single cluster comprising all the observations. Following the divisive approach, HCA starts from the cluster of all observations, and builds a hierarchy by iteratively splitting each cluster of at least two observations into two clusters. In both cases, HCA results in a hierarchy of clusters, which can be represented by a tree-like structure called a dendrogram. Given that we performed agglomerative (or divisive) HCA over n observations, the dendrogram consists of $n-1$ pairs of branches representing the $n-1$ merge (or split) operations. The height of each pair of branches represents the distance of the two subclusters merged (split) at each step. Without loss of generality, and in order to simplify the text, in the rest of this text I assume that agglomerative clustering was used.

dendrogram

A comprehensive study of the structure of data requires proper visualization and interactive inspection of the dendrogram. While plotting the whole dendrogram presents the overall structure of the data, any finer structure becomes visible only when focused on, by zooming in the dendrogram. Moreover, inspecting the dendrogram alone cannot tell which observations form particular clusters. Decorating elementary clusters (i.e., observations) with their labels can help. However, we may still want to know what the values of individual features of observations in particular clusters are. This can be resolved by plotting feature space projections of the data and linking them to the dendrogram.

While HCA can be performed easily in R (e.g., using the `hclust` function in the `stats` package (part of R), or the `agnes` or `diana` functions in the `cluster` package[13]; for an overview, see `http://cran.r-project.org/web/views/Cluster.html`), the set of tools enabling interactive dendrogram visualization is rather limited, especially when it comes to large data sets. The low-level dendrogram plotting and interaction functions `plot.dendrogram`, `plot.hclust` and `identify.hclust` are in the `stats` package, which, however, can satisfy basic needs only, as they offer limited interactivity.

[13] Maechler M, Rousseeuw P, Struyf A, Hubert M, and Hornik K. *cluster: Cluster Analysis Basics and Extensions*, 2012. URL `http://CRAN.R-project.org/package=cluster`. R package version 1.14.3

The following text describes the `idendro` package for R, an interactive dendrogram visualization and inspection tool. `idendro` enables the user to plot large dendrograms, which can be zoomed, panned and inspected interactively by selecting and coloring clusters anywhere in the dendrogram. Feature space projections of the data can be visualized in a built-in heat map, or also in any interactive plot implemented in the `cranvas` package (e.g., a scatter plot, or a parallel coordinate plot). Such plots can be used to visualize the data, to highlight observations forming selected clusters, and the user can also select (*brush*)

observations there and then look back in the dendrogram to determine what clusters contain the selected observations.

4.2.2 *Installation*

I decided to implement graphic functionality using the `qtbase`[14] and `qtpaint`[15] packages based on Qt (`http://qt-project.org/`), a cross-platform application and user interface framework. A comparison with alternative frameworks revealed that the R interfaces to Qt were more convenient for implementing fast interactive graphics due to their stability, speed, and rich features, including the ability to build a powerful GUI. In addition, the use of `qtbase` and `qtpaint` enabled seamless integration with interactive plots from the `cranvas` package, which also builds on top of `qtbase` and `qtpaint`. On the other hand, the support for `qtbase` and `qtpaint` was (at the time of writing) limited on the Windows platform (see below).

[14] Lawrence M and Sarkar D. *qtbase: Interface between R and Qt*, 2012a. URL `http://CRAN.R-project.org/package=qtbase`. R package version 1.0.4

[15] Lawrence M and Sarkar D. *qtpaint: Qt-based Painting Infrastructure*, 2012b. URL `http://CRAN.R-project.org/package=qtpaint`. R package version 0.9.0

Prior to installing `idendro`, its prerequisites must be installed, namely the `qtbase`, `qtpaint` and `cranvas` packages. The availability of these packages depended on the operating system that was to be used. On linux systems, they could be installed quite easily. However, we experienced trouble with NVIDIA Optimus graphics ("errors linking simple shader"), which could be resolved by installing bumblebee (`https://github.com/Bumblebee-Project/Bumblebee`). On Mac OS X, manual installation was needed. On Windows, challenging manual installation involving getting additional libraries and software and building from sources led to successful results for some users. Readers are referred to the `cranvas` installation instructions at `https://github.com/ggobi/cranvas/wiki` to learn the current status.

Provided you have the `qtbase`, `qtpaint`, and `cranvas` packages installed, you can install `idendro` from `https://github.com/tsieger/idendro` using the `devtools` package[16]:

[16] Wickham H and Chang W. *devtools: Tools to Make Developing R Code Easier*, 2012. URL `http://CRAN.R-project.org/package=devtools`. R package version 0.8

```
library("devtools")
install_github("idendro", "tsieger")
```

Note: `idendro` should appear on The Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/` if the packages it depends on are there.

4.2.3 *`idendro` Invocation*

Let us demonstrate the `idendro` functionality on the *iris* data set[17] available from the `datasets` R package. The data set consists of 150 observations of Iris flowers, 50 observations for each of *Setosa*, *Versi-*

[17] Fisher R. A. The use of multiple measurements in axonomic problems. *The Annals of Eugenics*, 7(2):179–188, 1936

color, and *Virginica* species. For each flower, the *sepal length*, *sepal width*, *petal length* and *petal width* were measured (in centimeters) and stored in the first four columns of the data set. The species indicator (coded as an R `factor`) comes in the fifth column.

First, we try to identify clusters (i.e., subgroups of flowers) in the data by performing agglomerative hierarchical clustering over the measurements, using the `hclust` function in the `stats` package:

```
hc <- hclust(dist(iris[, 1:4]))
```

To visualize `hc`, the resulting hierarchy of clusters represented by a dendrogram, we can simply pass the return value of `hclust`[18] to `idendro`:

```
idendro(hc)
```

We get an interactive dendrogram drawn (not shown). This can be zoomed and panned, and clusters can be selected in it. However, we cannot see what flowers constitute the individual clusters. We therefore pass the *iris* data set as the second argument to `idendro`, which, by default, enables a heat map to be drawn next to the dendrogram and the names of the observations to be displayed next to the heat map:

```
idendro(hc, iris)
```

Now, we get the dendrogram drawn with a heat map attached to it (Fig. 4.7[19]). This plot gives a quite reasonable insight into which observations form which clusters. For example, we can see that the top most cluster (colored in green) includes flowers with short petals.

If we want to visualize data and clusters in other feature space projections, we can make use of any `cranvas` interactive plot, e.g., a scatter plot, by passing the `idendro` return value[20] to the `qscatter` `data` argument:

```
mdf.iris <- idendro(hc, iris)
# 'mdf.iris' holds a mutable data frame
print(qscatter(Sepal.Length, Sepal.Width, data = mdf.iris))
```

Now, we can enjoy the bidirectional integration of `idendro` with

[18] We could also use return values of other HCA functions, provided they are convertible to class `hclust` by the `as.hclust` function. Also, we could optimize the dendrogram using the `dser` function from `DendSer` (Hurley and Earle, 2013), as shown in the `idendroDendSer` demo.

[19] which will be discussed in full detail in Section 4.2.4

[20] This return value holds a so-called *mutable data frame*, the original *iris* data frame enriched with special hidden attributes over which the dendrogram and the other interactive plot can communicate. See Section 4.2.6 to learn more.

`cranvas` (Fig. 4.8, left). The points in the scatter plot get automatically colored according to the currently selected clusters in the dendrogram. Moreover, the points *brushed* in the scatter plot can be directly tracked in a so-called *brushed map* in the dendrogram window, which we describe in the following section.

4.2.4 *`idendro` Window Description*

The code given above should result in the plot shown in Fig. 4.7. We can see a window consisting of two main components: a simple GUI on the left side and a dendrogram enriched with a heat map and a brushed map on the right side.

graphic user interface

The top part of the GUI is populated by three columns:

- the current cluster selector, a radio button group determining which cluster is the *current cluster*. The *current cluster* determines which color and ID will be associated with a cluster selected in the dendrogram,
- cluster-specific statistics telling how many observations out of the total number of observations fall into each cluster, and
- cluster-specific statistics telling how many observations out of the observations brushed currently fall into each cluster.

The number of clusters shown in the GUI can be controlled using the `maxClusterCount` argument to `idendro`. The colors of the clusters can be controlled using the `clusterColors` argument.

In the bottom part of the GUI there are buttons that give access to the zoom and cluster selection history, and radio buttons controlling the heat map smoothing mode.

dendrogram

On the right side, there is a dendrogram with a heat map and a brushed map attached to it. The dendrogram depicts the process of agglomerative HCA, in which, initially, there were 150 elementary clusters (individual Iris flowers). Iteratively, at each stage, two closest clusters got merged, continuing until there was just a single cluster comprising all the 150 observations. The dendrogram is thus formed by 149 pairs of branches, each pair representing one merge operation. The distance of the two clusters merged at a specific stage is called the *height* of the newly merged cluster, and can be read from the axis below the dendrogram. The first merge operation occurred at height close to 0, while the height of the last (the biggest) cluster is close to 7.

heat map

The heat map, which is attached to the right side of the dendrogram, consists of a row of five colored rectangles drawn next to each observation. The rectangles code graphically the four measurements made on each Iris flower (i.e., *sepal length*, *sepal width*, *petal length*,

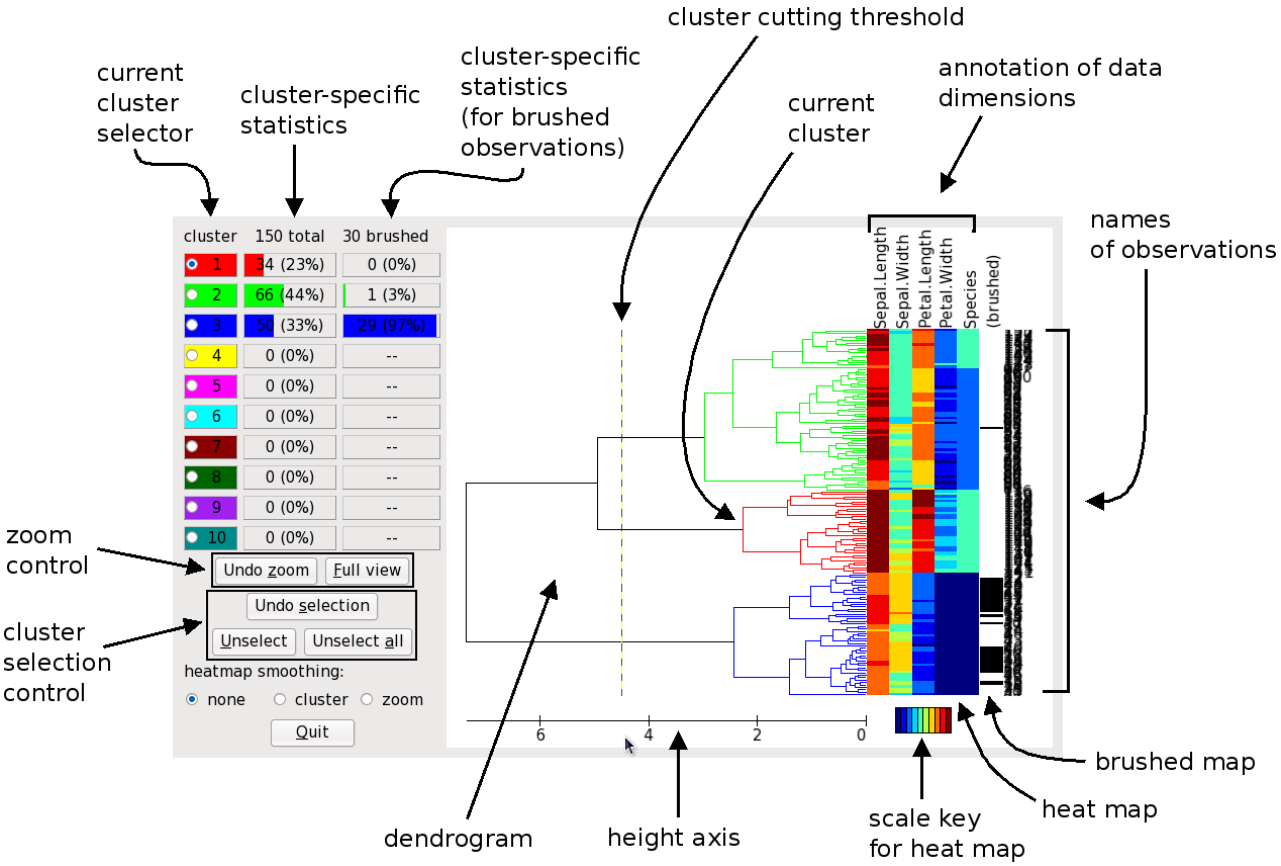

Figure 4.7: `idendro` window displaying *iris* data in terms of a dendrogram and a heat map.

petal width, as shown above the heat map) and the species of each flower[21]. The heat map colors are defined using the `heatmapColors` argument, which defaults to a list of 10 colors picked from the blue-green-yellow-red color spectrum, but any color spectrum can be used, e.g., `brewer.pal` from the `RColorBrewer` package[22], or `gray.colors`, `rainbbow`, `heat.colors`, `terrain.colors`, `topo.colors`, or `cm.colors` from the `grDevices` package (part of R). The heat map appearance is controlled using the `heatmapEnabled` argument, and is enabled by default, provided data was passed to `idendro`. The relative size of the heat map can be controlled using the `heatmapRelSize` argument, which determines how much space is reserved for the heat map out of the space reserved for both the dendrogram and the heat map. The default is 0.2, i.e., the heat map takes 20% and the dendrogram 80% of the space. For example, to use the gray color scale of 25 shades of gray in a heat map enlarged to 50% of space, `idendro` can be invoked by:

```
idendro(hc, iris, heatmapColors = gray.colors(25),
    heatmapRelSize = .5)
```

[21] the species were coded as a `factor` in the data set and got converted to a numeric type by idendro internally, such that the species can be included in the heat map

[22] Neuwirth E. *RColorBrewer: ColorBrewer Palettes*, 2011. URL `http://CRAN.R-project.org/package=RColorBrewer`. R package version 1.0-5

brushed map

The brushed map, displayed immediately next to the heat map, is formed by black/white rectangles, indicating whether the corresponding observation is/is not currently brushed in plots integrated with the dendrogram (not shown). The brushed map is enabled, by default, provided data was passed to `idendro`. Brushed map visibility can be controlled using the `brushedmapEnabled` argument.

names of observations

The names of individual observations are displayed on the right side of the `idendro` window. They are unreadable in Fig. 4.7, but will become clear once we zoom-in the dendrogram. The appearance of the names of the observation can be controlled using the `observationAnnotationEnabled` argument, which is `TRUE`, by default.

4.2.5 *Interacting with `idendro`*

`idendro` enables us to select a few clusters in the dendrogram, label and color them, and provide simple summary statistics for them. Initially, there are no clusters selected in the dendrogram[23]. To select a cluster, we can either click on a cluster in the dendrogram (on their top-level branch), or *cut* the dendrogram at a specified height.

[23] unless you pass a *mutable data frame* holding cluster selection metadata to `idendro` - see Section 4.2.6 for details

manual cluster selection

To select a cluster in the dendrogram manually, we simply click on the top-level branch of the cluster. The cluster gets colored according to the color of the *current cluster* selected in the GUI, and associated with the ID of the *current cluster*. Initially, the *current cluster* is the first one, which is colored in red, by default. To associate another dendrogram cluster with the *current cluster*, we can simply click on that cluster in the dendrogram, which results in unselecting the previous cluster and selecting the new one. To select some other cluster while keeping the first one selected, we simply change the *current cluster* in the current cluster selector in the GUI and pick another cluster from the dendrogram.

dendrogram cutting

We can also select clusters by *cutting* the dendrogram at a specified height threshold, i.e., select all clusters merged at or below the specified threshold. To *cut* the dendrogram, we move the mouse pointer below the dendrogram axis (this results in displaying the *cutting* threshold across the dendrogram, see Fig. 4.7) and press the left mouse button. `idendro` selects all the clusters merged at or below the specified height, and associates them with the first few clusters in the GUI. This is what we see in Fig. 4.7, in which we *cut* the dendrogram at a height of about 3.7, and obtained three selected clusters (red, green, and blue clusters consisting of 28, 50, and 72 observations, respectively).

We can see that these three selected clusters do not reflect the nature of the data, since there were 50 flowers of each species, but the

clusters consist of 34, 66, and 50 flowers, respectively. However, we can ask to what extent the clusters reflect the natural structure of the data. Luckily, the heat map can help to answer this question. The last column of the heat map shows the species of individual flowers, coded numerically (coerced from the levels of the *Species* factor). We can see that the second (as shown in GUI) cluster (colored in green), which consists of 50 flowers, matches the flowers of the first species perfectly. The first cluster (colored in red) matches the second species almost perfectly - there is only one misclassified flower in this cluster. The third cluster, however, seems to be a mixture of flowers of the second and the third species, though the subclusters of this cluster seem to reflect the structure of the data quite well.

unselecting clusters

To unselect the *current cluster*, i.e., to dissociate the *current cluster* shown in GUI from any cluster shown in the dendrogram, we can click the `"Unselect"` button. The `"Unselect all"` button can be used to unselect all clusters. The selection history is available - the previous selection can be recalled using the `"Undo selection"` button.

Dendrogram inspection usually involves iteratively focusing on clusters at different heights in different parts of the dendrogram. For example, we might want to study the internal structure of one of a few top-level clusters, taking a deeper and deeper look into it iteratively. `idendro` enables such inspection by zooming and panning the dendrogram.

zooming

To zoom in the dendrogram, we can either define a region to zoom to explicitly, using right mouse click and drag, or using the mouse wheel. In the latter case, the amount of zoom can be controlled using the `zoomFactor` argument.

To restore the original dendrogram view (i.e., to zoom out maximally), we can click the `"Full view"` button. The zoom history can be recalled by clicking the `"Undo zoom"` button.

panning

The dendrogram can be panned using mouse drag.

4.2.6 *Mutaframes: Data Structures for Dynamic Integration*

In this section I depict *mutable data frames*, or *mutaframes* - the data structure provided by the `plumbr` package[24] that enables the integration of interactive plots of `idendro` and `cranvas` with each other, and also with the user's code.

[24] Lawrence M and Wickham H. *plumbr: Mutable and Dynamic Data Models*, 2012. URL `http://CRAN.R-project.org/package=plumbr`. R package version 0.6.6

integration with other interactive plots

We keep in mind that it hardly suffices to look at a dendrogram and a heat map alone to learn what data tells us. We usually need to explore more feature space projections of the data. Hopefully, thanks to the effort made by the authors of the `cranvas` package, `idendro` can be bidirectionally integrated with modern high-speed interactive `cranvas` plots (Fig. 4.8). `idendro` automatically highlights observations

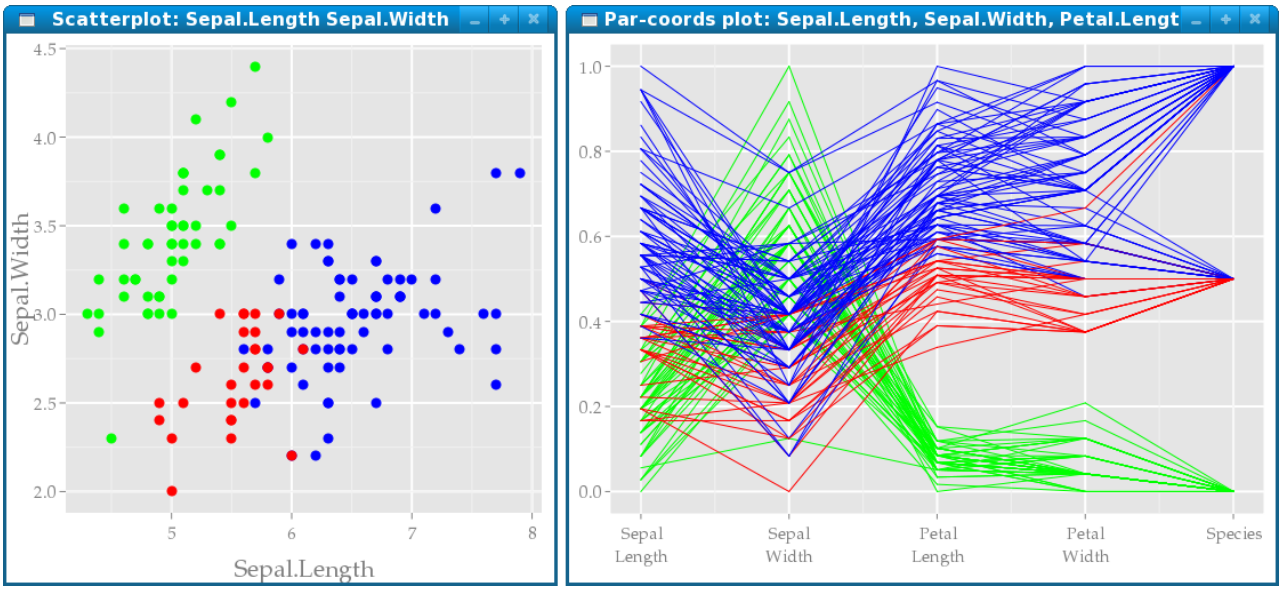

Figure 4.8: Interactive `cranvas` plots integrated with `idendro`. The scatter plot (left) and the parallel coordinate plot (right) display measures made on the iris flowers, reflecting the color of clusters selected in the dendrogram (Fig. 4.7).

forming the currently selected clusters in these plots. Moreover, selecting (*brushing*) observations in these plots propagates instantly into the brushed map, which then shows what clusters contain the selected observations (Fig. 4.7).

Technically speaking, the integration of `idendro` with `cranvas` interactive plots is enabled thanks to the concept of *mutable data frames* implemented in the `plumbr` package. In brief, *mutable data frames* are data frames enriched with hidden metadata (special columns in the data frame) that can be read, written and applications/users can also listen to changes being made to them. `cranvas` defines metadata controlling the color (`.color`, `.border`), size (`.size`) and visibility (`.visible`) of observations in plots. It also defines the `.brushed` metadata, which controls whether a given observation is brushed currently.

Mutable data frames can be explicitly constructed using the `qdata` function in `cranvas`:

```
mdf.iris <- qdata(iris)
# 'mdf.iris' holds a mutable data frame
```

Alternatively, `idendro` converts the data it gets into a *mutable data frame* automatically and returns it, such that you can get a mutable data frame as a side-effect of `idendro` invocation:

```
mdf.iris <- idendro(hc, iris)
% # 'mdf.iris' holds a mutable data frame
```

`idendro` alters the `.color` and `.border` metadata to color observations according to the color of the clusters they appear in, and listens to changes being made to the `.brushed` metadata to learn what observations are currently being brushed.

persisting cluster selection

When inspecting the dendrogram, we may wish to persist the clusters found so far, such that we will be able to get back to them later.

`idendro` introduces two more *mutable data frame* metadata in addition to those defined by `cranvas`: `.cluster` and `.inCurrentCluster`. For each observation, the `.cluster` holds the ID of the cluster that the observation is a member of (or 0, if the observation does not belong to any cluster). Similarly, the `.inCurrentCluster` metadata determines whether the given observation is a member of the *current cluster*.

The `.cluster` metadata can be used to persist the selected clusters in the dendrogram by simply saving the *mutable data frame* returned by `idendro`:

```
mdf.iris <- idendro(hc, iris)
```

To recall the persisted clusters, we can invoke `idendro` passing the saved *mutable data frame* as its second argument:

```
idendro(hc, mdf.iris)
```

The `.inCurrentCluster` metadata can be used by the user's code to compute information specific to the *current cluster* set in the GUI, as shown in the `idendroWithUserCallback.R` demo, in which we install a listener on the `mdf.iris` *mutaframe* and print the number and the mean *sepal length* of the observations in the current cluster whenever the cluster changes:

```
mdf.iris <- idendro(hc, iris)
my.listener <- add_listener(mdf.iris, function(i, j) {
  if (".inCurrentCluster" %in% j) {
    cat(sprintf(
      "The current cluster consists of %d observation(s).
      The mean sepal length is %.3f.\n",
      sum(mdf.iris$.inCurrentCluster),
      mean(mdf.iris$Sepal.Length[mdf.iris$.inCurrentCluster])))
  }
})
```

Note that the listener can be removed when not needed by:

```
remove_listener(mdf.iris , my.listener)
```

4.2.7 *Case Study: Exploration of Single-Neuron Recordings*

To demonstrate the `idendro` functionality, I show how `idendro` can be used to explore[25] the wave forms of spikes detected in a single-neuron recording[26] and interactively validate the results of spike sorting[27].

We start by reading data: we read a selected subset of 94 wave forms of spikes detected on a single microelectrode, and the assignment of these spikes to putative neurons as resulted from the spike sorting procedure. The wave forms are stored in a matrix, each row representing one wave form consisting of 64 signal samples located around the peak of each spike. Each wave form spans about 2.7 ms (the signal sampling frequency was 24 kHz). Next, we merge the wave form matrix with a one-column matrix of the spike assignment into a data frame. To identify wave forms emitted by a few putative neurons, we perform HCA over the wave forms. Finally, we display the result of the HCA in terms of an interactive dendrogram, a heat map depicting both the wave forms and the class assignment, and a parallel coordinate plot of the wave forms:

[25] other uses of `idendro` can be found in the package vignette and include exploration of flow-cytometry and spectroscopic data

[26] see Section 2.2

[27] see Section 3.1

```
library("idendro")

# read data
waveForms <- read.csv('waveForms.csv')
spikeSortingAssignment <- read.csv('spikeSortingAssignment.csv')
d <- cbind(waveForms, spikeSortingAssignment)

# compute HCA
h <- hclust(dist(waveForms), method = 'average')

# display an interactive dendrogram ...
qd <- idendro(hd, d, heatmapColors = gray.colors(20),
    heatmapRelSize = .7, clusterColors = c('red', 'blue'))
# ... and a parallel coordinate plot
print(qparallel(1:64, data = qd, scale = 'I'))
```

The code above results in Fig. 4.9. In the bottom window, there are the spike wave forms plotted in a parallel coordinate plot, indicating that there were probably two neurons detected on the recording microelectrode, as the wave forms seem to be of two kinds: wave forms of large and much smaller amplitudes. The separation stands out more vividly in the top window. On the right, there are the very same wave

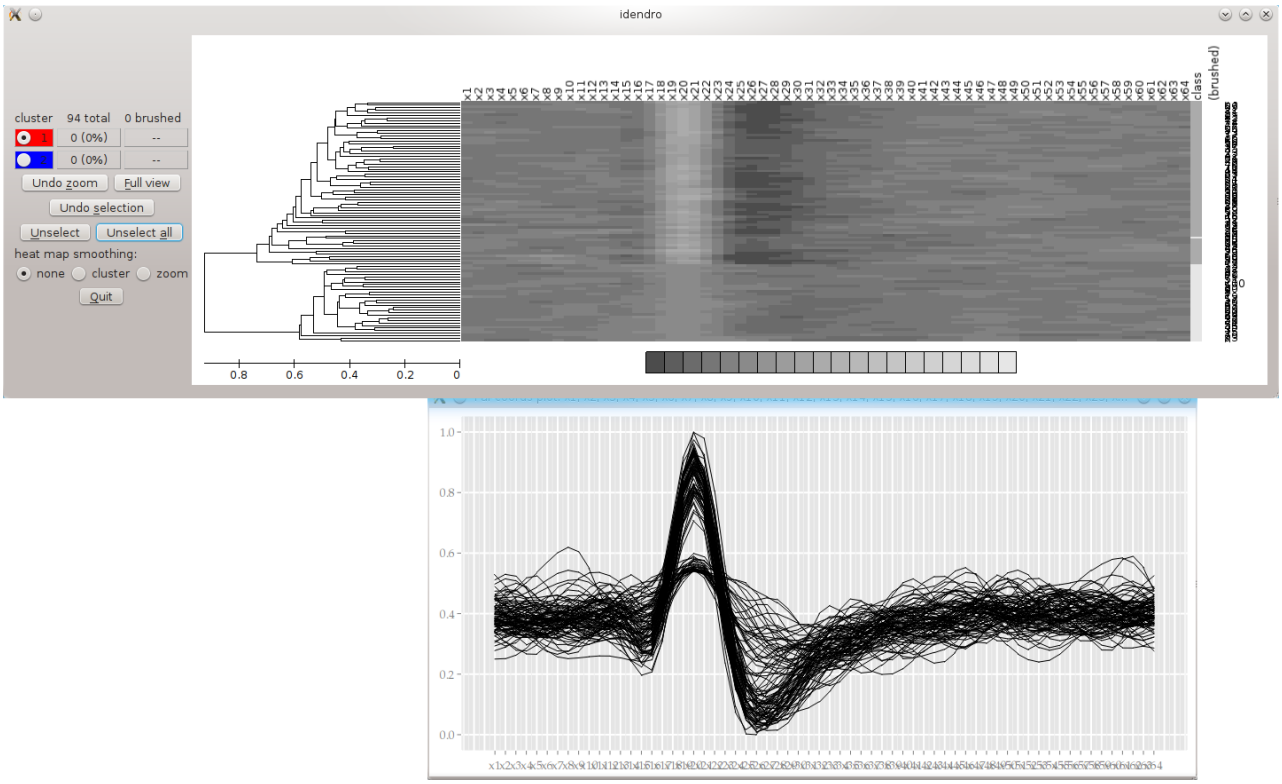

Figure 4.9: Interactive `idendro` dendrogram with a heat map (**top**) and `cranvas` parallel coordinate plot (**bottom**) depicting spike wave forms. The parallel coordinate plot has been manually aligned with the heat map to ease visual linkage of the plots. See text for details.

forms displayed in a heat map, in which the amplitudes are depicted in terms of shades of grey - the lighter the grey, the higher the amplitude. On the left to the heat map, there is a dendrogram displaying the data hierarchy as revealed by the HCA. The dendrogram communicates quite a convincing message of two well-separated and compact clusters of spike wave forms present in the data.

Thanks to the automatic dendrogram linking to the parallel coordinate plot, we can easily color the two clusters in the dendrogram (by cutting the dendrogram) and distinguish the wave forms forming the two clusters right in the convenient parallel coordinate plot using the same colors (Fig. 4.10). We can see that the clusters of spikes as revealed by the HCA agree with our apriori belief of two groups of spikes to be present in the data. From the GUI in the `idendro` window, we can learn that there are 30 (32%) spikes forming the red low-amplitude-spike cluster, and 64 (68%) spikes form the blue cluster of high-amplitude spikes.

The last column of the heat map (named "*class*") displays the assignment of the individual spikes to classes of putative neurons as computed by the spike sorting procedure. We can see that there is almost perfect agreement between the clusters revealed by the HCA

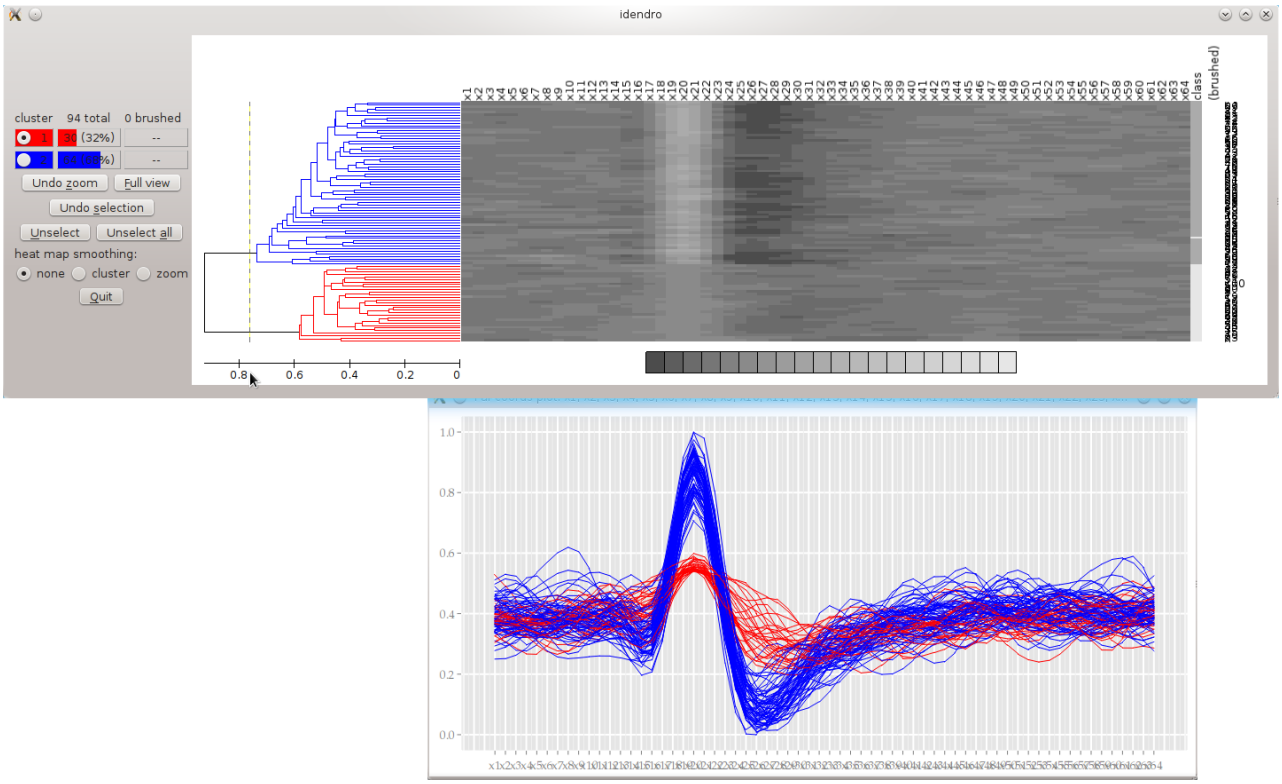

Figure 4.10: Interactive `idendro` dendrogram, heat map, and `cranvas` parallel coordinate plot depicting spike wave forms, having clusters colored by "cutting" the dendrogram, i.e. mouse clicking below the dendrogram height axis.

and the clusters identified by the spike sorting. There is only one wave form, in which the two assignments differ. By zooming in the dendrogram (Fig. 4.11), we can read the ID of this spike (#34) from the annotations right to the heat map, and study how close the raw wave form is to other wave forms in this cluster by inspecting the heat map. It seems that the height of the peak of the spike #34 is similar to other spikes in the cluster (see the heat map columns $x18$ to $x23$, and com-

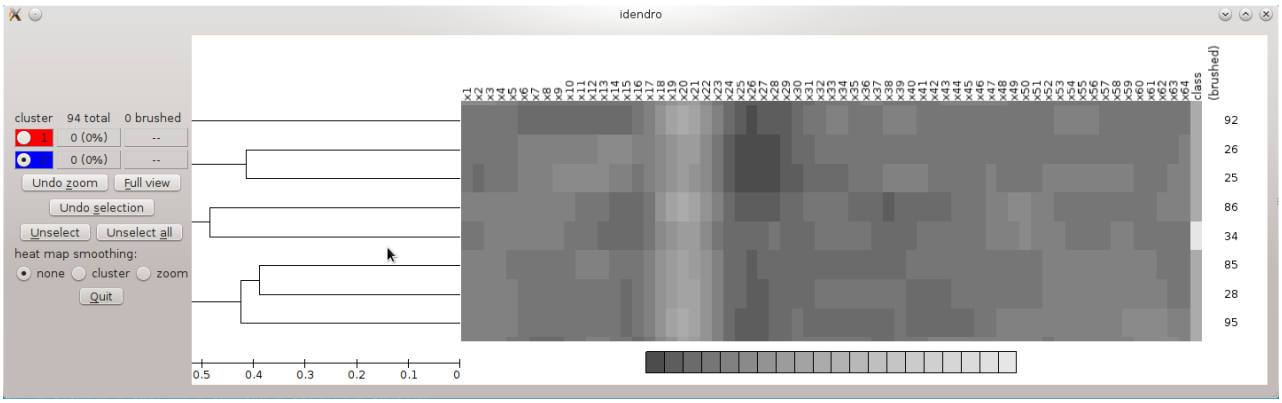

Figure 4.11: Zoomed `idendro` dendrogram of spike wave forms showing a misclassified spike #34.

pare the spike #34 with e.g. spikes #25 and #26). However, the decay of the wave form seem to be slower in spike #34 compared to spikes #25 and #26, which can be the reason why the spike sorting put spike #34 in the other cluster.

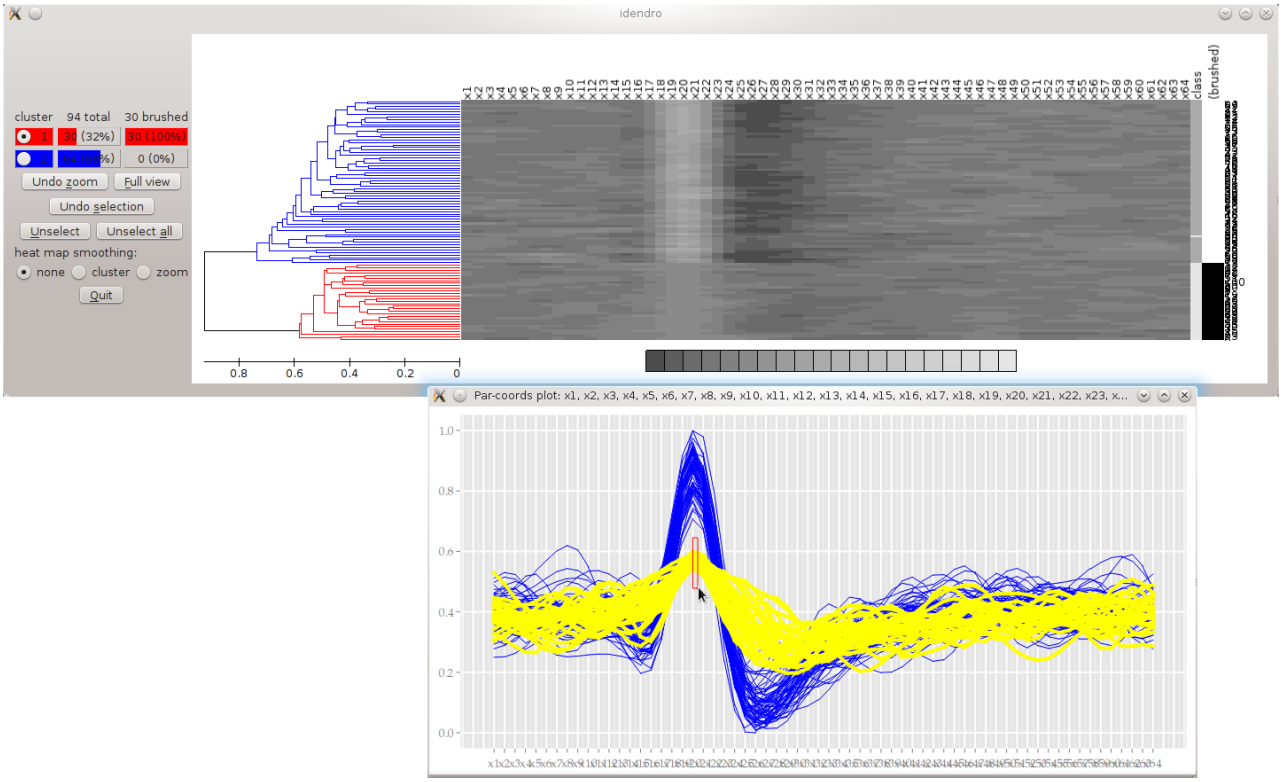

Figure 4.12: Interactive `idendro` dendrogram, heat map, and `cranvas` parallel coordinate plot depicting spike wave forms, with some wave forms brushed. The brushing area (red rectangle) highlights wave forms it coincides with (thick yellow). The brushing area can be moved by left mouse click & drag. The shape of the brushing area can be changed by right mouse click & drag.

The correspondence between the spike sorting result and the HCA-enabled structure of the data can also be studied by brushing the wave forms in the parallel coordinate plot (Fig. 4.12) and inspecting the brush map, i.e. the column right to the heat map (named "(*brushed*)"). While we keep brushing the wave forms in the parallel coordinate plot, the brush map gets colored automatically to show which spikes are being brushed currently. If we brush all the low-amplitude spikes, we can once again see the good correspondence between the result of the spike sorting and our apriori belief of the two groups of spikes in the data (Fig. 4.12). By inspecting the dendrogram together with the "*class*" column in the heat map and the "(*brushed*)" column in the brushed map, we can learn that the brushed low-amplitude spikes are exactly those contained in the red cluster, and that all the low-amplitude spikes were identified by the spike sorting as fired by a single putative neuron. However, as we've already seen above, the spike sorting put one more neuron (#34) in this group of spikes.

We can even identify the wave form of the spike #34. To do so, we first shrink the brushing area (by right mouse drag) to make it a small square, such that it can brush single wave forms. Then we keep brushing the wave forms until we find the wave form of the spike #34, as verified by the correspondence of the entries in the "*class*" column in the heat map and the brush map (Fig. 4.13). The figure shows that even though the amplitude of spike #34 is high, the post-peak part of the wave form (samples $x25$ to $x30$) are much higher than corresponding parts of the other spikes in the blue cluster. This might be the reason why the spike sorting procedure classified this spike as a low-amplitude spike. We can guess that the post-peak part of the wave form was contaminated with some other spike, as there seems to be a low-amplitude peak superimposed on top of the the post-peak wave form of spike #34 (Fig. 4.13 bottom, yellow thick curve).

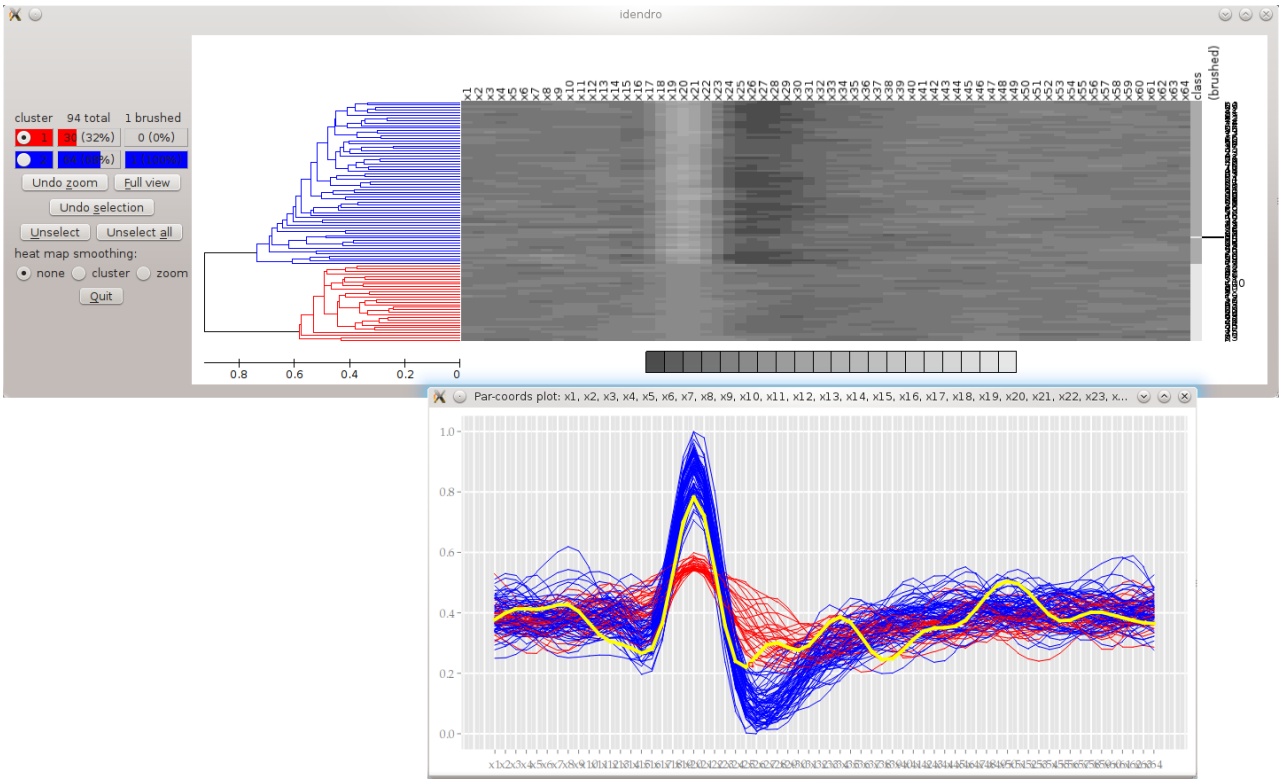

Figure 4.13: Interactive `idendro` dendrogram, heat map, and `cranvas` parallel coordinate plot depicting spike wave forms, with a brushed wave form, in which the HCA-based classification (cluster membership) differs from the spike sorting classification (the "*class*" column in the heat map).

4.2.8 *Conclusions*

`idendro`, a new R package enabling interactive dendrogram visualization and exploration has been introduced. To the best of my knowledge, this is the first package enabling really interactive exploration of large dendrograms in R. Moreover, the integration with interactive

plots provided by the `cranvas` package makes `idendro` a general data exploration tool.

However, at the time of writing, it was challenging to install this package on the Windows platform (due to the packages that it depended on).

4.3 Evaluation of Spike Sorting Methods

In Section 3.1 I described *spike detection and sorting*, i.e. the process of unmixing the activity of a neuronal ensemble into spike trains depicting the activity of individual neurons from the ensemble. However, there have been many such *spike detection and sorting* methods devised, and to the best of my knowledge, none of them can guarantee that the spikes they identify are exactly those emitted by the real neurons in the ensemble, and that they correctly distinguish the spikes fired by individual neurons from each other. We considered this to be a serious problem, and evaluated[28] the performance of a few widely used spike sorting methods.

Note: This work was a joint activity of a few authors, including the author of this thesis. However, the dominant contributor was Jiří Wild. We briefly sketch the work here for completeness. Details can be found in the original article.

[28] Wild J, Prekopcsák Z, **Sieger T**, Novák D, and Jech R. Performance comparison of extracellular spike sorting algorithms for single-channel recordings. *Journal of Neuroscience Methods*, 203(2):369–376, January 2012. ISSN 01650270. DOI: 10.1016/j.jneumeth.2011.10.013. *Contribution: 15%, citations: 1*

4.3.1 Introduction

There are many spike sorting algorithms available, e.g. (Quiroga et al., 2004; Rutishauser et al., 2006; Takahashi et al., 2003b; Franke et al., 2009; Fee et al., 1996, 1997; Cheeseman and Stutz, 1996; Takahashi et al., 2003a; Herbst et al., 2008; Delescluse and Pouzat, 2006; Cambridge Electronic Design Limited), but, in general, none of them can be regarded as optimal. There are spike sorting algorithms designed to cope well with a specific kind of signal, but they can fail to perform well on signals of different characteristics[29]. Other algorithms[30] require to be operated by an experienced user, which critically limits their use and makes their results subjective. Other methods can be too slow to be applicable to real-world problems. Last but not least, while most of the algorithms require a set of user-supplied parameters to be specified, the authors of the algorithms frequently do not specify how to set them, even though those parameters can drastically affect the performance of their methods.

Our task was to identify the most convenient spike sorting algorithm to process our single-channel single-neuron recordings.

[29] Takahashi S, Anzai Y, and Sakurai Y. Automatic sorting for multi-neuronal activity recorded with tetrodes in the presence of overlapping spikes. *Journal of neurophysiology*, 89(4):2245–2258, 2003b

[30] Cambridge Electronic Design Limited . Spike2, version 6. URL `http://ced.co.uk/pru.shtml?spk6wglu.htm?id=6`

4.3.2 Methods

We assessed the performance of several most popular and most widely used spike sorting methods: `WaveClus`[31], `KlustaKwik`[32], and `OSort`[33]. All these methods proceed in two steps: they detect spikes first, and sort them subsequently. To sort spikes, `WaveClus` employs wavelet transform to extract features from individual spikes, and then uses

[31] Quiroga R. Q, Nadasdy Z, and Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004

[32] Harris K. D, Henze D. A, Csicsvari J, Hirase H, and Buzsáki G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of neurophysiology*, 84(1):401–414, 2000

[33] Rutishauser U, Schuman E. M, and Mamelak A. N. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of neuroscience methods*, 154(1-2):204–224, 2006

superparamagnetic clustering to sort spikes into classes of putative neurons. `KlustaKwik` employs principal component analysis to extract spike features, and uses the `AutoClass` clustering algorithm[34] to sort spikes. `OSort` does not extract features from the spikes, but uses whole spike shapes to sort spikes employing a template matching algorithm.

Because the problem of spike detection had already been addressed[35], we focused solely on the sorting phase by skipping the spike detection phase and presenting the same set of pre-detected spikes to all the methods.

[34] Cheeseman P and Stutz J. Bayesian classification (autoclass): theory and results. pages 153–180, 1996

[35] Lewicki M. S. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4), November 1998. ISSN 0954-898X

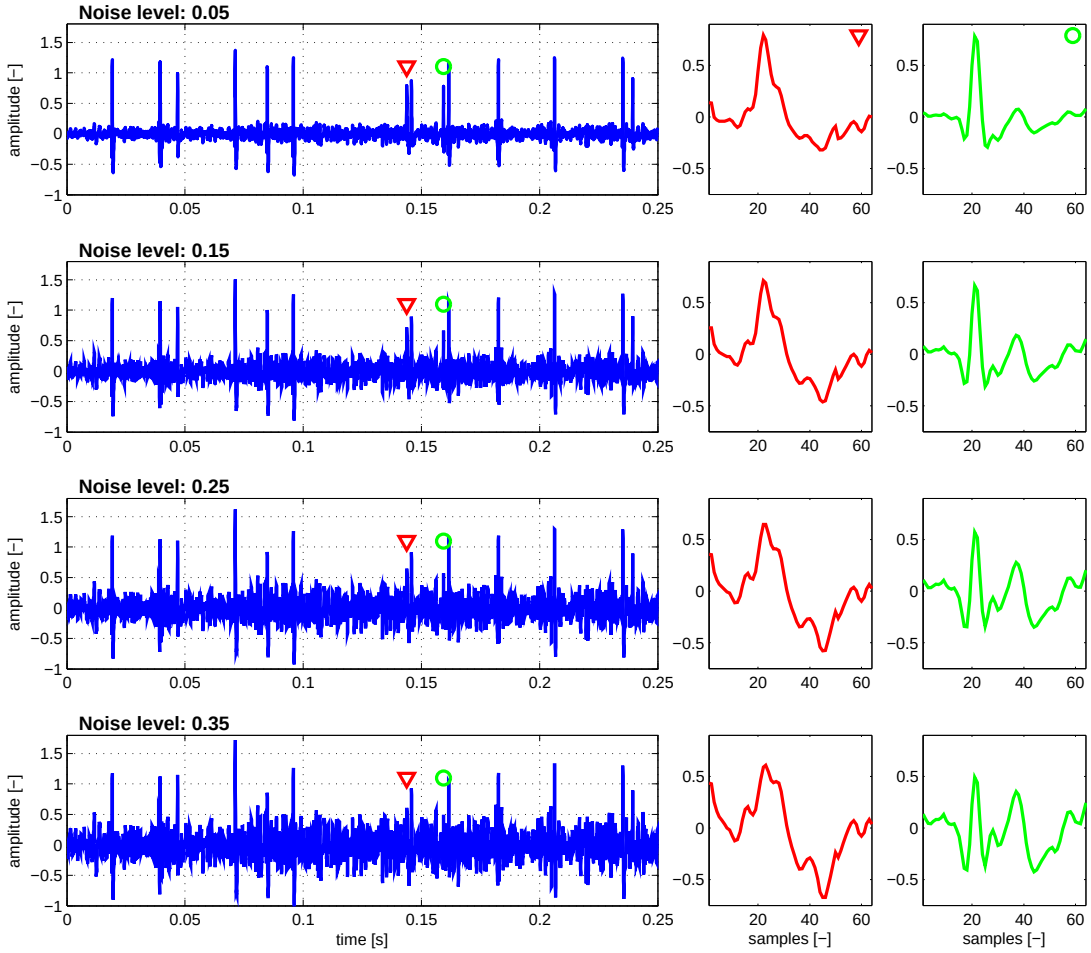

Figure 4.14: Example of the same 250 ms-long artificial single-neuron recordings with different noise levels ranging from 0.05 to 0.35. The spikes marked in the signal by a triangle and a circle each belonged to a different neuron and are shown in greater detail on the right side. *(Figure reproduced from Wild et al. (2012).)*

artificial signals

We proceeded by generating artificial single-neuron recordings, in which the *ground truth* was known, i.e. we knew how many neurons were active in each signal, and at which times each of the neurons fired. The artificial signals were generated by superimposing real spikes (as clearly stood out in our single-neuron recordings) on a random noise background. The noise itself was generated by superimposing a large number of 2,000 different low-amplitude spikes onto each other, simulating a noise formed by the independent contribution of a

large number of distant neurons. To study the influence of the background noise on the spike sorting performance, we generated artificial signals with different levels of background noise by adapting the amplitude of such "background" spikes. The noise level was defined as $N = \frac{P_{noise}}{P_{signal}}$, where P_{signal} represented the power (the mean square amplitude) calculated from spikes only, and P_{noise} represented the power of the rest of the signal. Fig. 4.14 depicts the same 250 ms long signal corrupted with four different noise levels (0.05, 0.15, 0.25, 0.35).

evaluation criterion

To rate the performance of the spike sorting methods, we presented them with sets of pre-detected spikes, and assessed their performance using the adjusted mutual information (AMI)[36], which calculated the correspondence between the ground truth (i.e. the real spikes) and the outcome of each of the methods. AMI is an information theoretic measure assessing the correspondence of two clusterings (i.e. assignments of spikes to neurons), providing a value between 0 (independent clusterings) and 1 (the same clusterings).

[36] Vinh N, Epps J, and Bailey J. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1073–1080, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. DOI: 10.1145/1553374.1553511

sensitivity to parameters

We also assessed how sensitive each spike sorting method was in respect to their parameters. We compared the performance obtained using the default parameters with the performance obtained using optimized parameters. To find the optimal parameters, an exhaustive search over the whole parameter space was performed on a training set of spikes. The performance was then evaluated on an independent testing set of spikes to overcome overfitting.

4.3.3 Results

First, we observed that the spike sorting methods evaluated were highly sensitive to their parameters. The use of optimized parameters led to better results in all the methods (Wilcoxon signed-rank test, $p < 0.01$), see Fig. 4.15.

Second, Fig. 4.16 shows that the performance of all the methods decreased with the increasing noise level, and that the performance of the individual spike sorting algorithms differed. While the median AMI performance reached by all the algorithms was about 0.7 for low (0 to 0.15) noise levels, the performance decreased dramatically to median AMI smaller than 0.2 for higher (0.3 to 0.45) noise levels. Moreover, we observed that `KlustaKwik` outperformed `OSort` (Wilcoxon signed-rank test, $p < 0.01$) when processing signals with low noise levels. In case of higher (0.15 to 0.3) noise levels, `WaveClus` outperformed both `KlustaKwik` and `OSort` (Wilcoxon signed-rank test, $p < 0.01$), reaching the median AMI of about 0.7 compared to the median AMI of about 0.5 (`KlustaKwik`) and 0.25 (`OSort`).

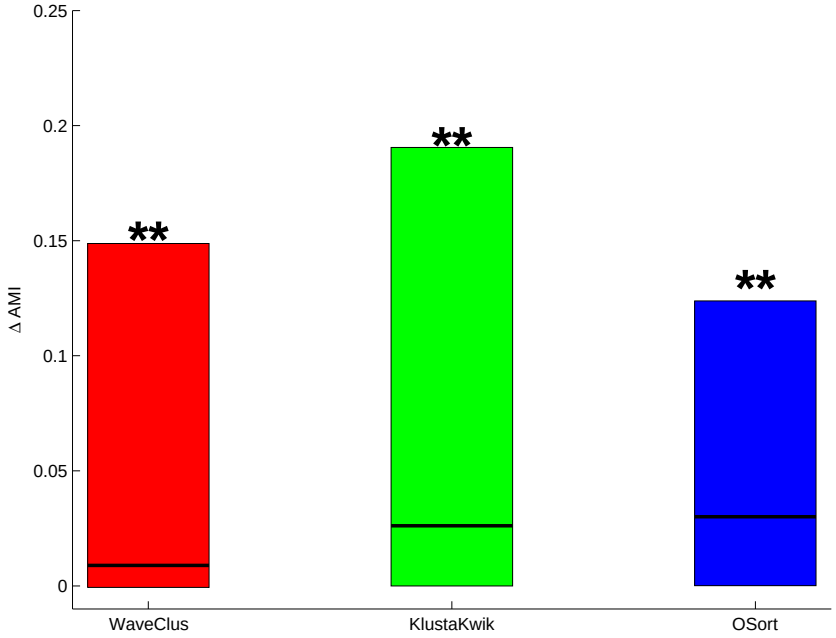

Figure 4.15: Comparison of the accuracy of the algorithms when used with default and optimized parameters. The y-axis represents the difference in the achieved AMI score between while using optimized parameters and while using default parameters, as evaluated on a number of artificial signals. The differences are summarized by box plots. Symbol ** indicates that the medians of the marked boxplots are significantly different from zero ($p < 0.01$). *(Figure reproduced from Wild et al. (2012).)*

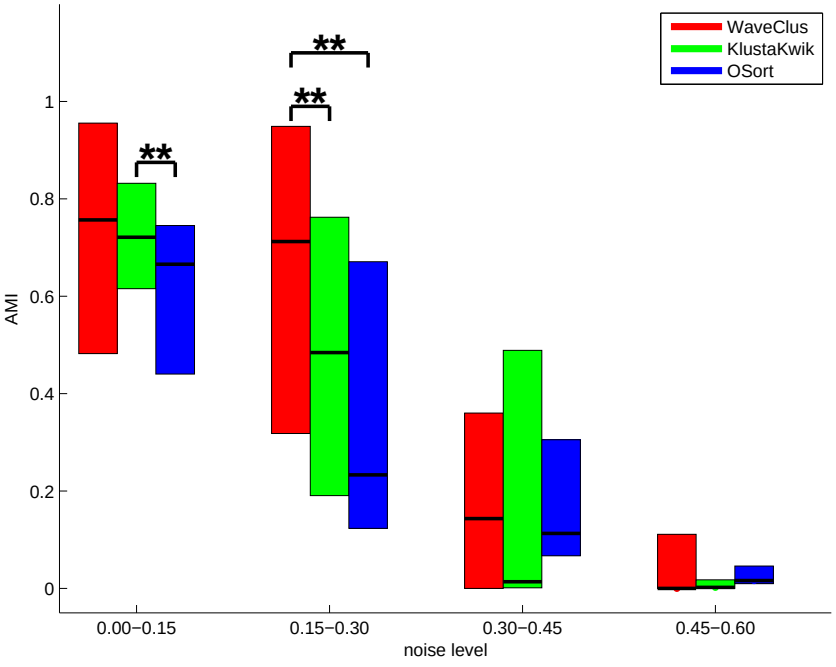

Figure 4.16: Performance of spike sorting algorithms on 30 s-long artificial signals, using optimized parameters. The y-axis represents the AMI score of each algorithm along with its spread. Symbol ** indicates that the medians of the marked boxplots are significantly different ($p < 0.01$ corrected for 3 comparisons). *(Figure reproduced from Wild et al. (2012).)*

4.3.4 *Conclusions*

Our evaluation showed that the performance of the individual spike sorting methods we tested differed (Fig. 4.16). The accuracy of all of them decreased with increasing level of background noise, which could be explained by the fact that all the methods had problems to distinguish spikes from each other when noise corrupted the shape of the spikes.

As we estimated the noise level in our real single-neuron recordings to lie in the range from 0.2 to 0.3, and the most accurate spike sorting method operating in this range was `WaveClus` (Fig. 4.16), `WaveClus` was used to process single-neuron recordings throughout this thesis. Because a proper choice of parameters to all the spike sorting methods was essential to reach high accuracy (Fig. 4.15), we did not use the default values of parameters to `WaveClus`, but used values empirically found to yield reasonable results. These parameters are listed in Tab. 4.2.

spike detection		
int_factor	4	spike shape interpolation degree
stdmin	3	multiplicative factor defining the min. amplitude of detected spikes relative to the estimate of the SD of the background noise
stdmax	30	multiplicative factor defining the max. amplitude of detected spikes relative to the estimate of the SD deviation of the background noise
ref	1.5	refractory period [ms]
spike sorting		
inputs	9	number of features used in clustering
min_clus_time_rel	1	minimum average number of spikes in 1 s of signal
scales	8	number of wavelet decomposition scales
mintemp	0.06	minimal temperature used in superparamagnetic clustering
maxtemp	0.15	maximal temperature used in superparamagnetic clustering
tempstep	0.01	temperature step size used in superparamagnetic clustering
num_temp	$\lfloor \frac{maxtemp-mintemp}{tempstep} \rfloor$	number of temperature levels used in superparamagnetic clustering
max_clus	13	maximum number of neurons detectable in single signal
template_sdnum	3	maximal radius of cluster in feature space (in SD units)

Table 4.2: `WaveClus` spike detection and sorting tool parameters found to give reasonable results on our single-neuron recordings and used throughout this thesis. SD: standard deviation.

5
Application Results

This chapter summarizes the answers to the neuroscientific questions raised. Section 5.1 describes our finding of basal ganglia neurons related to eye movements. Section 5.2 presents the process of searching for emotion-related neurons in the subthalamic nucleus. Section 5.3 identifies subthalamic neurons, whose activity differed between two specific tasks.

5.1 Eye Movement-Related Neurons in the Basal Ganglia

In this section I describe the way of searching for basal ganglia neurons whose activity was related to eye movements. I extensively use the DAO framework[1] to access data, employ the results of spike sorting evaluation[2] and apply bootstrap[3].

The content of this section has been published[4].

[1] see Section 4.1

[2] see Section 4.3

[3] see Section 3.2

[4] **Sieger T**, Cecilia B, Serranová T, Wild J, Novák D, Růžička F, Urgošík D, Růžička E, Gaymard B, and Jech R. Basal ganglia neuronal activity during scanning eye movements in Parkinson's disease. *PLoS ONE*, 8(11):e78581, 2013. DOI: 10.1371/journal.pone.0078581. *Contribution: 40%*

5.1.1 Introduction

The basal ganglia, in particular the subthalamic nucleus (STN), substantia nigra pars reticulata (SNr) and globus pallidus (GP) are known to play a role in the control of eye movement (EM) as reported by studies on animals[5] as well as on humans[6]. While neuronal activity was directly analysed in the animal studies, most of the human studies were limited to observing the effects of invasive treatments of the basal ganglia by analyzing EM parameters in oculographic experiments and studying saccades.

[5] Hikosaka et al. (2000); Matsumura et al. (1992)

[6] Fawcett et al. (2007, 2005)

The role of human basal ganglia in *scanning eye movements* has only received attention in studies focused on behavioral aspects of eye scanning path rather than to scanning EM control. However, there is evidence of association of the basal ganglia with scanning EM as reported by cerebral blood flow study in humans[7]. Therefore, we systematically searched for basal ganglia neurons participating in scanning EM.

[7] Tsunoda M, Kurachi M, Yuasa S, Kadono Y, Matsui M, and Shimizu A. Scanning eye movements in schizophrenic patients. Relationship to clinical symptoms and regional cerebral blood flow using 123I-IMP SPECT. *Schizophr. Res.*, 7(2):159–168, Jul 1992

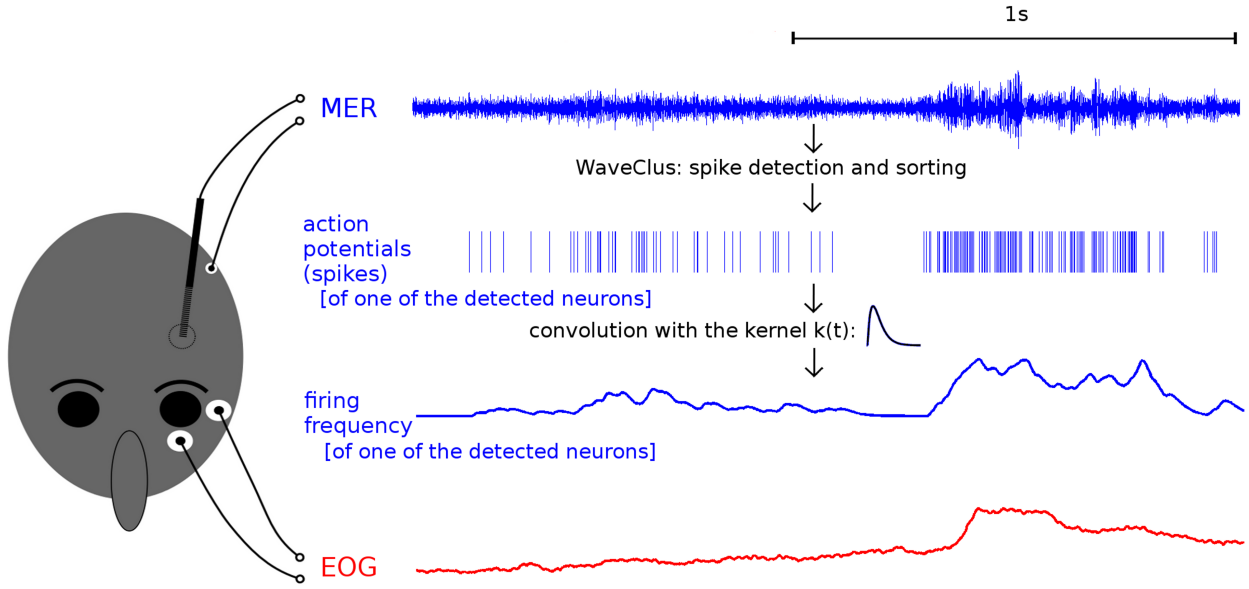

Figure 5.1: Single-neuron microelectrode recording (MER) and electrooculography (EOG) signal acquisition and processing. Action potentials of individual neurons were identified using WaveClus algorithm in the MER signal. The instantaneous firing rate (IFR) was then estimated by convolving a series of extracted action potentials generated by a single neuron with a causal kernel function. Finally, the IFR was correlated with the eye movement kinematic parameters derived from the EOG. *(Figure reproduced from Sieger et al. (2013).)*

5.1.2 *Methods*

We enrolled 19 Parkinson's disease patients treated with deep brain stimulation. Their single-neuron activity was mapped by single-channel microelectrode recordings (MER), which were band-passed filtered in the range of $500 - 5,000$ Hz and sampled at 24 kHz. `WaveClus`[8] spike detection and sorting tool[9] was used to extract the activity of single neurons from the MER signals.

In total, 183 neurons were detected in 137 MER signals. There were 130 neurons located in the STN, 23 in the GP and 30 in the SNr. In each neuron, the instantaneous firing rate (IFR) was then estimated by convolving the series of action potentials with the causal kernel function $k(t) = \alpha^2 t \exp(-\alpha t)$ defined for positive time t, where $1/\alpha$ was empirically set to 20 ms (Fig. 5.1).

Eye movements were recorded using one-channel electrooculography (EOG), a technique estimating the position of the eye from the electric potential induced by the eye dipole and recorded by a pair of surface electrodes attached near the outer canthus and the lower lid of the left eye (Fig. 5.1). The EOG signal was band-pass filtered in the range of $0.1 - 20$ Hz and recorded simultaneously with the MER acquisition. To reveal any potential association of neuronal activity not only with the eye position, but also with its motion or the dynamics of the motion, the EM was characterized by: i) the eye position (POS), defined to be the EOG signal itself, ii) the eye velocity (VELOC), defined to be the derivative of POS, and iii) the acceleration of the eye

[8] Quiroga R. Q, Nadasdy Z, and Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004

[9] see Section 4.3 for details and `WaveClus` parameters used

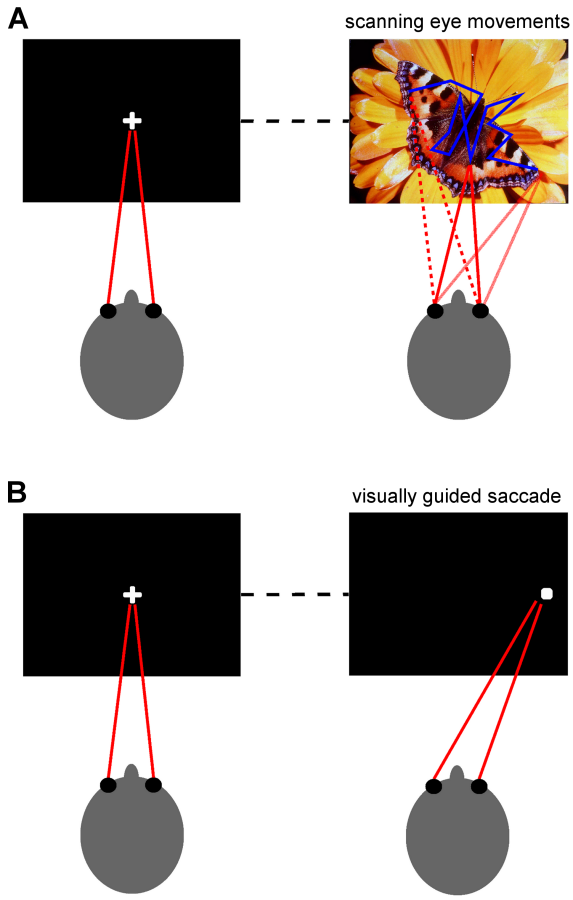

Figure 5.2: Eye movement (EM) tasks employed in the study. **A**: The scanning EM task. After presentation of the black screen with a central cross, a photograph chosen from the International Affective Picture System was presented for 2 s. Patients were asked to initially fix their eyes on the cross (left picture) and then simply watch the photograph (right picture). In total, 24 pictures were consecutively used during the task. The blue line highlights a possible eye scanpath. **B**: The visually guided saccade task consisted in a presentation of 10 pairs of indifferent central (left picture) and lateral GO (right picture) targets positioned pseudorandomly on the left/right side of the screen. Patients were instructed to initially fixate the central cross and then track to the lateral targets as fast as possible. *(Figure reproduced from Sieger et al. (2013).)*

(ACCEL), defined to be the derivative of VELOC.

To provoke EM, there were two visual tasks presented on a computer screen: the scanning EM task, and the saccade EM task. The goal of the *scanning EM task* was to induce self-initiated free-direction scanning EM. The task consisted of a presentation of a series of 24 photographs selected from the International Affective Picture System (IAPS, Fig. 5.2 A). Each picture was presented for 2 s and was preceded by a black screen for various durations ($750 - 2750$ ms) with a white cross in the center. Patients were instructed to watch the photographs and to fix their eyes on the cross on the black screen. The MER and EOG signals were recorded simultaneously in 2 s epochs both during the picture presentation and the black screen. The goal of the *visually guided saccade task* was to induce 10 externally generated horizontal saccades (Fig. 5.2 B). Patients were instructed to initially fix their eyes on a white cross shown in the center of a black screen for a pseudorandom period of 2, 2.25, or 2.5 s, and the to follow a peripheral target (a small white square) presented for 1 s pseudorandomly on the left (5 trials) or the right (5 trials) side of the screen. The MER and EOG

scanning EM task

visually guided saccade task

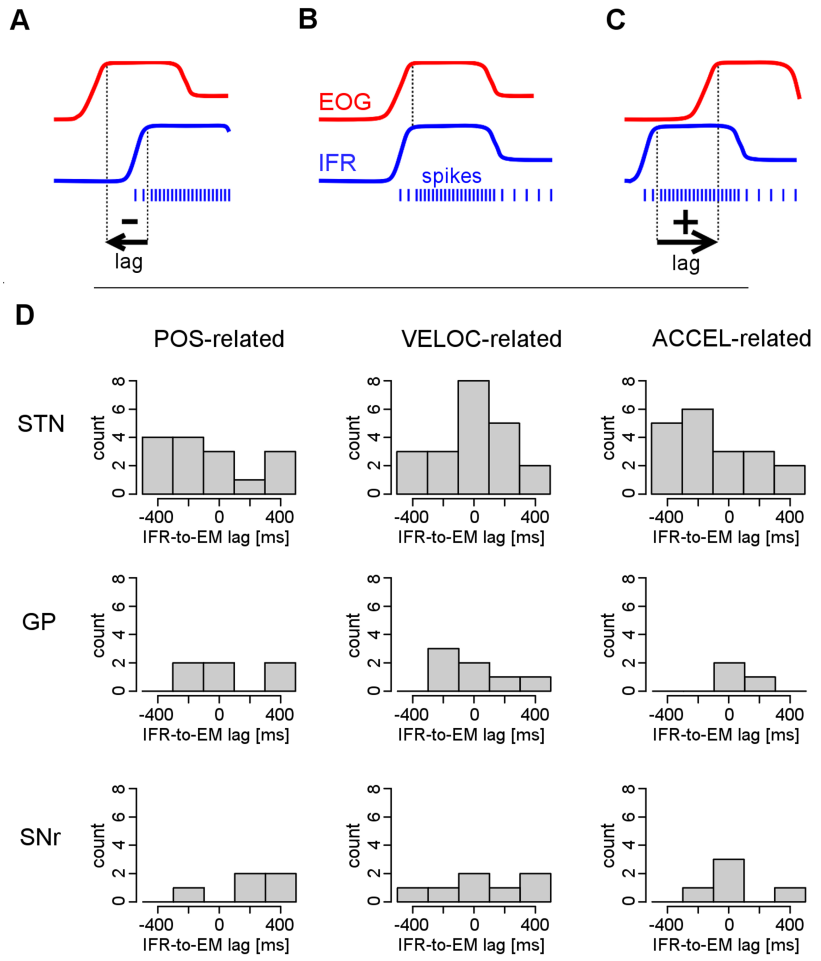

Figure 5.3: Time lag of neuronal activity with respect to electrooculography (EOG). **A, B, C**: Explanation of the cross-correlation procedure in three examples. Action potentials of three hypothetical neurons along with corresponding instantaneous firing rate (IFR) were correlated with theoretical EOG signal. **A**: the IFR correlates with past EOG signal suggesting sensory function of the neuron. **B**: the IFR correlates with concurrent EOG signal suggesting an executive function of the neuron. **C**: the IFR correlates with future EOG signal suggesting preparatory function of the neuron. The time lag of the IFR in which the maximal (and significant) correlation with EM is reached is called the optimal IFR-to-EM cross-correlation lag. This lag is positive in A, zero in B and negative in C. **D**: Frequency histograms of the optimal instantaneous firing rate (IFR) to eye movement cross-correlation lags in all eye movement-related neurons during the scanning eye movement task across the subthalamic nucleus (STN), globus pallidus (GP), and substantia nigra pars reticulata (SNr) considering kinematic parameters of the electrooculography (POS, VELOC, ACCEL, in columns). No significant differences in locations of these distributions were found. *(Figure reproduced from Sieger et al. (2013).)*

signals were recorded simultaneously in 2 s epoch in all 10 trials. The visually guided saccade task was executed in 4 patients only.

To identify EM-related neurons, the association between IFR and POS, IFR and VELOC, and IFR and ACCEL were assessed. A neuron was considered to be EM-related if its IFR was related to at least one of POS, VELOC, and ACCEL at the Bonferroni-corrected significance level of $p < 0.05$. The association between IFR and the EM characteristics were analysed using cross-correlation, allowing to detect relation not only between the concurrent IFR and EM, but also the relation between IFR and past/future EM (Fig. 5.3, A-C). The cross-correlation lags from the range of -500 to 500 ms with steps of 2.5 ms were considered. As longer lags lead to shorter overlaps of the two signals, correlation coefficients computed over longer lags were based on a smaller number of samples and were thus less informative. For

	STN (130 neurons)	GP (23 neurons)	SNr (30 neurons)	Total (183 neurons)
EM-related†	26(20%)**	5(22%)*	6(20%)***	37(20%)***
POS-related	15(12%)**	6(26%)***	5(17%)*	26(14%)***
VELOC-related	21(16%)***	7(30%)***	7(23%)***	35(19%)***
ACCEL-related	19(15%)***	3(13%)	5(17%)*	27(15%)***

Table 5.1: Number of neurons related to eye movements in the scanning eye movement task. EM-related neurons: the number of eye movement-related neurons associated with at least one kinematic parameter († Bonferroni-corrected number of neurons for three kinematic parameters). Neurons functionally associated with one or more kinematic parameters (POS - eye position; VELOC - eye velocity; ACCEL - eye acceleration) are reported for each nucleus separately. Number of neurons significantly greater than expected 5% false positive rate is denoted: *(p<0.05), **(p<0.01) ***(p<0.001). *(Table reproduced from Sieger et al. (2013).)*

this reason, biased estimates of correlation coefficients were computed:

$$CC[m] = \frac{1}{n} \sum_{i=1+max(0,m)}^{n+min(0,m)} IFR_i \; EM_{i-m} \tag{5.1}$$

where $CC[m]$ was the cross-correlation coefficient computed for lag m, and n was the length of each signal. The extreme (i.e. the maximal in absolute value) cross-correlation coefficient between each two signals was found and the lag in which this extreme cross-correlation was reached was called *the optimal EM-to-IFR cross-correlation lag*. The statistical significance of the cross-correlation was assessed using bootstrap[10]. The extreme cross-correlation coefficient CC_0 computed from IFR and EM signals was compared with the sampling distribution of the extreme cross-correlation coefficients $CC_b, b = 1, \ldots, 999$ computed from surrogate IFR and surrogate EM signals generated under the null hypothesis of their independence. If CC_0 fell outside the central mass of the distribution of CC_b, we could reject the null hypothesis of the independence of the original EOG and IFR signals. The p-value of the bootstrap test was computed using formula 3.6 as

$$p = \frac{\#\{|CC_b| \geq |CC_0|\} + 1}{1000}, \tag{5.2}$$

where $\#\{|CC_b| \geq |CC_0|\}$ was the number of CC_b at least as extreme as CC_0.

The surrogate signals were computed by transforming the original signals into the spectral domain, changing the phases of the spectra randomly according to the uniform distribution, and transforming the signals back to the time domain[11]. Specifically, the original signal x_i, $i = 1 \ldots n$ was transformed into the spectral domain representation s_k, $k = 0 \ldots n-1$ using the discrete Fourier transformation:

$$s_k = \sum_{i=1}^{n} x_i \; e^{-2\pi j k \frac{i-1}{n}}, \quad k = 0 \ldots n-1, \tag{5.3}$$

where j represents the imaginary unit ($j^2 = -1$). The spectral representation of the signal was complex and could be expressed as

$$s_k = a_k(\cos \varphi_k + j \sin \varphi_k), \quad k = 0 \ldots n-1, \tag{5.4}$$

[10] see Section 3.2

[11] this approach had been used previously e.g. in Simpson et al. (2001)

where a_k was the amplitude and φ_k the phase of s_k. As the spectral representation of a real signal is symmetric:

$$s_k = s^*_{n-k} \text{ for } k = 1 \dots n-1, \tag{5.5}$$

where the star denotes complex conjugation, and s_0 is real (as well as $s_{\frac{n}{2}}$ for n even), we needed to preserve these properties in the surrogate spectra. Thus, to generate b-th surrogate signal ($b = 1, \dots, 999$), we first replaced the phases φ_k with random ones:

$$\tilde{\varphi}^b_k = \begin{cases} 0 & \text{if } k = 0, \text{ and } k = \frac{n}{2} \text{ (for } n \text{ even)}, \\ \text{i.i.d} \sim U(0, 2\pi) & \text{for } k = 1 \dots \lfloor \frac{n-1}{2} \rfloor, \\ -\tilde{\varphi}^b_{n-k} & \text{for } k = \lfloor \frac{n-1}{2} \rfloor + 1 \dots n-1 \end{cases}, \tag{5.6}$$

then computed the spectrum of the surrogate signal by preserving the amplitudes of the original spectrum, but supplying random phases:

$$\tilde{s}^b_k = a_k(\cos \tilde{\varphi}^b_k + j \sin \tilde{\varphi}^b_k), \quad k = 0 \dots n-1, \tag{5.7}$$

and finally computed the surrogate signal $\tilde{x}^b_i$, $i = 1 \dots n$ by taking the inverse discrete Fourier transform of the surrogate spectrum $\tilde{s}^b_k$:

$$\tilde{x}^b_i = \sum_{k=0}^{n-1} \tilde{s}^b_k \, e^{2\pi j k \frac{i-1}{n}}, \quad i = 1 \dots n. \tag{5.8}$$

The resulting signals retained their original intrinsic properties, namely the power spectra and, as implied by the Wiener-Khinchin theorem[12], also their autocorrelations, but lost their potential time association.

Data processing and analyses were performed in MATLAB (R2007b, The MathWorks, Natick, MA) and R[13] (version 2.15.2).

[12] Chatfield C. *The analysis of time series: an introduction*. CRC Press, Florida, US, 6th edition, 2004

[13] R Core Team . *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL `http://www.R-project.org/`. ISBN 3-900051-07-0

5.1.3 *Results*

Out of 183 neurons, 37 (20%) EM-related neurons were found in the basal ganglia during the scanning EM task. We found 26 out of 130 neurons (20%) in the STN, 5 out of 23 neurons (22%) in the GP and 6 out of 30 neurons (20%) in the SNr. The exact number of neurons related to POS, VELOC, and ACCEL signals are shown in Table 5.1. Note that the numbers of neurons found was significantly higher compared to the expected false positive rate in each of the analysed nucleus (binomial test, $p < 0.001$).

The neuronal firing rate was found to be related either to concurrent, previous, or future EM (Fig. 5.3), and we found no prevalence of any kind of the time-related neurons to be present more frequently in some of the nuclei than in the others.

	STN (out of 46 neurons)			GP (out of 2 neurons)			SNr (out of 5 neurons)		
	SEM	VGS	Both	SEM	VGS	Both	SEM	VGS	Both
EM-related neurons[†]	10	10	2	0	1	0	1	2	0
POS-related	4	9	0	0	0	0	0	2	0
VELOC-related	9	4	1	0	0	0	1	0	0
ACCEL-related	8	11	3	0	1	0	2	0	0
POS&VELOC-related	3	4	0	0	0	0	0	0	0
POS&ACCEL-related	2	4	0	0	0	0	0	0	0
VELOC&ACCEL-related	4	2	0	0	0	0	1	0	0
POS&VELOC&ACCEL-rel.	2	2	0	0	0	0	0	0	0

Table 5.2: Eye movement-related neurons detected in the scanning eye movement task and/or visual guided saccade tasks. EM-related neurons the number of eye movement-related neurons associated with at least one kinematic parameter ([†] Bonferroni-corrected number of neurons for three kinematic parameters) identified from patients 16-19 which performed both the scanning eye movement task (SEM) and visual guided saccade task (VGS) in the subthalamic nucleus (STN), globus pallidus (GP) and substantia nigra pars reticulata (SNr). Neurons functionally associated with one or more kinematic parameters (POS - eye position; VELOC - eye velocity; ACCEL - eye acceleration) are reported for each nucleus separately. *(Table reproduced from Sieger et al. (2013).)*

The firing rate of the neurons related to eye position (POS) significantly correlated with fluctuations of the EOG (Pearson's $r = 0.89$ (STN), 0.91 (GP), 0.86 (SNr); df $= 18$, $p < 0.001$) (Fig. 5.4).

The locations of the EM-related neurons are depicted in Fig. 5.5. In the STN, according to our expectations, the EM-related neurons seemed to be located more frequently in the ventral part (0 to 1 mm (including) from the ventral STN border): the ratio of the 6/26 EM-related neurons was higher in the ventral part compared to the ratio of the 11/119 neurons in the rest of the nucleus (one-sided proportion test with continuity correction[14], $\chi^2 = 2.722$, df $= 1$, $p < 0.05$).

In the visually guided saccade task, we found 10 out of 46 neurons (22%) to be related to the saccades in the STN, 1 of 2 neurons in the GP and 2 of 5 in the SNr. Interestingly, most of the neurons related to scanning EM were found not to be related to the visually guided saccades and vice versa (Table 5.2). Out of 46 STN neurons found in patients, in which both the visually guided saccade task and the scanning EM task were executed, ten neurons were related to scanning EM, other ten neurons were related to visually guided saccades and only two were found to be active in both the tasks. These populations of neurons seemed to be independent, as we found no evidence against the null hypothesis of independence (Fisher exact test, $p = 1.0$), even though the test had enough power to reject the null hypothesis given the sample size[15]. Therefore, the fact that the test did not reject the null was because the data was in agreement with the null, not because the test had no power to reject the null. In the GP and SNr, the small number of neurons disallowed to rigorously assess the independence of neuronal populations being active in the two tasks. However, we found no GP or SNr neurons to be activate in both the tasks.

[14] Newcombe (1998), implemented in `prop.test` in R

[15] e.g., had the number of co-activated neurons were 5, the test would have rejected the null at the significance level of 5%

5.1.4 *Conclusions*

Our results suggest that basal ganglia, as represented by the STN, SNr and GP studied, contain relatively large populations (about 20% in each nucleus) of neurons related to scanning EM (Table 5.1). This implies that basal ganglia function is not only limited to saccade control[16], but play perhaps a more general role in EM systems, as suggested by previous findings[17]. Interestingly, oculomotor systems linked with spontaneous scanning EM and saccadic EM seemed to segregated (Table 5.2).

[16] Matsumura et al. (1992); Fawcett et al. (2007, 2005)

[17] Hikosaka et al. (2000)

I wish to emphasize that to the best of our knowledge this is the first study to investigate the neuronal activity of single neurons in the human basal ganglia in scanning EM.

From the methodological point of view, these results demonstrate that the association between a continuous (EOG) and discrete (a series of action potentials) signals can be assessed using traditional methods (cross-correlation analysis) provided proper transforms have been applied (i.e., the series of action potentials were converted to IFR). Furthermore, our results demonstrate how bootstrap can be used to generate the empirical sampling distribution of a statistic of interest, which in turn can easily be used to test statistical hypotheses.

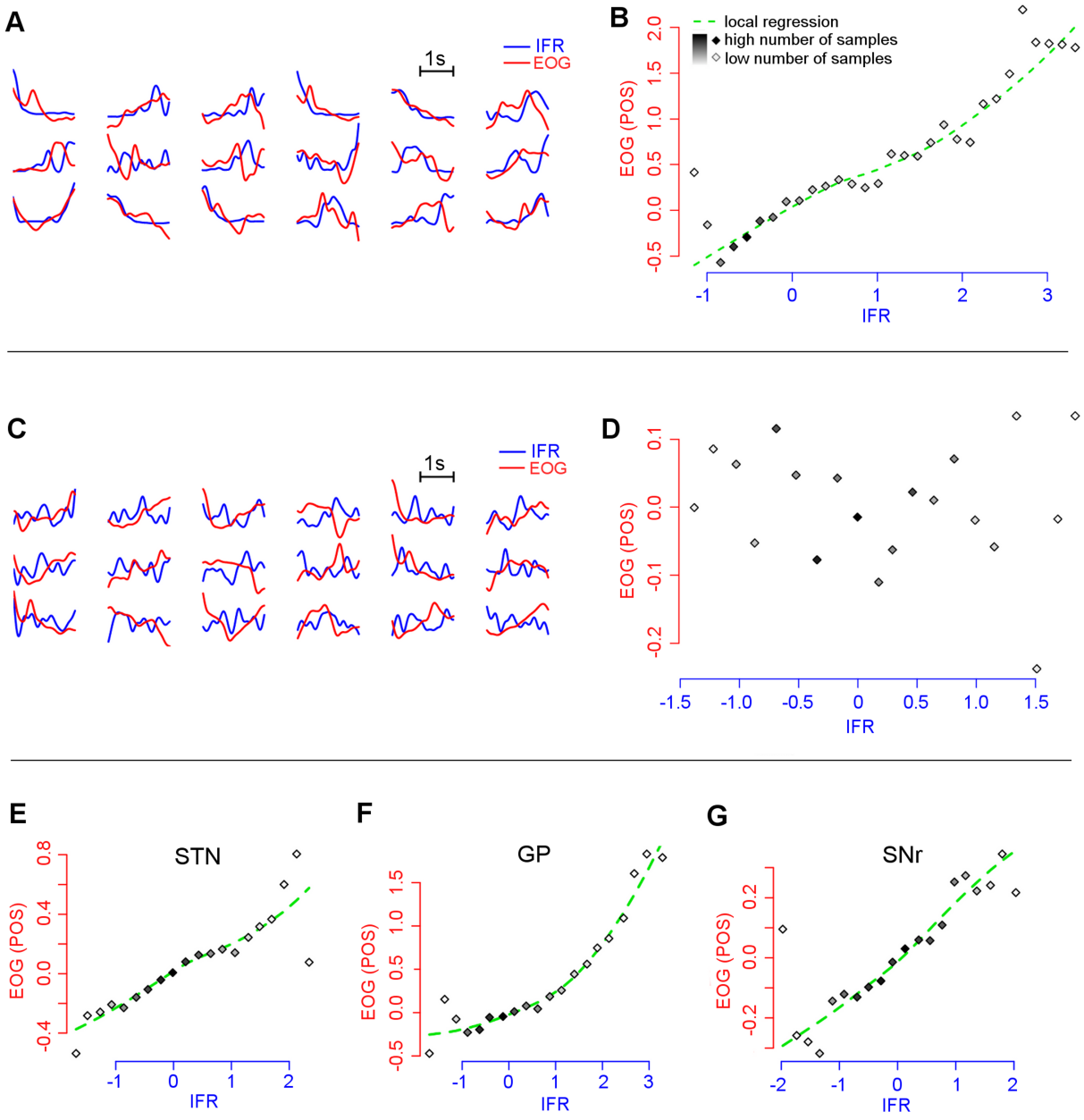

Fig. 5.4: Neuronal activity during the scanning movement task. Example of neuron related (**A, B**) and unrelated (**C, D**) to eye movements based on correlation analysis of the instantaneous firing rate (IFR) and eye position (POS) derived from the electrooculography (EOG). All eye movement-related neuronal populations in the STN, GP and SNr are plotted on the figures **E, F, G**. Figures **A, C** show the IFR (blue) and EOG (red) pairs recorded during epochs of the task involving both the black screen and pictures presentations. Figures **B, D, E, F, G** show the dependency of the normalized eye position (POS) derived from the electrooculography (EOG) on the normalized, sorted and binned amplitude of the instantaneous firing rate (IFR). While the IFR from a single neuron was used on figures **B** and **D**, the IFR from all eye sensitive neurons were used on figures **E**, **F**, and **G** for each nucleus separately. The amplitudes of the POS signals which correlated negatively with the IFR signal were reversed. The number of signal samples in each bin is expressed by different a grade shade of the diamond glyphs. *(Figure reproduced from Sieger et al. (2013).)*

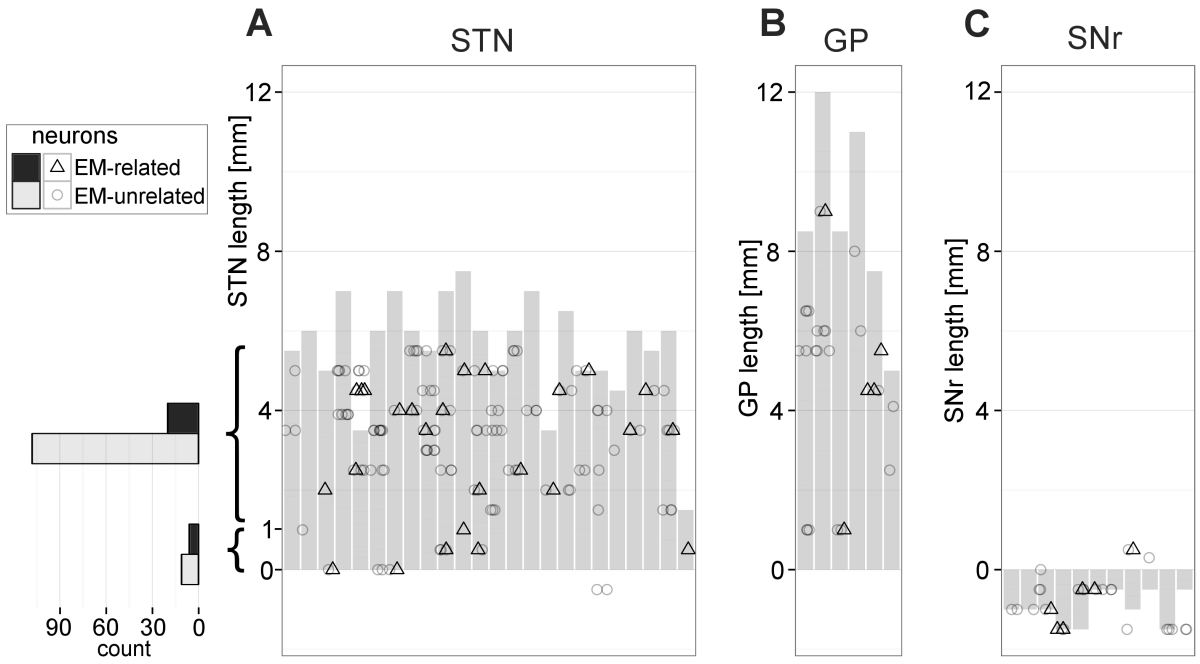

Fig. 5.5: Positions of the eye movement-related neurons along dorso-ventral microelectrode trajectory within the basal ganglia. **A**: length of the subthalamic nucleus (STN), **B**: length globus pallidus (GP) and **C**: length of the substantia nigra pars reticulata (SNr) explored intraoperatively by the five microelectrodes in both the left and right hemispheres and projected to one-dimensional space aligned to the ventral border of the STN and GP and to the dorsal border of the SNr. Position of each neuron along dorso-ventral axis is shown in each subject. Proportion of the eye movement-related neurons (EM) was significantly higher in the ventral part of the STN. *(Figure reproduced from Sieger et al. (2013).)*

5.2 Emotion-Related Neurons in the STN

In this section I demonstrate the application of linear models[18] in search for emotion-related neurons of the STN. This section is a follow-up of the previous Section 5.1.

[18] see Section 3.3

5.2.1 Introduction

Deep brain stimulation (DBS) in Parkinson's disease (PD) have demonstrated involvement of the subthalamic nucleus (STN) in emotional processes[19]. Recently, we observed[20] such alteration in terms of subjective ratings of rewarding and aversive stimuli in PD patients: in the STN DBS ON condition, the PD patients attributed lower valence scores to the aversive pictures compared with the DBS OFF condition and compared with controls, suggesting that STN DBS increases activation of the aversive motivational system. We further supplemented this finding by observing that the STN DBS in PD modifies blink reflex provoked by acoustic stimuli[21]: blink reflex to aversive stimuli was larger in PD patients in the DBS ON condition compared to controls, and also larger to erotic stimuli in PD patients in the DBS ON condition compared to the DBS OFF condition and controls. These results suggest that the STN DBS affects emotional processing.

[19] Huebl et al. (2011); Kuhn et al. (2005)

[20] Serranová T, Jech R, Dušek P, **Sieger T**, Růžička F, Urgošík D, and Růžička E. Subthalamic nucleus stimulation affects incentive salience attribution in Parkinson's disease. *Mov. Disord.*, 26(12):2260–2266, Oct 2011. *Contribution: 5%, citations: 5*

[21] Serranová T, **Sieger T**, Dušek P, Růžička F, Urgošík D, Růžička E, Valls-Sole J, and Jech R. Sex, food and threat: startling changes after subthalamic stimulation in Parkinson's disease. *Brain Stimul*, 6(5):740–745, Sep 2013. *Contribution: 10%, citations: 1*

We note that the involvement of the STN in emotional processing is not limited to PD patients. Recently, functional magnetic resonance studies gave evidence of this phenomenon also in healthy subjects[22].

[22] Frühholz and Grandjean (2012); Karama et al. (2011)

However, involvement of the STN in emotional processing has not been studied extensively on the single-neuron level. We thus systematically searched for STN neurons whose activity was related to emotional characteristics of visual stimuli presented to PD patients.

5.2.2 Methods

We analysed the data described previously in Section 5.1, focusing on STN neurons unrelated to eye movements (i.e., we excluded the eye movement-related neurons from this study). The number of the studied movement-unrelated neurons of the STN was 90.

To find emotion-related neurons, we related the alpha band activity of the studied neurons to the subjective valence and arousal ratings of the visual stimuli presented during the *scanning eye movements task*[23]. The emotional valence and arousal ratings of the visual stimuli were obtained from patients postoperatively. *Valence* rated the content of each picture on the discrete scale of $-3, -2, -1, 0, 1, 2, 3$, where -3 stood for extremely negative pictures (threat, or victimization), 0 stood for neutral pictures (e.g. household objects), and 3 stood for positive pictures (pictures depicting sex, food, or adventure). *Arousal*

[23] see page 71

emotional ratings

rated the intensity of each picture on the discrete scale of $1, 2, \ldots, 9$, where 1 stood for low-intensity pictures, and 9 for high-intensity ones. These scales were orthogonal to each other, e.g. there were negative (low-valence) pictures of both low and high arousal, and there also were highly arousing pictures either negative (low-valence), or positive (high-valence).

alpha band power

The power in the alpha band ($8 - 12$ Hz) was computed from spike trains as follows. The spike trains, i.e. the series $s_1, s_2, \ldots$ of action potentials, were binned into 5 ms time intervals $(t_{k-1}, t_k]$, yielding a sequence $D_k,\ k = 1 \ldots K$[24] of the number of action potentials in each time interval. The sequence $D_k,\ k = 1 \ldots K$ was treated as a discrete signal, which was subsequently standardized to zero mean and the discrete Fourier transform was carried out applying the Hann window of length 100 with the overlap of 75%. The mean power in the alpha band was estimated from whole FIX epochs, and from periods starting 0.5 s after picture onset in PIC epochs, because we did not expect the onset of emotion-related activity in the STN to come early after picture onset[25]. To stabilize power variance, the square root transformation was applied.

[24] see Section 3.5.2 on page 28 for details

[25] Brucke C, Kupsch A, Schneider G. H, Hariz M. I, Nuttin B, Kopp U, Kempf F, Trottenberg T, Doyle L, Chen C. C, Yarrow K, Brown P, and Kuhn A. A. The subthalamic region is activated during valence-related emotional processing in patients with Parkinson's disease. *Eur. J. Neurosci.*, 26(3):767–774, Aug 2007

model of valence-related neurons

The relation between the alpha band activity and the valence and arousal ratings was assessed using linear models. To find valence-related neurons, the following model was built for each neuron:

$$\begin{aligned} Y_i^{PIC} =& \beta_0 + \beta_{valence}\ valence_i + \beta_{FIX}\ Y_i^{FIX} + \beta_{PIC}\ Y_{i-1}^{PIC} + \epsilon_i, \\ & i = 2, \ldots, 24. \end{aligned} \tag{5.9}$$

The alpha band activity Y_i^{PIC} during the i-th PIC epoch was modeled using $valence_i$, the valence rating of the i-th picture. In addition, as exploratory analysis revealed that strong serial correlation was present in the alpha band activity (see below), each model also included other two covariates representing the activity during the two epochs preceding each PIC epoch analysed: Y_i^{FIX}, the activity in the preceding FIX epoch, and Y_{i-1}^{PIC}, the activity in the preceding PIC epoch of the experiment[26]. This way we adjusted for the past activity.

[26] this lead to the restriction of $i > 1$ in model 5.9, as there was no preceding PIC epoch for the first PIC epoch

model of arousal-related neurons

To find arousal-related neurons, the following model was built for each neuron:

$$\begin{aligned} Y_i^{PIC} =& \beta_0 + \beta_{arousal}\ arousal_i + \\ & \beta_{positive}\ \mathbb{1}\{category_i = positive\} + \\ & \beta_{negative}\ \mathbb{1}\{category_i = negative\} + \\ & \beta_{FIX}\ Y_i^{FIX} + \beta_{PIC}\ Y_{i-1}^{PIC} + \epsilon_i, \\ & i = 2, \ldots, 24. \end{aligned} \tag{5.10}$$

The alpha band activity Y_i^{PIC} during the i-th PIC epoch was modeled using $arousal_i$, the arousal rating of the i-th picture. To adjust for the category of each picture[27], there were two extra dummy covariates $\mathbb{1}\{category_i = positive\}$[28] and $\mathbb{1}\{category_i = negative\}$, respectively, which took the value of 1 if the category of the i-th picture was positive (or negative, respectively), and 0 otherwise. This way, the $\beta_{positive}$ and $\beta_{negative}$ coefficients stood for extra offsets of the alpha activity for positive and negative pictures, respectively. Thus, the activity during presentation of a neutral picture was modeled as $Y_i^{PIC} = \beta_0 + \beta_{arousal}\ arousal_i + \beta_{FIX}\ Y_i^{FIX} + \beta_{PIC}\ Y_{i-1}^{PIC} + \epsilon_i$, the activity during presentation of a negative picture was modeled as $Y_i^{PIC} = \beta_0 + \beta_{arousal}\ arousal_i + \beta_{negative}\ + \beta_{FIX}\ Y_i^{FIX} + \beta_{PIC}\ Y_{i-1}^{PIC} + \epsilon_i$, and the activity during presentation of a positive picture was modeled as $Y_i^{PIC} = \beta_0 + \beta_{arousal}\ arousal_i + \beta_{positive}\ + \beta_{FIX}\ Y_i^{FIX} + \beta_{PIC}\ Y_{i-1}^{PIC} + \epsilon_i$.

Similarly to model 5.9, we adjusted for the past activity using two extra covariates of Y_i^{FIX} and Y_{i-1}^{PIC}.

A neuron was considered to be related to valence, if the valence covariate was significant, i.e., if the $\beta_{valence}$ coefficient was significantly different from zero. Similarly, a neuron was considered to be related to arousal, if the arousal covariate was significant, i.e., if the $\beta_{arousal}$ coefficient was significantly different from zero.

serial correlation

The serial correlation in the alpha band was quantified in terms of the ability of past activity to predict the present activity. For each neuron, five linear models explaining the present activity in PIC epochs by particular types of past activity were built as follows:

$$Y_i^{PIC} = \beta_0 + \beta_{FIX}\ Y_i^{FIX} + \epsilon_i,\ i = 1, \ldots, 24 \qquad \text{(model 1)}$$

$$Y_i^{PIC} = \beta_0 + \beta_{PIC}\ Y_{i-1}^{PIC} + \epsilon_i,\ i = 2, \ldots, 24 \qquad \text{(model 2)}$$

$$Y_i^{PIC} = \beta_0 + \beta_{FIX}\ Y_{i-1}^{FIX} + \epsilon_i,\ i = 2, \ldots, 24 \qquad \text{(model 3)}$$

$$Y_i^{PIC} = \beta_0 + \beta_{PIC}\ Y_{i-2}^{PIC} + \epsilon_i,\ i = 3, \ldots, 24 \qquad \text{(model 4)}$$

$$Y_i^{PIC} = \beta_0 + \beta_{FIX}\ Y_{i-2}^{FIX} + \epsilon_i,\ i = 3, \ldots, 24 \qquad \text{(model 5)}$$

The significance of the explanatory past activity and the R^2 coefficient of determination[29] were computed for each of the models. The observed R^2 coefficients were compared to those expected assuming the present activity to be independent on the past. The critical values for the R^2 coefficient were derived considering that in the normal linear model the F statistic (usable for testing the full model against

[27] e.g. to be able to model the hypothetical case of negative pictures evoking (on average) higher activity compared to neutral pictures of the same arousal

[28] $\mathbb{1}\{X\}$ stands for the "identity" operator taking the value of 1, or 0, depending on whether the argument X is true, or false, respectively

[29] expressing the percentage of the present activity explained by the past activity

a submodel having no covariates except the absolute term) can be expressed using R^2 as follows[30]:

[30] see Zvára (2008), eq. 3.25

$$F = \frac{R^2}{1 - R^2} \cdot \frac{n - r}{r - 1}, \tag{5.11}$$

where n is the number of observations in the model, and r is the number of the parameters in the model, such that

$$R^2 = \frac{F}{\frac{n-r}{r-1} + F}. \tag{5.12}$$

Thus, the critical values (95% quantiles) of R^2 were computed as

$$R^2_{95\%} = \frac{\mathbf{F}^{-1}_{r-1,n-r}(95\%)}{\frac{n-r}{r-1} + \mathbf{F}^{-1}_{r-1,n-r}(95\%)}, \tag{5.13}$$

where $\mathbf{F}^{-1}_{r-1,n-r}$ was the inverse distribution function (i.e. the quantile function) of the $\mathbf{F}$ distribution with $r - 1$ and $n - r$ degrees of freedom.

5.2.3 *Results*

Out of the 90 eye movement-unrelated STN neurons, the alpha band activity of 14 neurons (16%) was related to emotional characteristics of the visual stimuli presented. The activity of 9 (10%) neurons was associated with the arousal ratings of the presented pictures (linear model coefficient test, $p < 0.05$). Seven neurons were associated positively with the arousal ratings, two neurons were associated negatively (Fig. 5.6). The activity of other 6 (7%) neurons was related to the valence ratings (linear model coefficient test, $p < 0.05$). Four neurons were associated negatively, and two neurons were associated positively with the valence ratings (Fig. 5.7).

Locations of the arousal- and valence-related neurons are depicted in Fig. 5.8. The location of two valence-related neurons could not be assessed due to missing data. The valence-related neurons were located more posteriorly compared to the arousal-related neurons (Hotelling's test, $T^2 = 5.47$, df $= 3$ and 9, $p < 0.05$). The anteroposterior difference in the means of the neuronal populations was 2.0 mm.

Serial correlation in the alpha band activity was found as illustrated by the increased R^2 coefficients of determination in models predicting the present activity in terms of the past activity (Fig. 5.9). In a significant number of neurons (binomial test, $p < 0.05$), the activity in PIC_i could be predicted using the past activity in FIX_i epochs, but also by the activity in PIC_{i-1}, FIX_{i-1}, and PIC_{i-2} epochs (separately). I.e., serial correlation was detected over periods of up to 13 s.

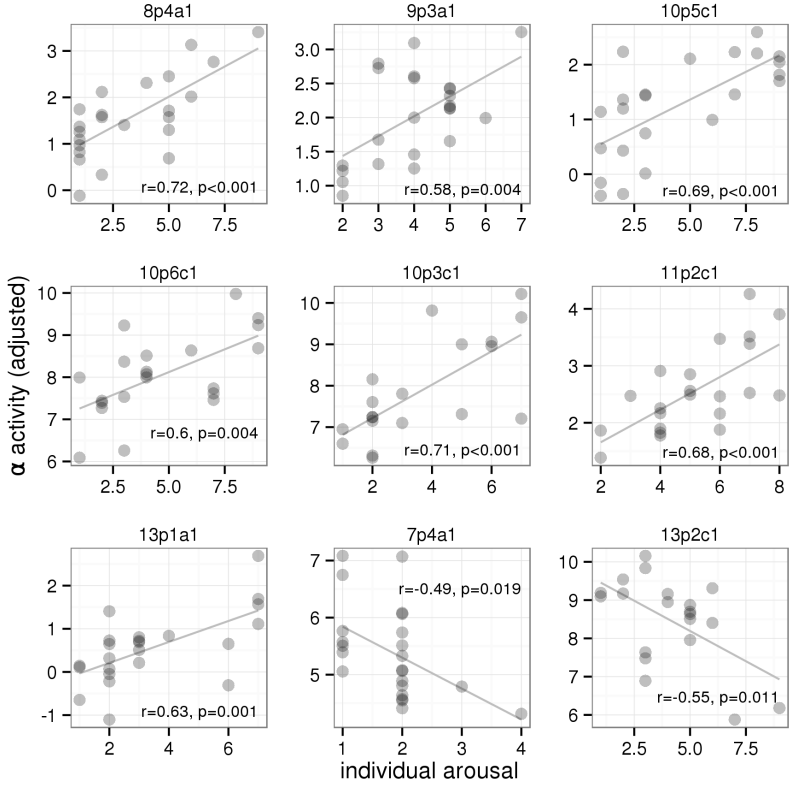

Figure 5.6: The dependency of the alpha band activity during picture presentation (PIC epochs) on the individual arousal ratings of the presented pictures in 9 neurons, for which the dependency was significant. The activity was adjusted for the past activity (two immediately preceding FIX and PIC epochs) and picture categories. For visualization purposes, correlation coefficients and their significances were included. Neuron identifications in titles consist of the patient number, the recording position, the first letter of the identification of the recording electrode and the serial number of the neuron detected at the electrode. For example, the identification "8*p*4*a*1" refers to the first neuron detected on the anterior electrode in the fourth recording position in patient 8.

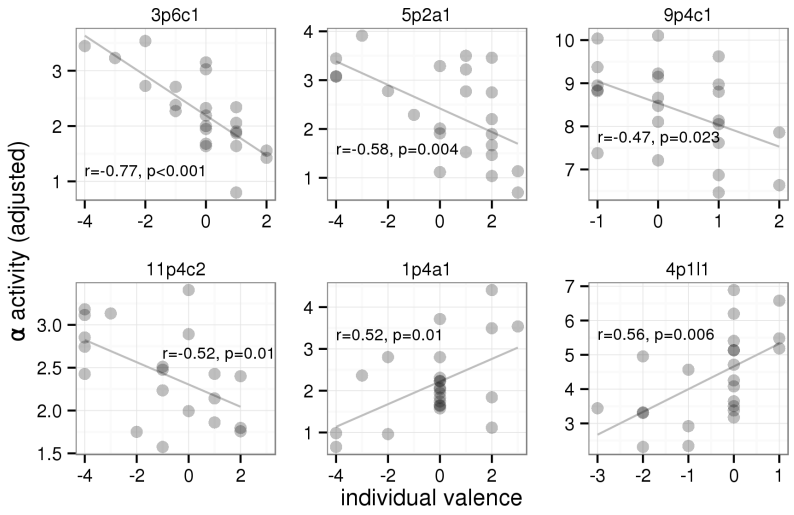

Figure 5.7: The dependency of the alpha band activity during picture presentation (PIC epochs) on the individual valence ratings of the presented pictures in 6 neurons, for which the dependency was significant. The activity was adjusted for the past activity (two immediately preceding FIX and PIC epochs). For visualization purposes, correlation coefficients and their significances were included. For explanation of neuron identification in titles, see Fig. 5.6.

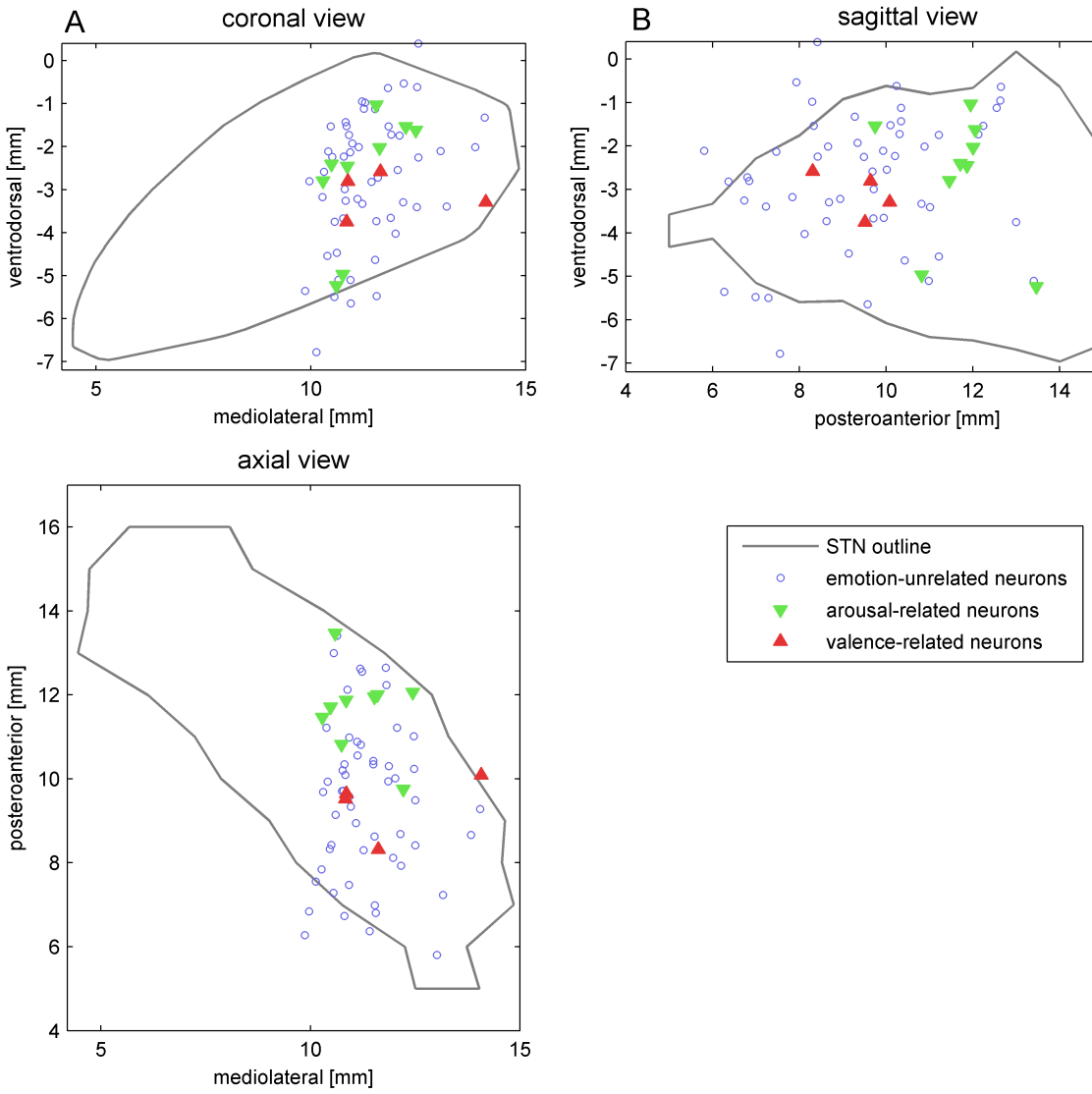

Figure 5.8: Positions of the STN neurons related to emotional characteristics of the presented pictures.

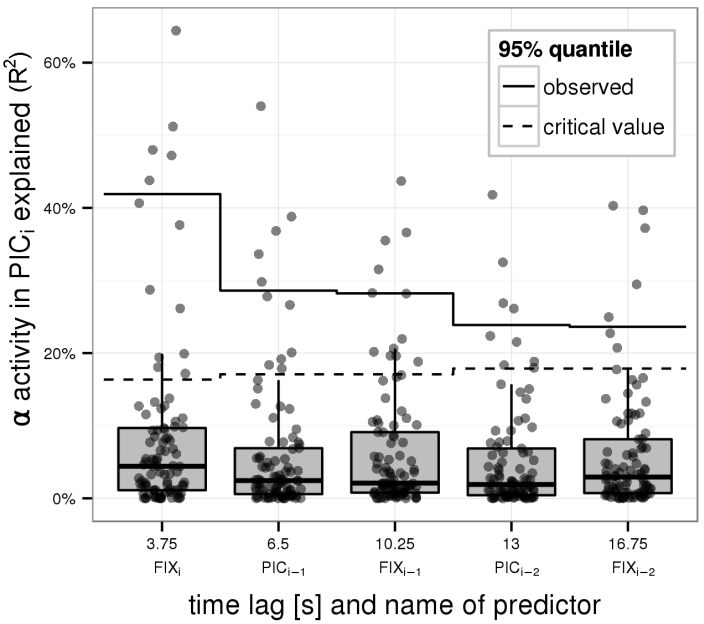

Figure 5.9: The serial correlation in the alpha band activity as expressed by the ability of past activity to predict the present activity. The percentage of activity during PIC_i epochs (vertical axis) explained by particular types of the past activity (horizontal axis) is shown for each of the 90 STN neurons. The observed percentages are summarized by boxplots and 95% quantiles (solid line) and contrasted with their critical value (dashed line) computed assuming the present activity to be independent on the past. Covariates predicting the activity in PIC_i epochs included, from left to right, the FIX_i epochs immediately preceding the PIC_i epochs, the previous PIC_{i-1} epochs, the FIX_{i-1} epochs preceding the previous PIC_{i-1} epochs, etc.

5.2.4 *Conclusions*

This is the first study to provide a direct evidence of the presence of single emotion-related neurons in the human STN. This finding is in agreement with previous studies observing STN to be connected to emotional processing [31]. We observed that distinct neuronal populations in the STN responded to emotional valence and arousal, suggesting functional separation of processing of these two emotional dimensions, which could be observed in physiological studies[32], but has not been observed at the level of single neurons of the STN.

Moreover, we revealed that strong serial correlation was present in the activity of individual neurons in the STN, suggesting some slow processes can take place in the STN simultaneously to relatively fast processing of emotional stimuli. We thus highly recommend to adjust for the past activity in models of neuronal activity of the STN.

[31] Peron J, Fruhholz S, Verin M, and Grandjean D. Subthalamic nucleus: a key structure for emotional component synchronization in humans. *Neurosci Biobehav Rev*, 37(3):358–373, Mar 2013

[32] Cacioppo J. T, Petty R. E, Losch M. E, and Kim H. S. Electromyographic activity over facial muscle regions can differentiate the valence and intensity of affective reactions. *J Pers Soc Psychol*, 50(2): 260–268, Feb 1986

5.3 Task-Related Neurons in the STN

In this section I demonstrate the application of spike train models[33] in combination with bootstrap[34] in the search for STN neurons, whose activity differed between two visuo-attentional tasks.

[33] see Section 3.5

[34] see Section 3.2

5.3.1 Introduction

The role of the subthalamic nucleus (STN) in influencing the basal ganglia is crucial. While motor functions of STN have been suggested by both clinical and physiological studies[35], there is also evidence that STN is involved in cognitive and associative functions[36]. Such evidence comes mainly from reports on the behavioral side-effects of STN deep brain stimulation (DBS) in Parkinson disease (PD)[37], or animal studies[38]. We studied the cognitive functions of the STN in terms of searching for individual task-related neurons in the STN in PD patients.

[35] Purves D. *Neuroscience*. Sinauer, 4th edition, July 2008. ISBN 9780878936977

[36] Temel Y, Blokland A, Steinbusch H. W, and Visser-Vandewalle V. The functional role of the subthalamic nucleus in cognitive and limbic circuits. *Prog. Neurobiol.*, 76(6):393–413, Aug 2005

[37] Herzog et al. (2003)

[38] Temel et al. (2006); Baunez et al. (2001)

5.3.2 Methods

We analysed the data described previously in Section 5.1 and 5.2, focusing on the 90 STN neurons unrelated to eye movements.

The data consisted of repeated spike trains recorded under two different conditions:

- during FIX epochs consisting of the presentation with a black screen with a central cross (Fig. 5.2 A, left[39]), and
- during PIC epochs consisting of the presentation with IAPS pictures (Fig. 5.2 A, right).

[39] see page 71

The duration of each epoch was 2 s, with the exception of 8 neurons, in which, by design, the epochs lasted for 1 s only. Usually, there were 24 trials recorded under each condition in each neuron. However, due to technical problems, there were 1 or 2 trials missing in some neurons.

To evaluate whether the activity of a neuron in PIC epochs differed from the activity in FIX epochs, the mean instantaneous firing rate (MIFR) in PIC epochs (referred to as $MIFR_{PIC}$) was computed and contrasted with MIFR in FIX epochs ($MIFR_{FIX}$) (Fig. 5.10 A).

mean instantaneous firing rate (MIFR)

MIFR was computed by binning the spike trains recorded in PIC (or FIX) epochs into 5 ms bins, averaging the spike counts in each bin, and smoothing the averages by the normal kernel with inter-quartile range set to 250 ms. Formally, MIFR in the time bin $(t_k, t_{k+1}]$ was defined as the weighted average of probabilities of firing a spike in nearby time

bins:

$$MIFR((t_k, t_{k+1}]) = \sum_{i=0}^{K-1} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(t_i - t_k)^2}{2\sigma^2}\right) \cdot \hat{P}(\text{spike in } (t_i, t_{i+1}]), \tag{5.14}$$

where σ was set to 185 ms (such that the inter-quartile range was 250 ms) and the probability $\hat{P}(\text{spike in } (t_i, t_{i+1}])$ of firing a spike in the interval $(t_k, t_{k+1}]$ was estimated by the average number of spikes in that interval:

$$\hat{P}(\text{spike in } (t_k, t_{k+1}]) = \frac{1}{R} \sum_{i=1}^{R} D_{i,k} \tag{5.15}$$

where $D_{i,k}$ was the number of spikes fired in the i-th trial in the time interval $(t_k, t_{k+1}]$, and R was the number of trials recorded from given neuron.

To find neurons whose $MIFR_{PIC}$ differed from $MIFR_{FIX}$, we followed a parametric bootstrap approach[40]. Under the null hypothesis of no difference in the processes driving the neuronal firing under the two conditions, $MIFR_{PIC}$ should be similar to $MIFR_{FIX}$. However, they could not be exactly the same, due to the stochastic nature of the underlying neuronal processes. To assess what amount of fluctuations in MIFR can be accounted for by pure chance alone, 999 $\widetilde{MIFR}^b_{FIX}$, $b = 1 \ldots 999$ bootstrapped samples based on $MIFR_{FIX}$ were generated, and point-wise 95% confidence boundaries were computed from them by taking the 2.5% and 97.5% quantiles in each time bin.

[40] see Section 3.2; similar approach was applied in a different setting in Ventura et al. (2005a)

MIFR confidence boundaries

The $\widetilde{MIFR}^b_{FIX}$ bootstrap samples were generated by parametric bootstrap[41] as follows. First, bootstrap spike trains were generated using a Poisson process by drawing the number of spikes in each run i in each time interval $(t_k, t_{k+1}]$ from the Poisson distribution, whose parameter was set equal to $MIFR_{FIX}((t_k, t_{k+1}])$:

$$\tilde{D}^b_{i,k} \sim Po(MIFR_{FIX}(t_k, t_{k+1}])),$$

and then computing the bootstrap probabilities of firing $\hat{\tilde{P}}^b$ according to Eq. 5.15 and the bootstrap $\widetilde{MIFR}^b_{FIX}$ according to Eq. 5.14.

MIFR parametric bootstrap

[41] nonparametric bootstrap, in which spike trains would have been sampled with replacement, could also been used, but we preferred the parametric approach which was shown to be more appropriate in case of small or moderate number of trials, and less sensitive to trial-to-trial variability (Ventura et al., 2005b).

It should be noted that the $\widetilde{MIFR}^b_{FIX}$ samples generated under the null hypothesis can go outside the point-wise confidence boundaries much more often than 5% of the time (just because the confidence boundaries are point-wise), and that such excursions can happen in multiple time intervals. This makes these boundaries impractical in assessment of whether $MIFR_{PIC}$ differs from $MIFR_{FIX}$. Therefore, we based the assessment of $MIFR_{PIC} \stackrel{?}{=} MIFR_{FIX}$ on the largest area of excursion of $MIFR_{PIC}$ outside the confidence boundaries of $MIFR_{FIX}$. Thus, we estimated the sampling distribution of the areas of excursion

outside these confidence boundaries under the null hypothesis (i.e. for each $\widetilde{MIFR}^b_{FIX}$), and contrasted the area of excursion of $MIFR_{PIC}$ with this distribution. Informally, had the area of excursion of $MIFR_{PIC}$ been located in the right tail of the sampling distribution, there would be some evidence of $MIFR_{PIC}$ to differ from $MIFR_{FIX}$.

Formally, the largest area of excursion $A(f)$ of a function f[42] outside the confidence bands cb_{low} and cb_{high} was defined as:

[42] defined on time intervals $(t_k, t_{k+1}]$

$$A(f) = max_{i_1 \leq i_2}(\sum_{i=i_1}^{i_2} f((t_i, t_{i+1}]) - cb_{high}((t_i, t_{i+1}]), \sum_{i=i_1}^{i_2} cb_{low}((t_i, t_{i+1}]) - f((t_i, t_{i+1}])). \tag{5.16}$$

The assessment of whether $MIFR_{PIC}$ differed from $MIFR_{FIX}$ in given neuron, an *excursion-based bootstrap test* was employed. This test contrasted the largest excursion area $A(MIFR_{PIC})$ of $MIFR_{PIC}$ outside the $MIFR_{FIX}$ confidence boundaries (Fig. 5.10 A) with areas of excursions occurred by chance alone, i.e. with the sampling distribution of the largest excursion areas $A(\widetilde{MIFR}^b_{FIX})$. Formally, the p-value of the bootstrap test was defined as:

excursion-based bootstrap test

$$p = \frac{\#\{A(\widetilde{MIFR}^b_{FIX}) \geq A(MIFR_{PIC})\} + 1}{1000}, \tag{5.17}$$

where $\#\{A(\widetilde{MIFR}^b_{FIX}) \geq A(MIFR_{PIC})\}$ was the number of $\widetilde{MIFR}^b_{FIX}$ null hypothesis-compatible excursion areas being greater or equal to the observed excursion area of $MIFR_{PIC}$. Small p-values thus represented cases in which the observed excursion area was considerably larger compared to excursions occurring by chance alone. P-values smaller than 5% were considered significant.

5.3.3 *Results*

In 45 (50%) neurons the mean activity during PIC epochs differed from the activity in FIX epochs as demonstrated by a significant excursion of $MIFR_{PIC}$ outside the 95% probability boundaries of $MIFR_{FIX}$ (Fig. 5.10 B). In the significant majority of 31 of them (binomial test, $p < 0.05$), the largest excursion of $MIFR_{PIC}$ occurred above the confidence boundaries of $MIFR_{FIX}$, i.e., the activity specific to PIC epochs was characterized by increased firing in some time interval. In the rest of 14 neurons, the largest excursion occurred below the boundaries, and these neurons were located in the dorsal part of the STN exclusively (Fig.5.11)(Fisher test, $p < 0.01$), as opposed to the majority sub-population, in which there was no apparent tendency to cluster in some part of the STN.

Out of 9 arousal-related neurons[43], a significant excursion was ob-

[43] which we found and described in Section 5.2

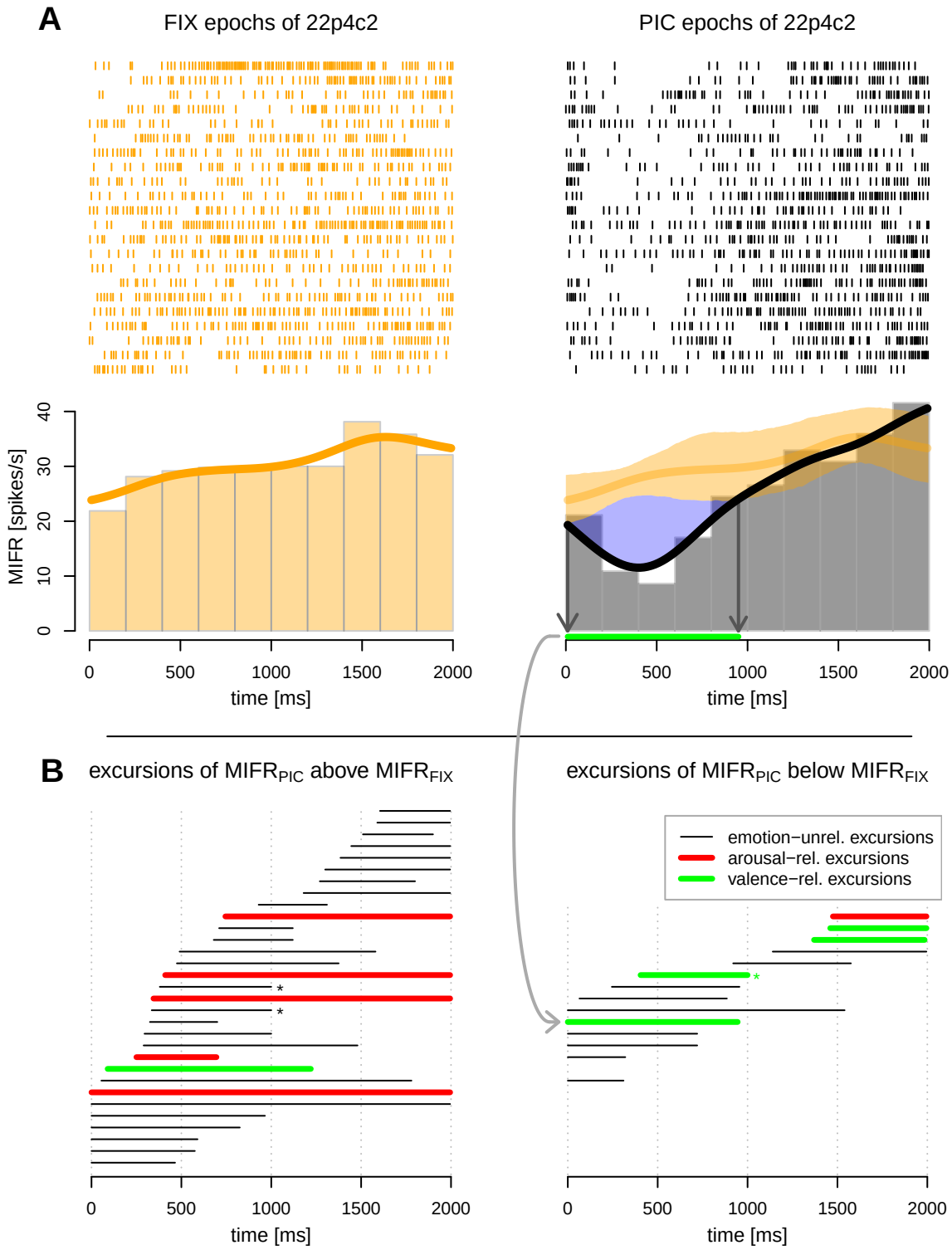

Fig. 5.10: Differences in the mean instantaneous firing rate (MIFR) between FIX and PIC epochs. **A**: spike trains recorded during FIX (left) and PIC (right) epochs from neuron 22*p*4*c*2 were binned to form a MIFR histogram and a smoothed MIFR estimate (bold curve). For comparison purposes, the $MIFR_{PIC}$ is overlaid with $MIFR_{FIX}$ and its 95% confidence band (light orange area). The excursion of $MIFR_{PIC}$ outside the 95% confidence boundaries of $MIFR_{FIX}$ (blue area) lasted from 10 ms to 950 ms (green line). **B**: excursions of $MIFR_{PIC}$ above/below the 95% confidence boundaries of $MIFR_{FIX}$ (left/right). To ease visual evaluation of the duration of the excursions, all excursions are sorted by their start time. The excursions made by arousal- and valence-related neurons are highlighted in red and green, respectively. The excursion made by neuron 22*p*4*c*2 as shown in **A** is pointed to by an arrow. Asterisks mark excursions made by neurons for which data after 1 s are missing.

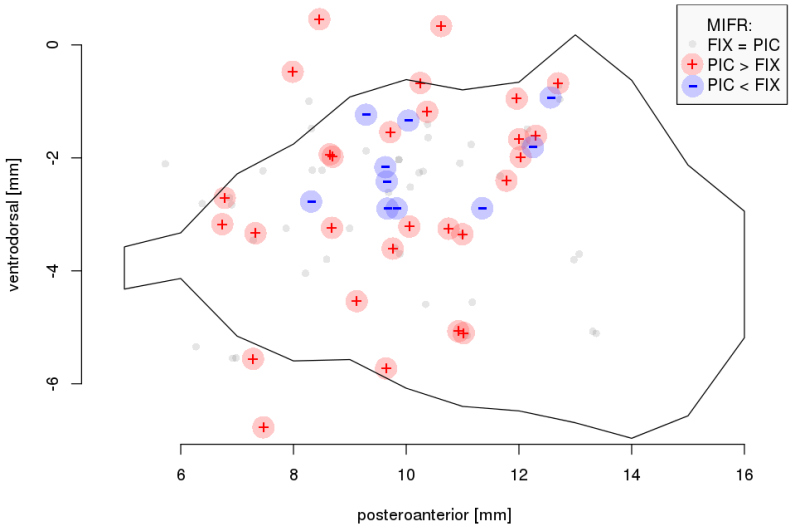

Figure 5.11: Positions of neurons whose mean instantaneous firing rate (MIFR) differed between FIX and PIC epochs. Neurons that fired less frequently during PIC epochs tended to be located in the dorsal part of the STN.

served in 6 of them, and in 5 of them, the largest excursion of PIC MIFR occurred above the 95% confidence boundaries of $MIFR_{FIX}$ (Fig. 5.10 B). Out 6 valence-related neurons[44], a significant excursion of $MIFR_{PIC}$ outside the 95% confidence boundaries of $MIFR_{FIX}$ was observed in 5 of them, and in 4 of them, the largest excursion of $MIFR_{PIC}$ occurred below the $MIFR_{FIX}$ boundaries. In other words, most emotion-related neurons that exhibited an excursion of $MIFR_{PIC}$ above $MIFR_{FIX}$ were arousal-related (5 out of 6). Most emotion-related neurons that exhibited an excursion of $MIFR_{PIC}$ below $MIFR_{FIX}$ were valence-related (4 out of 5).

[44] see Section 5.2

5.3.4 *Conclusions*

We have identified a large number of 45 neurons (50%) of the STN, whose activity was related to the presented visual task. As we focused on non-motor neurons only, this finding suggests that in the STN there exist neurons related to sensing visual information and/or paying attention to the course of the task. This supplements the previous finding of neurons in the monkey STN that were related to visual processing[45] and vindicates the cognitive role of the STN[46].

Thanks to the parametric bootstrap approach taken, we could identify not only which neurons exhibited a task-specific difference, but also to locate these differences in time. We suggest to prefer such a direct analytical approach over indirect methods operating over summary statistics aggregated from the spike train data.

[45] Matsumura M, Kojima J, Gardiner T. W, and Hikosaka O. Visual and oculomotor functions of monkey subthalamic nucleus. *J. Neurophysiol.*, 67(6):1615–1632, Jun 1992

[46] Temel Y, Blokland A, Steinbusch H. W, and Visser-Vandewalle V. The functional role of the subthalamic nucleus in cognitive and limbic circuits. *Prog. Neurobiol.*, 76(6):393–413, Aug 2005

6
Conclusions

I successfully met all the objectives established. I reviewed existing and developed new original methods for processing and analysing single-neuron recordings. Two of the three methods developed can serve as general-purpose tools applicable not only in the context of neuroscientific data, but virtually in any field of science or industry.

I also successfully applied both the existing and original methods to important neuroscientific problems in the quest for elucidating the function of the basal ganglia in the human brain. I wish to emphasize that our contribution to neuroscience is significant: thanks to the methods developed, we could investigate the activity of single neurons in the human basal ganglia and were the first to report their activity to be related to eye movements.

6.1 Thesis Achievements

I reached all the goals set in Section 1.1. When it comes to methodological achievements:

- I designed a general concept of a software framework[1] enabling **automated access to and processing of data**, and implemented the framework in the MATLAB programming language. The framework has become a de facto standard for accessing neurophysiological data in our research group, having provided access to the data for over five years. It enables analytical applications to focus purely on the data of their interest, not on technical details of how to access or transform the data.
- I implemented `idendro`[2], an original **data exploratory tool** enabling users to interactively inspect the structure of data[3]. This tool was proven to be valuable especially when exploring high-dimensional data, e.g. spikes detected in single-neuron recordings[4], but also flow-cytometry or spectroscopic data. The tool is implemented in the R programming language and freely available as an R package

[1] see Section 4.1

[2] see Section 4.2

[3] in terms of browsing a dendrogram resulting from hierarchical cluster analysis of the data

[4] see Section 4.2.7

from `https://github.com/tsieger/idendro`. The description of the tool has been submitted to the Journal of Statistical Software (2012 IF 4.91).

- We evaluated existing methods for detecting the activity of individual neurons in single-neuron recordings[5]. This evaluation was published[6] in the Journal of Neuroscience Methods (2012 IF 2.484). `WaveClus`[7], the best performing method on our type of data, was chosen to transform our single-neuron recordings into traces of the activity of individual neurons.

I also met the neuroscientific goals set. In particular:

- About 20% of the basal ganglia neurons studied were found to be related to planning, execution and/or control of **eye movements** (EM)[8]. These results suggest that oculomotor systems linked with spontaneous scanning EM and saccadic EM are segregated. To the best of my knowledge, this is the **first study** to investigate the neuronal activity of single neurons in the human basal ganglia in scanning EM. These results were published[9] in PLoS ONE (2012 IF 3.73).
- 15% of the basal ganglia neurons studied were identified to be related to processing or responding to distinct types of **emotional stimuli**[10]. This is the **first study** to provide direct evidence of individual emotion-related neurons in the human STN. These results are about to be published.
- One half of the basal ganglia neurons studied were revealed to be **activated during a visuo-attentional task**[11]. This is also a novel finding. A publication presenting those results is under preparation currently.

To summarize, I contributed to computer science, applied statistics, and neuroscience.

In the context of **computer science**, I proposed a simple yet efficient way of accessing data. The methodology designed is general, applicable to many different study areas.

My contribution to **applied statistics** lied mainly in implementing a general-purpose interactive exploratory tool that makes it possible to study the structure of high-dimensional data. The tool has already proven to be applicable not only in the context of neurophysiological data, but also when analysing flow-cytometry and spectroscopic data. Furthermore, I reviewed several general statistical techniques (the bootstrap, linear models, generalized linear models) as well as modern methods for analysing spike train data, and presented these techniques concisely to facilitate their adoption.

[5] see Section 4.3

[6] Wild J, Prekopcsák Z, **Sieger T**, Novák D, and Jech R. Performance comparison of extracellular spike sorting algorithms for single-channel recordings. *Journal of Neuroscience Methods*, 203(2):369–376, January 2012. ISSN 01650270. DOI: 10.1016/j.jneumeth.2011.10.013. *Contribution: 15%, citations: 1*

[7] Quiroga R. Q, Nadasdy Z, and Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004

[8] see Section 5.1

[9] **Sieger T**, Cecilia B, Serranová T, Wild J, Novák D, Růžička F, Urgošík D, Růžička E, Gaymard B, and Jech R. Basal ganglia neuronal activity during scanning eye movements in Parkinson's disease. *PLoS ONE*, 8(11):e78581, 2013. DOI: 10.1371/journal.pone.0078581. *Contribution: 40%*

[10] see Section 5.2

[11] see Section 5.3

From the **neuroscientific** point of view, we contributed to the understanding of the human brain in general, and to the mechanism of the deep brain stimulation in particular. Our results should help to advance the future treatment of Parkinson's disease, making the treatment more effective and increasing the quality of life of Parkinson's disease patients.

6.2 List of Candicate's Publications Related to the Thesis

6.2.1 Publications in Impacted Journals

Sieger T, Cecilia B, Serranová T, Wild J, Novák D, Růžička F, Urgošík D, Růžička E, Gaymard B, and Jech R. Basal ganglia neuronal activity during scanning eye movements in Parkinson's disease. *PLoS ONE*, 8(11):e78581, 2013. *Contribution: 40%*.

Sieger T, Hurley C. B, Fišer K, and Beleites C. Interactive dendrograms: the `idendro` package for R. *Journal of Statistical Software*. *(submitted); Contribution: 70%*.

Wild J, Prekopcsák Z, **Sieger T**, Novák D, and Jech R. Performance comparison of extracellular spike sorting algorithms for single-channel recordings. *Journal of Neuroscience Methods*, 203(2):369–376, January 2012. *Contribution: 15%, citations: 1*.

Serranová T, **Sieger T**, Dušek P, Růžička F, Urgošík D, Růžička E, Valls-Sole J, and Jech R. Sex, food and threat: startling changes after subthalamic stimulation in Parkinson's disease. *Brain Stimul*, 6(5):740–745, Sep 2013. *Contribution: 10%, citations: 1*.

Serranová T, Jech R, Dušek P, **Sieger T**, Růžička F, Urgošík D, and Růžička E. Subthalamic nucleus stimulation affects incentive salience attribution in Parkinson's disease. *Mov. Disord.*, 26(12):2260–2266, Oct 2011. *Contribution: 5%, citations: 5*.

6.2.2 Selected Conference Reports

Sieger T, Bonnet C, Serranová T, Wild J, Novák D, Růžička F, Urgošík D, and Jech R. Neuronal activity of the basal ganglia and subthalamus in relation to eye movement in Parkinson's disease. In *The 14th European Congress on Clinical Neurophysiology - 4th International Conference on Transcranial Magnetic and Direct Current Stimulation*, pages s89–s90. Elsevier Irland Ltd., 2011. *Contribution: 76%*.

Novák D, Wild J, **Sieger T**, and Jech R. Identifying number of neurons in extracellular recording. In *2009 4th International IEEE/EMBS*

Conference on Neural Engineering, NER '09, pages 742–745, 29 April through 2 May 2009. *Contribution: 20%, citations: 3*.

Sieger T, Jech R, Serranová T, Wild J, Bonnet C, Novák D, Růžička F, and Urgošík D. Neuronální aktivita bazálních ganglií vázaná na oční pohyby u pacientů s Parkinsonovou nemocí. In *57. společný sjezd České a Slovenské společnosti pro klinickou neurofyziologii*, pages 11–11. MH Consulting, 2010. *Contribution: 80%*.

Wild J, **Sieger T**, Novák D, and Jech R. Spike sorting algorithms comparison. In *2nd INCF Congress of Neuroinformatics*, pages 53–54, 2009. *Contribution: 25%*.

Wild J, Novák D, **Sieger T**, Prekopscsák Z, Vostatek P, and Jech R. Detekční algoritmy pro klasifikaci neuronální aktivity. In *57. společný sjezd České a Slovenské společnosti pro klinickou neurofyziologii*, pages 52–52. MH Consulting, 2010. *Contribution: 10%*.

7
Index

8
Bibliography

Baunez C, Humby T, Eagle D. M, Ryan L. J, Dunnett S. B, and Robbins T. W. Effects of STN lesions on simple vs choice reaction time tasks in the rat: preserved motor readiness, but impaired response selection. *Eur. J. Neurosci.*, 13(8):1609–1616, Apr 2001.

Benjamini Y and Hochberg Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995. ISSN 00359246. DOI: 10.2307/2346101.

Brockwell P. J and Davis R. A. *Time Series: Theory and Methods (Springer Series in Statistics)*. Springer, 2nd ed. 1991. 2nd printing 2009 edition, April 2009. ISBN 1441903194.

Brown E. N, Barbieri R, Ventura V, Kass R. E, and Frank L. M. The time-rescaling theorem and its application to neural spike train data analysis. *Neural Comput.*, 14(2):325–346, February 2002. DOI: 10.1162/08997660252741149.

Brucke C, Kupsch A, Schneider G. H, Hariz M. I, Nuttin B, Kopp U, Kempf F, Trottenberg T, Doyle L, Chen C. C, Yarrow K, Brown P, and Kuhn A. A. The subthalamic region is activated during valence-related emotional processing in patients with Parkinson's disease. *Eur. J. Neurosci.*, 26(3):767–774, Aug 2007.

Cacioppo J. T, Petty R. E, Losch M. E, and Kim H. S. Electromyographic activity over facial muscle regions can differentiate the valence and intensity of affective reactions. *J Pers Soc Psychol*, 50(2): 260–268, Feb 1986.

Cambridge Electronic Design Limited . Spike2, version 6. URL `http://ced.co.uk/pru.shtml?spk6wglu.htm?id=6`.

Chatfield C. *The analysis of time series: an introduction*. CRC Press, Florida, US, 6th edition, 2004.

Cheeseman P and Stutz J. Bayesian classification (autoclass): theory and results. pages 153–180, 1996.

Daley D. J and Vere-Jones D. *An Introduction to the Theory of Point Processes, Volume 1 (2nd ed.)*. Springer, New York, June 2003. ISBN 0387955410.

Dayan P and Abbott L. F. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 12-01 2001. ISBN 0262041995.

Delescluse M and Pouzat C. Efficient spike-sorting of multi-state neurons using inter-spike intervals information. *Journal of neuroscience methods*, 150(1):16–29, 2006.

Eden U. T, Gale J. T, Amirnovin R, and Eskandar E. N. Characterizing the spiking dynamics of subthalamic nucleus neurons in Parkinson's disease using generalized linear models. *Front Integr Neurosci*, 6:28, 2012.

Eden U. T. Point Process Models for Neural Spike Trains. pages 43–51, 2008.

Efron B. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, January 1979.

Faraway J. J. *Practical Regression and Anova using R*. 2002.

Fawcett A. P, Dostrovsky J. O, Lozano A. M, and Hutchison W. D. Eye movement-related responses of neurons in human subthalamic nucleus. *Exp Brain Res*, 162(3):357–365, Apr 2005.

Fawcett A. P, Cunic D, Hamani C, Hodaie M, Lozano A. M, Chen R, and Hutchison W. D. Saccade-related potentials recorded from human subthalamic nucleus. *Clin Neurophysiol*, 118(1):155–163, Jan 2007.

Fee M. S, Mitra P. P, and Kleinfeld D. Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-gaussian variability. *Journal of neuroscience methods*, 69(2):175–188, 1996.

Fee M. S, Mitra P. P, and Kleinfeld D. Erratum: Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-gaussian variability (j. neurosci. methods 69 (1996) (175-188)) (pii: S0165020796000507). *Journal of neuroscience methods*, 71(2):233, 1997.

Feynman R, Leighton R. B, and Sands M. L. *The Feynman Lectures on Physics*. Addison-Wesley, 1963. 3 volumes.

Fisher R. A. The use of multiple measurements in axonomic problems. *The Annals of Eugenics*, 7(2):179–188, 1936.

Franke F, Natora M, Boucsein C, Munk M. H. J, and Obermayer K. An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes. *Journal of computational neuroscience*, pages 1–22, 2009.

Friston K. J, Holmes A. P, Worsley K. J, Poline J. P, Frith C. D, and Frackowiak R. S. J. Statistical parametric maps in functional imaging: A general linear approach. *Hum. Brain Mapp.*, 2(4):189–210, 1994. ISSN 1097-0193. DOI: 10.1002/hbm.460020402.

Frühholz S and Grandjean D. Towards a fronto-temporal neural network for the decoding of angry vocal expressions. *Neuroimage*, 62(3): 1658–1666, Sep 2012.

Galton F. Regression Towards Mediocrity in Hereditary Stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15: 246–263, 1886. ISSN 09595295. DOI: 10.2307/2841583.

Galton, Francis . Family Likeness in Stature. *Proceedings of the Royal Society of London*, 40:42–83, Jan 1886. DOI: 10.1098/rspl.1886.0009.

Gold C, Henze D. A, Koch C, and Buzsaki G. On the origin of the extracellular action potential waveform: A modeling study. *Journal of neurophysiology*, 95(5):3113–3128, May 2006.

Harris K. D, Henze D. A, Csicsvari J, Hirase H, and Buzsáki G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of neurophysiology*, 84(1):401–414, 2000.

Hastie T, Tibshirani R, and Friedman J. *The Elements of Statistical Learning (2nd ed.)*. New York: Springer-Verlag, 2009. ISBN 0-387-84857-6.

Herbst J. A, Gammeter S, Ferrero D, and Hahnloser R. H. R. Spike sorting with hidden Markov models. *Journal of neuroscience methods*, 174(1):126–134, 2008.

Herzog J, Volkmann J, Krack P, Kopper F, Pötter M, Lorenz D, Steinbach M, Klebe S, Hamel W, Schrader B, Weinert D, MÃijller D, Mehdorn H. M, and Deuschl G. Two-year follow-up of subthalamic deep brain stimulation in parkinson's disease. *Movement Disorders*, 18(11):1332–1337, 2003.

Hikosaka O, Takikawa Y, and Kawagoe R. Role of the basal ganglia in the control of purposive saccadic eye movements. *Physiological Reviews*, 80(3):953–978, July 2000. ISSN 1522-1210. URL http://physrev.physiology.org/content/80/3/953.abstract.

Huebl J, Schoenecker T, Siegert S, Brucke C, Schneider G. H, Kupsch A, Yarrow K, and Kuhn A. A. Modulation of subthalamic alpha activity to emotional stimuli correlates with depressive symptoms in Parkinson's disease. *Mov. Disord.*, 26(3):477–483, Feb 2011.

Hurley C. B and Earle D. *DendSer: Dendrogram Seriation: Ordering for Visualisation*, 2013. URL http://CRAN.R-project.org/package=DendSer. R package version 1.0.0.

Jacobs A. L, Fridman G, Douglas R. M, Alam N. M, Latham P. E, Prusky G. T, and Nirenberg S. Ruling out and ruling in neural codes. *Proc. Natl. Acad. Sci. U.S.A.*, 106(14):5936–5941, Apr 2009.

Karama S, Armony J, and Beauregard M. Film excerpts shown to specifically elicit various affects lead to overlapping activation foci in a large set of symmetrical brain regions in males. *PloS one*, 6(7), 2011. ISSN 1932-6203. DOI: 10.1371/journal.pone.0022343.

Kuhn A. A, Hariz M. I, Silberstein P, Tisch S, Kupsch A, Schneider G. H, Limousin-Dowsey P, Yarrow K, and Brown P. Activation of the subthalamic region during emotional processing in Parkinson disease. *Neurology*, 65(5):707–713, Sep 2005.

Lawrence M and Sarkar D. *qtbase: Interface between R and Qt*, 2012a. URL http://CRAN.R-project.org/package=qtbase. R package version 1.0.4.

Lawrence M and Sarkar D. *qtpaint: Qt-based Painting Infrastructure*, 2012b. URL http://CRAN.R-project.org/package=qtpaint. R package version 0.9.0.

Lawrence M and Wickham H. *plumbr: Mutable and Dynamic Data Models*, 2012. URL http://CRAN.R-project.org/package=plumbr. R package version 0.6.6.

Lewicki M. S. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4), November 1998. ISSN 0954-898X.

Maechler M, Rousseeuw P, Struyf A, Hubert M, and Hornik K. *cluster: Cluster Analysis Basics and Extensions*, 2012. URL http://CRAN.R-project.org/package=cluster. R package version 1.14.3.

Matsumura M, Kojima J, Gardiner T. W, and Hikosaka O. Visual and oculomotor functions of monkey subthalamic nucleus. *J. Neurophysiol.*, 67(6):1615–1632, Jun 1992.

McCullagh P and Nelder J. A. *Generalized linear models (Second edition)*. London: Chapman & Hall, 1989.

Nelder J. A and Wedderburn R. W. M. Generalized linear models. *Journal of the Royal Statistical Society, Series A, General*, 135:370–384, 1972.

Neuwirth E. *RColorBrewer: ColorBrewer Palettes*, 2011. URL `http://CRAN.R-project.org/package=RColorBrewer`. R package version 1.0-5.

Newcombe R. G. Interval estimation for the difference between independent proportions: comparison of eleven methods. *Stat Med*, 17(8):873–890, Apr 1998.

Ogata Y. Statistical Models for Earthquake Occurrences and Residual Analysis for Point Processes. *Journal of the American Statistical Association*, 83(401):9+, March 1988. ISSN 01621459. DOI: 10.2307/2288914.

Parkinson J. An Essay on the Shaking Palsy. *J Neuropsychiatry Clin Neurosci*, 14(2):223–236, May 2002. DOI: 10.1176/appi.neuropsych.14.2.223.

Pedoto G, Santaniello S, Fiengo G, Glielmo L, Hallett M, Zhuang P, and Sarma S. V. Point process modeling reveals anatomical non-uniform distribution across the subthalamic nucleus in Parkinson's disease. *Conf Proc IEEE Eng Med Biol Soc*, 2012:2539–2542, 2012.

Peron J, Fruhholz S, Verin M, and Grandjean D. Subthalamic nucleus: a key structure for emotional component synchronization in humans. *Neurosci Biobehav Rev*, 37(3):358–373, Mar 2013.

Pouzat C and Chaffiol A. On Goodness of Fit Tests For Models of Neuronal Spike Trains Considered as Counting Processes. 2008. URL `https://sites.google.com/site/spiketrainanalysiswithr/`.

Purves D. *Neuroscience*. Sinauer, 4th edition, July 2008. ISBN 9780878936977.

Quiroga R. Q, Nadasdy Z, and Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004.

R Core Team . *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

Rutishauser U, Schuman E. M, and Mamelak A. N. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of neuroscience methods*, 154(1-2):204–224, 2006.

Sahani M. *Latent Variable Models for Neural Data Analysis*. PhD thesis, California Institute of Technology, 1999. URL `http://www.gatsby.ucl.ac.uk/~maneesh/thesis/`.

Sarma S. V, Eden U. T, Cheng M. L, Williams Z. M, Hu R, Eskandar E, and Brown E. N. Using point process models to compare neural spiking activity in the subthalamic nucleus of Parkinson's patients and a healthy primate. *IEEE Trans Biomed Eng*, 57(6):1297–1305, Jun 2010.

Senn S. A conversation with John Nelder. *Statist. Sci.*, 18(1):118–131, 2003. ISSN 0883-4237.

Serranová T, Jech R, Dušek P, **Sieger T**, Růžička F, Urgošík D, and Růžička E. Subthalamic nucleus stimulation affects incentive salience attribution in Parkinson's disease. *Mov. Disord.*, 26(12):2260–2266, Oct 2011. *Contribution: 5%, citations: 5*.

Serranová T, **Sieger T**, Dušek P, Růžička F, Urgošík D, Růžička E, Valls-Sole J, and Jech R. Sex, food and threat: startling changes after subthalamic stimulation in Parkinson's disease. *Brain Stimul*, 6(5): 740–745, Sep 2013. *Contribution: 10%, citations: 1*.

Shalizi C. R. *Advanced Data Analysis from an Elementary Point of View*. 2012. URL `http://www.stat.cmu.edu/~cshalizi/uADA/12/`.

Sieger T, Cecilia B, Serranová T, Wild J, Novák D, Růžička F, Urgošík D, Růžička E, Gaymard B, and Jech R. Basal ganglia neuronal activity during scanning eye movements in Parkinson's disease. *PLoS ONE*, 8(11):e78581, 2013. DOI: 10.1371/journal.pone.0078581. *Contribution: 40%*.

Simoncelli E. P and Olshausen B. A. Natural image statistics and neural representation. *Annu. Rev. Neurosci.*, 24:1193–1216, 2001.

Simpson D. M, Infantosi A. F. C, and Rosas D. A. B. Estimation and significance testing of cross-correlation between cerebral blood flow velocity and background electro-encephalograph activity in signals with missing samples. *Medical and Biological Engineering and Computing*, 39(4):428–433, 2001.

Takahashi S, Anzai Y, and Sakurai Y. A new approach to spike sorting for multi-neuronal activities recorded with a tetrode - how ica can be practical. *Neuroscience research*, 46(3):265–272, 2003a.

Takahashi S, Anzai Y, and Sakurai Y. Automatic sorting for multi-neuronal activity recorded with tetrodes in the presence of overlapping spikes. *Journal of neurophysiology*, 89(4):2245–2258, 2003b.

Temel Y, Blokland A, Steinbusch H. W, and Visser-Vandewalle V. The functional role of the subthalamic nucleus in cognitive and limbic circuits. *Prog. Neurobiol.*, 76(6):393–413, Aug 2005.

Temel Y, Blokland A, Ackermans L, Boon P, Kranen-Mastenbroek vV. H, Beuls E. A, Spincemaille G. H, and Visser-Vandewalle V. Differential effects of subthalamic nucleus stimulation in advanced Parkinson disease on reaction time performance. *Exp Brain Res*, 169(3):389–399, Mar 2006.

The Mind Project . URL `http://www.mind.ilstu.edu/`.

Truccolo W, Eden U. T, Fellows M. R, Donoghue J. P, and Brown E. N. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *J. Neurophysiol.*, 93(2):1074–1089, Feb 2005.

Tsunoda M, Kurachi M, Yuasa S, Kadono Y, Matsui M, and Shimizu A. Scanning eye movements in schizophrenic patients. Relationship to clinical symptoms and regional cerebral blood flow using 123I-IMP SPECT. *Schizophr. Res.*, 7(2):159–168, Jul 1992.

Tufte E. R. *Envisioning Information*. Graphics Press, Cheshire, Connecticut, 1990. ISBN 0-9613921-1-8.

Tufte E. R. *Visual Explanations*. Graphics Press, Cheshire, Connecticut, 1997. ISBN 0-9613921-2-6.

Tufte E. R. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 2001. ISBN 0-9613921-4-2.

Tufte E. R. *Beautiful Evidence*. Graphics Press, LLC, first edition, May 2006. ISBN 0-9613921-7-7.

Ventura V, Cai C, and Kass R. E. Statistical assessment of time-varying dependency between two neurons. *Journal of neurophysiology*, 94(4):2940–2947, October 2005a. ISSN 0022-3077. DOI: 10.1152/jn.00645.2004.

Ventura V, Cai C, and Kass R. E. Trial-to-trial variability and its effect on time-varying dependency between two neurons. *Journal of neurophysiology*, 94(4):2928–2939, October 2005b. ISSN 0022-3077. DOI: 10.1152/jn.00644.2004.

Vinh N, Epps J, and Bailey J. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1073–1080, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. DOI: 10.1145/1553374.1553511.

Wickham H and Chang W. *devtools: Tools to Make Developing R Code Easier*, 2012. URL http://CRAN.R-project.org/package=devtools. R package version 0.8.

Wild J, Prekopcsák Z, **Sieger T**, Novák D, and Jech R. Performance comparison of extracellular spike sorting algorithms for single-channel recordings. *Journal of Neuroscience Methods*, 203(2):369–376, January 2012. ISSN 01650270. DOI: 10.1016/j.jneumeth.2011.10.013. *Contribution: 15%, citations: 1*.

Wolters E. C. h. Deep brain stimulation and continuous dopaminergic stimulation in advanced Parkinson's disease. *Parkinsonism Relat. Disord.*, 13 Suppl:18–23, Sep 2007.

Xie Y, Hofmann H, Cook D, Cheng X, Schloerke B, Vendettuoli M, Yin T, Wickham H, and Lawrence M. *cranvas: Interactive Statistical Graphics Based on Qt*, 2013. URL http://github.com/ggobi/cranvas. R package version 0.8.2.

Zvára K. *Regrese*. Martfyzpress, 2008.